HL-LHC ATLAS 4D tracking
ACTS CKF timing reconsustruction

Rodrigo Estevam de Paula

Marco Aurelio Lisboa Leite

Vitor Heloiz Nascimento

30 July, 2024

Quick Updates

ACTS Developers Workshop 2024

- The workshop aims to cover multiple objectives:
  - Discuss the development of tools (past and future implementations)
  - Hands-on tutorials of techniques
  - Present ongoing studies using ACTS
- Will happen 18th-21st November
- It is possible to submit for a PhD studies report at the event
  - Deadline for registration is September 15
  - https://indico.cern.ch/event/1397634/overview

ACTS Developers Workshop 2024

18 – 21 de nov. de 2024
Chateau de Bossey
Fuso horário America/Sao_Paulo

Visão Geral
Tabela de Horários
Registration
Lista de Contribuição
Lista de participantes
Fee Payment

Contact
Connie.Potter@cern.ch
Noemi.Calace@cern.ch
Andreas.Salzburger@c...
Paul.Gessinger@cern.ch

The acts project

supported by

cooperations

Venue

- Location: Chateau de Bossey, in the forest of Bogis-Bossey, Vaud, Switzerland
  - Transportation to/from CERN will be provided on the first/last day
  - Salle Lausanne/Geneva (AM & PM)
  - Dates: 18th - 21st November 2024

Registration Fees

The deadline for registration is 15 September 2024

Two categories are available.

- **RESIDENTIAL**: An all-inclusive package covers
  - room: single room with private bathroom
  - board: 3x breakfast, 4x lunch, 3x dinner + coffee breaks
  - transportation: bus from CERN to venue on Monday / from venue to CERN on Thursday
    - FEE is **740 CHF**

- **NON-RESIDENTIAL**: For those with local obligations who cannot stay full-time.
  - board: 4x lunch, 3x dinner + coffee breaks
    - FEE is **350 CHF**

CMS paper on tracking

Line Segment Tracking: Improving the Phase 2 CMS High Level Trigger Tracking with a Novel, Hardware-Agnostic Pattern Recognition Algorithm

E Vourliotis[1a] and P Chang[2], P Elmer[3], Y Gu[1], J Guiang[1], V Krutelyov[1], B V Sathia Narayanan[1], G Niendorf[4], M Reid[4], M Silva[2], A Rios Tascon[3], M Tadel[1], P Wittich[4], A Yagil[1]
on behalf of the CMS Collaboration

[1]University of California San Diego, CA, US
[2]University of Florida, FL, US
[3]Princeton University, NJ, US
[4]Cornell University, NY, US

E-mail: [a]emmanouil.vourliotis@cern.ch

Abstract. Charged particle reconstruction is one the most computationally heavy components of the full event reconstruction of Large Hadron Collider (LHC) experiments. Looking to the future, projections for the High Luminosity LHC (HL-LHC) indicate a superlinear growth for required computing resources for single-threaded CPU algorithms that surpass the computing resources that are expected to be available. The combination of these facts creates the need for efficient and computationally performant pattern recognition algorithms that will be able to run in parallel and possibly on other hardware, such as GPUs, given that these become more and more available in LHC experiments and high-performance computing centres. Line Segment Tracking (LST) is a novel such algorithm which has been developed to be fully parallelizable and hardware agnostic. The latter is achieved through the usage of the Alpaka library. The LST algorithm has been tested with the CMS central software as an external package and has been used in the context of the CMS HL-LHC High Level Trigger (HLT). When employing LST for pattern recognition in the HLT tracking, the physics and timing performances are shown to improve with respect to the ones utilizing the current pattern recognition algorithms. The latest results on the usage of the LST algorithm within the CMS HL-LHC HLT are presented, along with prospects for further improvements of the algorithm and its CMS central software integration.

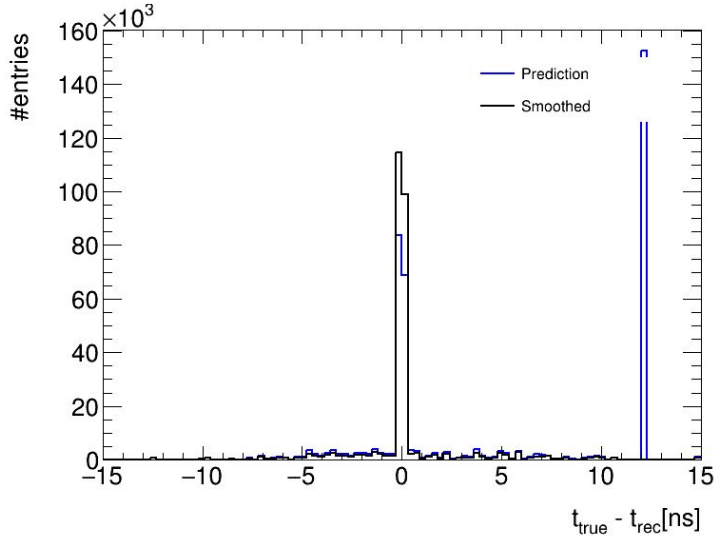
Arxiv link

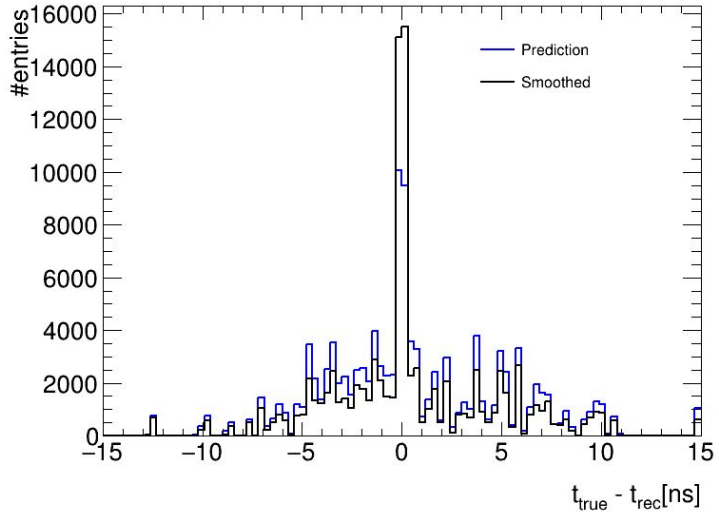
- Proposes a new track reconstruction method, the LST, and compares it to CKF
- Aims to be time efficient and hardware agnostic

ACTS Time reconstruction

Investigating weird residual bin at residual histogram

- In the previous meeting there was a weird residual bin for predicted samples
- This bin is composed of a singe value of ~12,25 ns repeated multiple times and all comes from event 96 (1 event out of 100)
- Filtering the event solves it out
  - Will investigate further later





Beam conditions at the HL-LHC

- HGTD TDR states (pag 5):

A major challenge for the ITk is the pileup suppression in the primary vertex and in the object reconstruction in this high pileup environment, especially in the end-cap region. The luminous region will have an estimated Gaussian spread of 30 to 60 mm along the beam axis (z direction1 .) The width in time could range from 175 to 260 ps. The case considered in this report is the “nominal” scenario, with Gaussian standard deviation of approximately 50 mm along the beam axis and spreads of 175 ps in time.

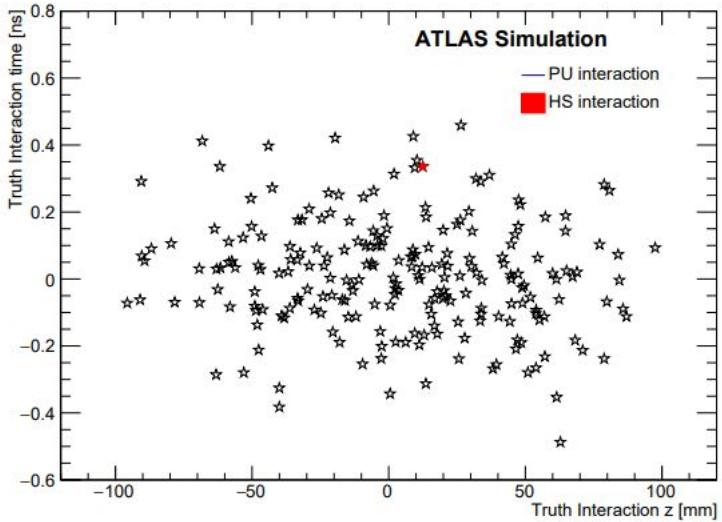
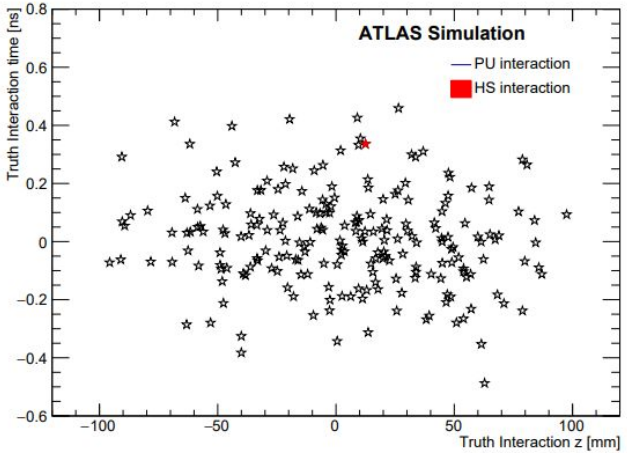


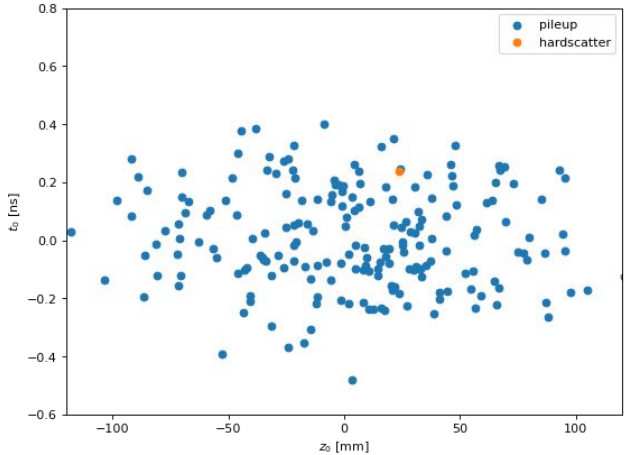
Figure: Visualisation of the truth interactions in a single bunch crossing in the z–t plane, showing the simulated Hard Scatter (HS) ttbar event interaction (red) with pileup interactions superimposed (black) for <μ> = 200

Simulating HL-LHC beam with ACTS Pythia8

- Managed to simulate the same scenario as the TDR
  - z0 ~ N(0,50 mm), t0 ~ N(0,175ps)
- Simulated ttbar events with 1000 events
- Using Fatras
  - Got some propagation errors which needs to be addressed, but its not a blocking problem

```
addPythia8(
    s,
    hardProcess=["Top:qqbar2ttbar=on"],
    npileup=200,
    vtxGen=acts.examples.GaussianVertexGenerator(
        stddev=acts.Vector4(0.0125 * u.mm,
                            0.0125 * u.mm,
                            50 * u.mm,
                            0.175 * u.ns),
        mean=acts.Vector4(0, 0, 0, 0),
    ),
    rnd=rnd,
    outputDirRoot=outputDir,
)
```

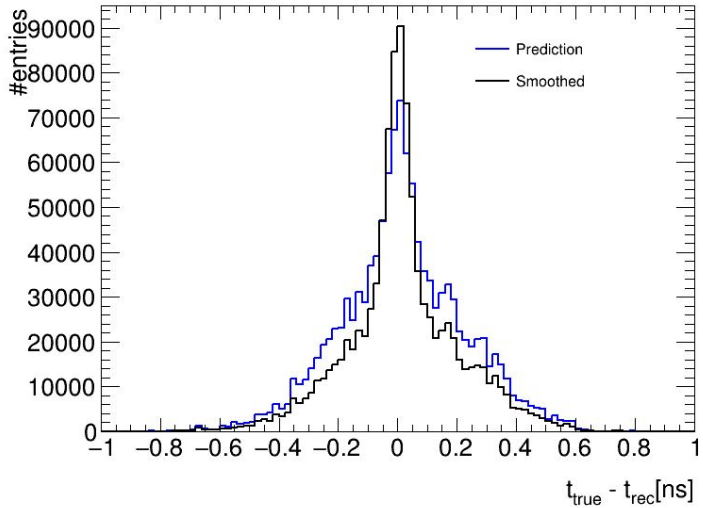




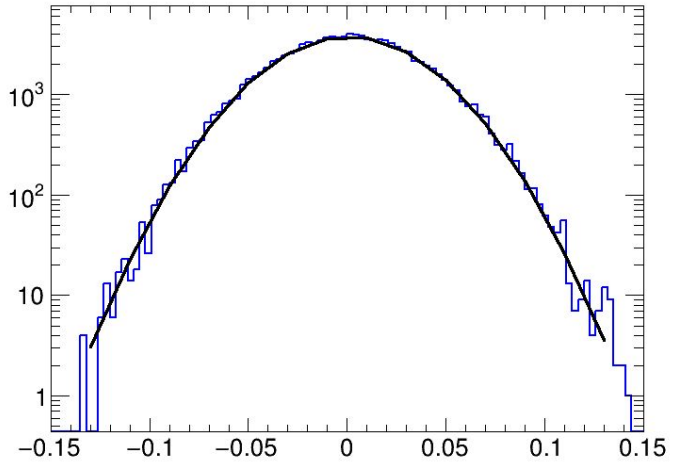
ttbar <u> = 0 residue plots

- Needed to filter 3 events, for the same reason as slide 6
- Residue spams from -0.8 to 0.8 ns
  - This is the whole range of our vertex generation, so it's no much of a positive result
- HGTD volumes presents the resolution expected by specification (35 ps)

whole detector



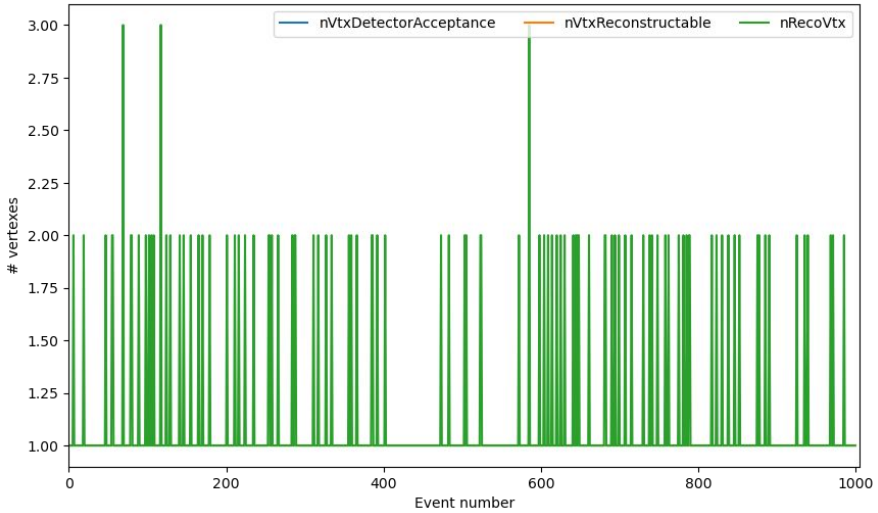
HGTD volumes



smoothed fitted (σ = 35 ps)

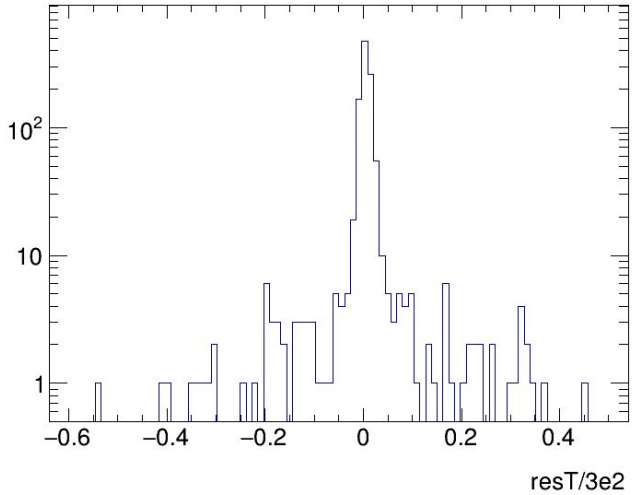
ttbar <u> = 0 vertex reconstruction efficience

- Without pileup there's only one vertex to be reconstructed
- Sometimes the reconstruction splits the vertexes and reconstructs 2 or 3
  - Need further analysis to see if the recovered vertexes are close in space and time

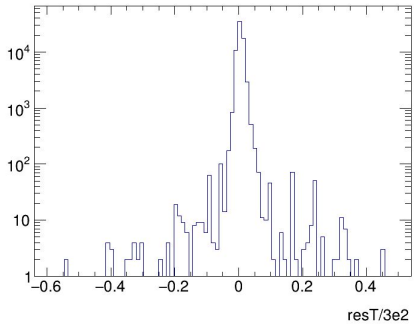


ttbar <u> = 0 vertex reconstruction residue

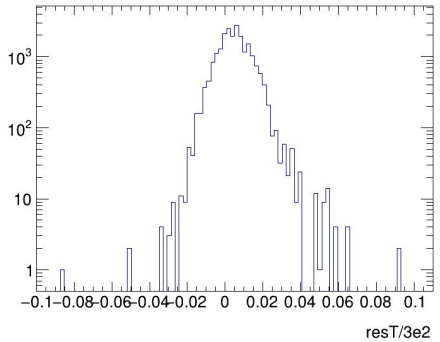
- t0 resolution is slightly better than whole time estimation, spams from -0.6 to 0.4
- If we isolate barrel region tracks ($|\eta| < 2.4$) from endcap region ($|\eta| > 2.4$) we get different distributions
  - Endcap region has better resolution, indicating that HGTD tagging is improving the reconstruction



whole detector



barrel ($|\eta| < 2.4$)

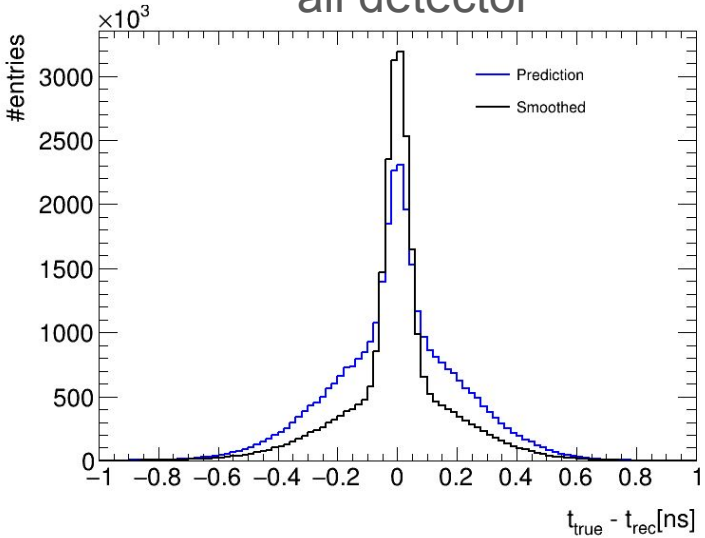


endcaps ($|\eta| > 2.4$)

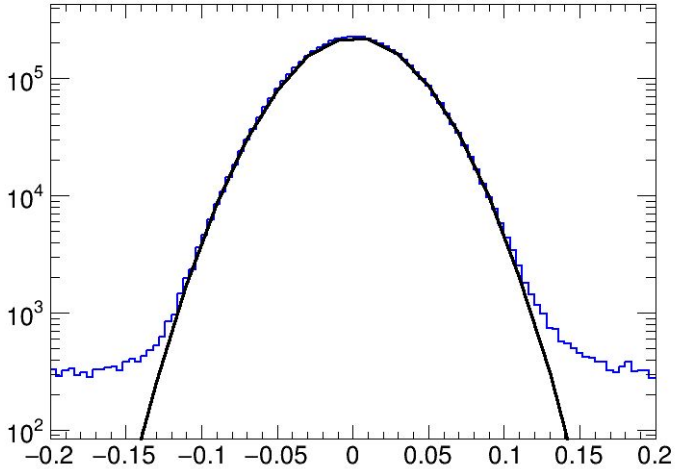
ttbar <u> = 200 residue plots

- Better separation between smoothed and predicted samples
  - Indicating the smoothing is important to mitigate pileup
- Same resolution as the simulation without pileup
- HGTD volumes show expected resolution but with pronounced tails

all detector



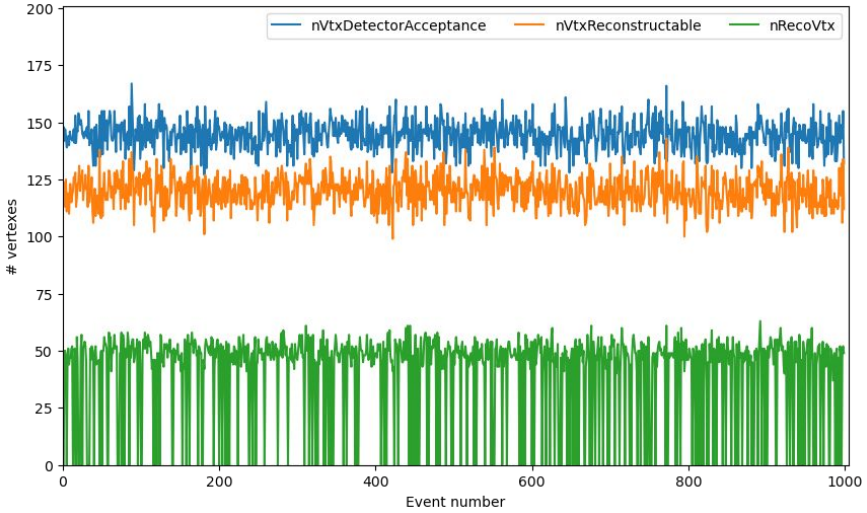
HGTD volumes

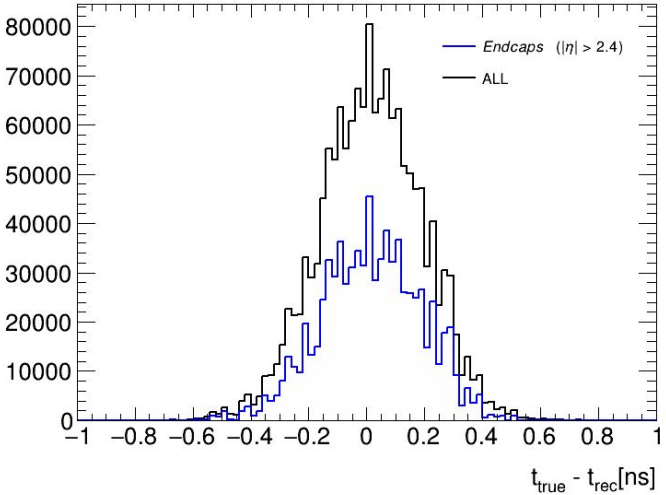


smoothed fitted (σ = 35 ps)

ttbar <u> = 200 vertex reconstruction

- Not all of the 201 vertex were reconstructed
  - Limited by detector geometry and preselection parameters on the reconstruction
- t0 resolution spams from -0.6 to 0.6
- No significant difference from endcaps to barrel region
  - Probably due to pileup tracks not tagged by HGTD





Next steps

Improve CKF timing integration

- Use Geant4 for
- Establish value for first estimate covariance
- Evaluate tracking efficiency performance
- Write an article about it (?) -> Submit to ACTS workshop ?

Explore ACTS (on going)

- Continue investigating the CKF until a full understanding
- Study the implementation of the GSF in the core library
- Start investigate ExaTrk plugin to test ML based reconstruction methods
    - Get a general understanding of these methods but not to jump to it right away

Follow HGTD ACTS integration campaign

- Start AQP

Theoretical Study

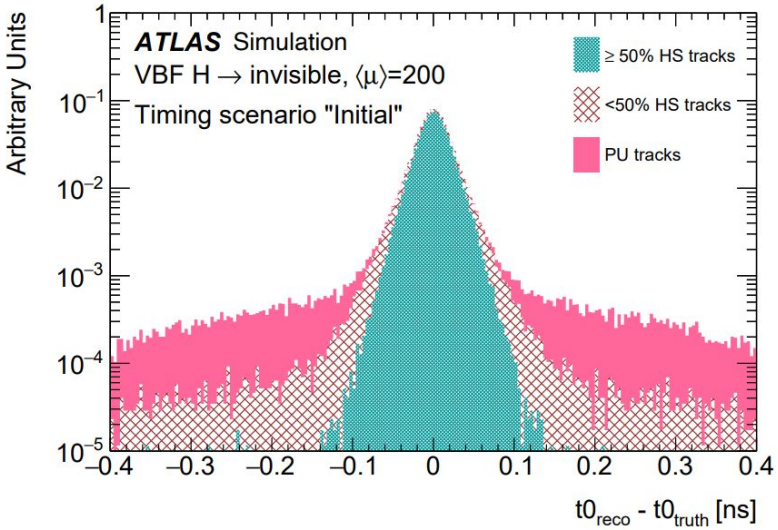
- H. Kolanosky, Particle Detectors (2020)
    - Next: Chapters 8-9
- Advance on the study notes
- Read papers of systematic review
    - Search more papers on high dimensional combinatorial optimization problems

Backup

Importance of primary vertex time (t0) determination

- HGTD TDR states that (pag 37):

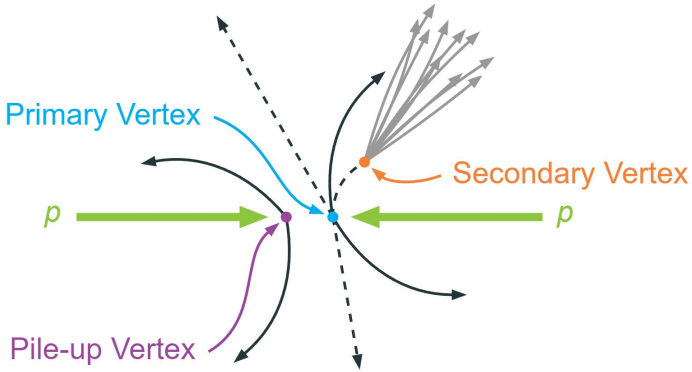
Due to the large uncertainty of the longitudinal impact parameter for tracks in the forward region (Figure 2.6), the association of tracks to nearby vertices purely based on spatial information is ambiguous in high-pileup environments, especially for low transverse momentum tracks. The ability to determine the time of the primary vertex of the hard-scatter process, here denoted as t0, provides a new handle to enhance the capability of the ATLAS detector to remove pileup tracks contaminating physics objects originating from the hard-scatter vertex.

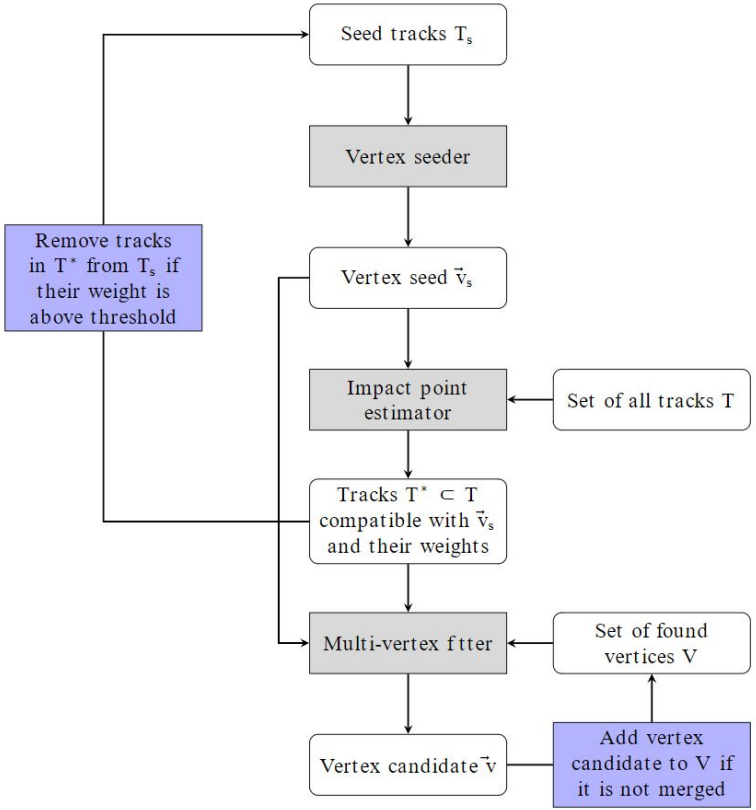


Vertex t0 resolution separately for various cases, where “HS” (“PU”) stands for hard scatter (pileup). **Only track clusters tagged by HGTD were evaluated.**

Vertex reconstruction

- **Vertex finding:** cluster together the origin of tracks
- **Vertex fitting:** assume helicoidal (or linear) trajectory to enhance the estimate of the vertex
- Will deepen this explanation in future meetings





Time extrapolation

- ACTS time propagation is described in this paper: (Klimpel, 2021)

$$\frac{dt}{ds} = \frac{1}{v} = \frac{E}{p} = \frac{\sqrt{m^2 + p^2}}{p} = \sqrt{m^2/p^2 + 1}$$

- In vacuum the extrapolation becomes: $t_{n+1} = t_n + h\sqrt{\frac{m^2}{p^2} + 1}.$

- Necessary to include particle mass in the state vector
- The propagation implemented in ACTS is done in vacuum, but its also possible to use Range Kutta to make this extrapolation with more precision
- This propagation is already included in the CKF but as no estimates or measurements are evaluated, this parameter is not properly reconstructed

Time measurements and smearing

- At the digitization step, ACTS uses a geometry config file to simulate smearing of measurements
- We included the time parameter at volumes 2 and 25, which represent HGTD
  - Other volumes just measure l_0 and l_1

$$\vec{x} = (l_0, l_1, \phi, \theta, q/p, t)^T$$

- Got the smearing time with HGTD group

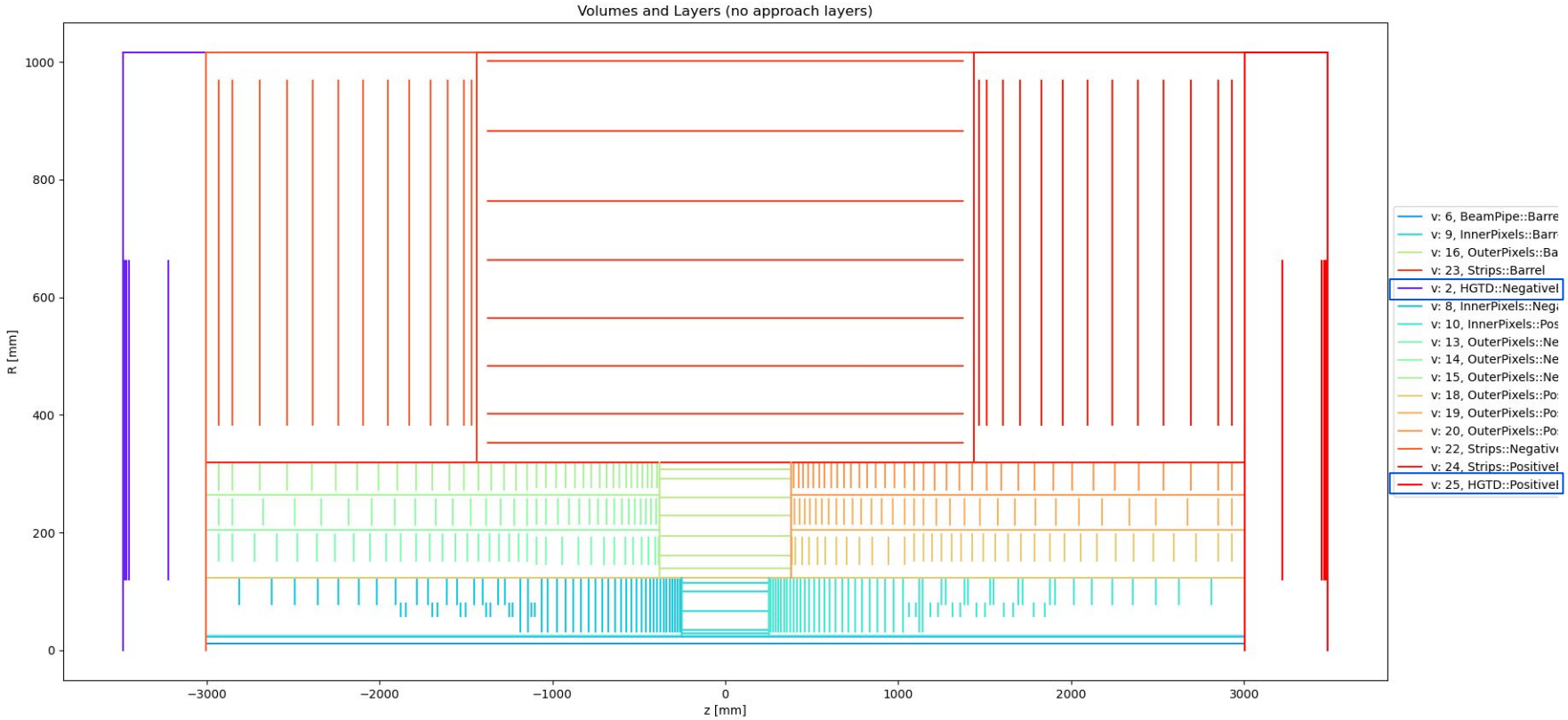
$$\sigma = 35 \text{ ps} \qquad \sigma_t = s \times \sigma$$

$$s = 299792458 \text{ mm/s} \qquad \sigma_t = 10.5 \text{ mm}$$


```
{
    "acts-geometry-hierarchy-map": {
        "format-version": 0,
        "value-identifier": "digitization-configuration"
    },
    "entries": [
        {
            "volume": 2,
            "value": {
                "smearing": [
                    {
                        "index": 0,
                        "mean": 0.0,
                        "stddev": 0.37527767,
                        "type": "Gauss"
                    },
                    {
                        "index": 1,
                        "mean": 0.0,
                        "stddev": 0.37527767,
                        "type": "Gauss"
                    },
                    {
                        "index": 5,
                        "mean": 0.0,
                        "stddev": 10.5,
                        "type": "Gauss"
                    }
                ]
            }
        },
        ...
    }
```

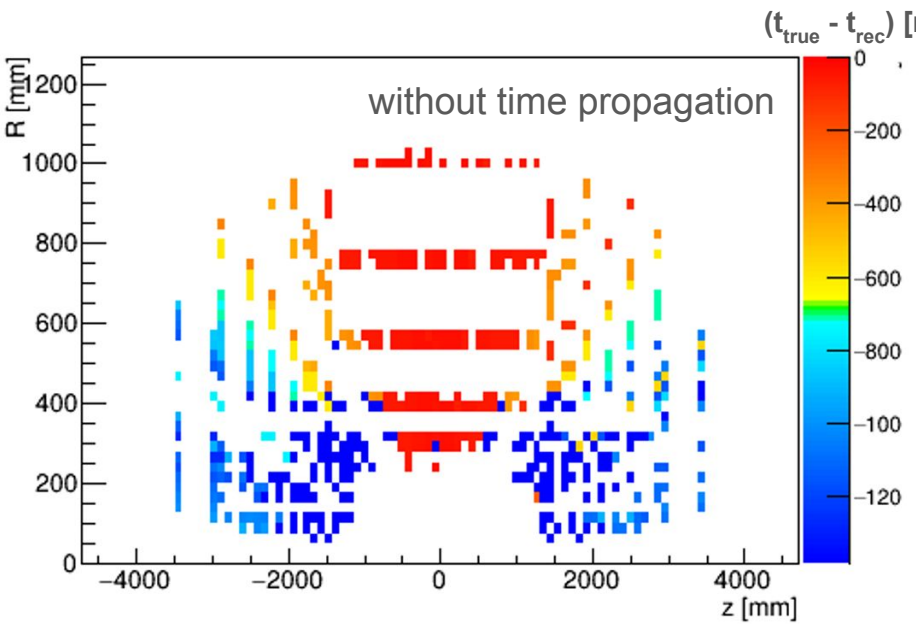
ITK + HGTD geometry at ACTS

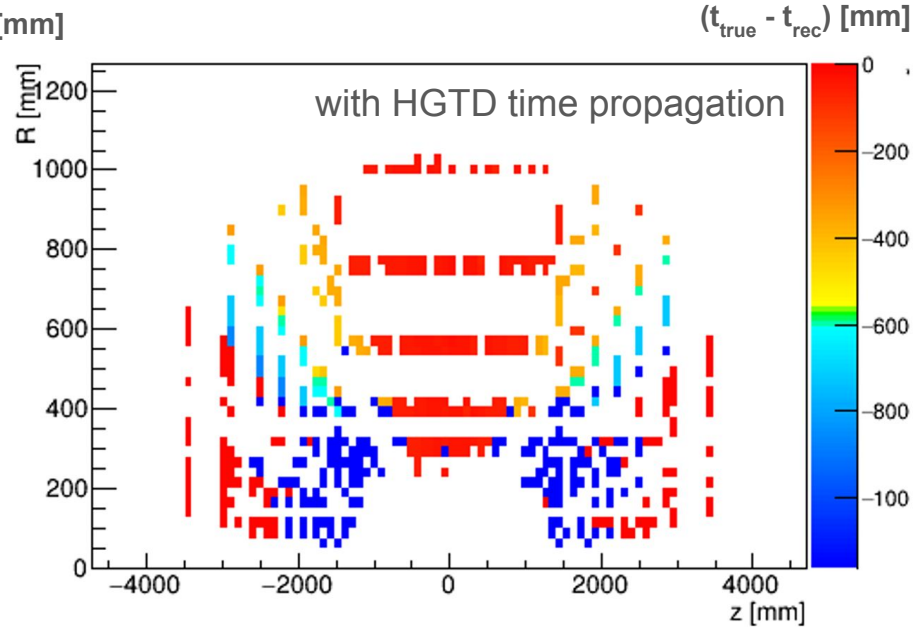
- HGTD endcaps are mapped to volume 2 and 25 of our geometry file



Simulation with particle guns

- Aimed to analyse the time residue (t_{true} - t_{rec}) before and after the smearing inclusion
- 100 events with particle gun of a single muon distributed uniformly between eta -4 and 4
- Regional plots below show the **mean time residue after smoothing** for each region





Changing first time estimate

- The actual CKF implementation uses the first measurement as the initial estimate
- As there's no time reading at the ITk sensors, the first estimate will have the time coordinate being 0.

What would be a good first estimate for the time at the first hit?

- First idea: distance between first hit and collision centre

$$t_1 = \sqrt{x_1^2 + y_1^2 + z_1^2}$$

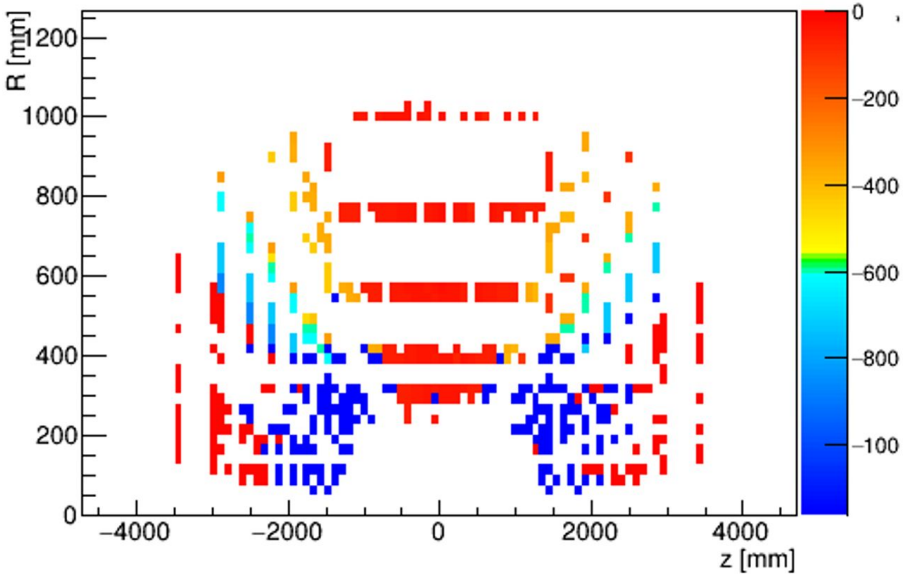
- Works well if p >> m then $t_{n+1} = t_n + h$
- And if the particle is generated at the position and time 0

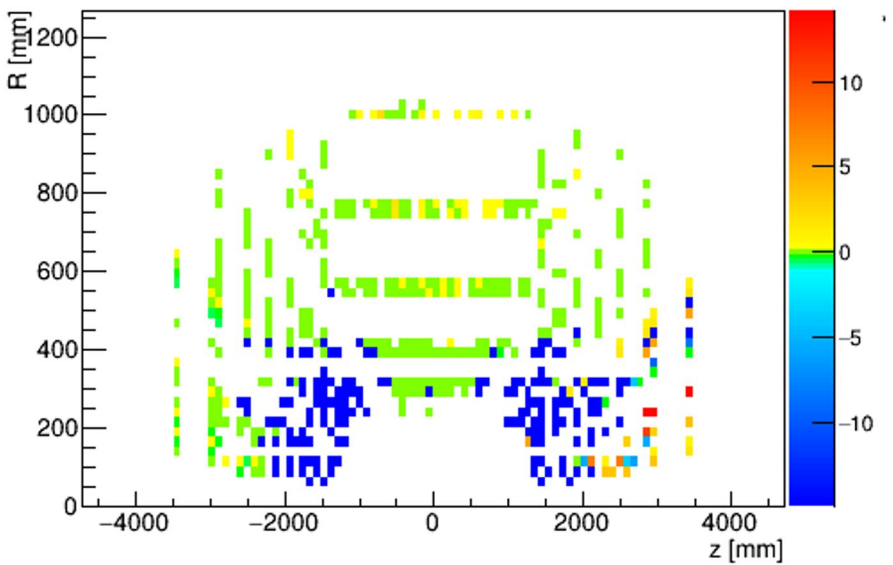
Changing first time estimate (code implementation)

```
// acts/Core/include/Acts/TrackFinding/CombinatorialKalmanFilter.hpp
class CombinatorialKalmanFilter {
 private:
  class Actor {
   public:
    void createSourceLinkTrackStates(const Acts::GeometryContext& gctx,
                                     result_type& result,
                                     const BoundState& boundState,
                                     std::size_t prevTip,
                                     source_link_iterator_t slBegin,
                                     source_link_iterator_t slEnd) const {

        ...

        if (it == slBegin) {
          // only set these for first
          auto predicted = boundParams.parameters();

          const auto freeParams = transformBoundToFreeParameters(ts.referenceSurface(),gctx, boundParams.parameters());
          if(!ts.hasPrevious() && predicted(5) == 0){
            ACTS_VERBOSE("Dont have previous");
            predicted(5) = sqrt(freeParams(0)*freeParams(0) + freeParams(1)*freeParams(1) + freeParams(2)*freeParams(2));
            ACTS_VERBOSE("Initial Parameters Setting:"<<predicted);
          }
          ts.predicted() = predicted;
          if (boundParams.covariance()) {
            ts.predictedCovariance() = *boundParams.covariance();
          }
          ts.jacobian() = jacobian;

          ...
```

Simulation with particle guns

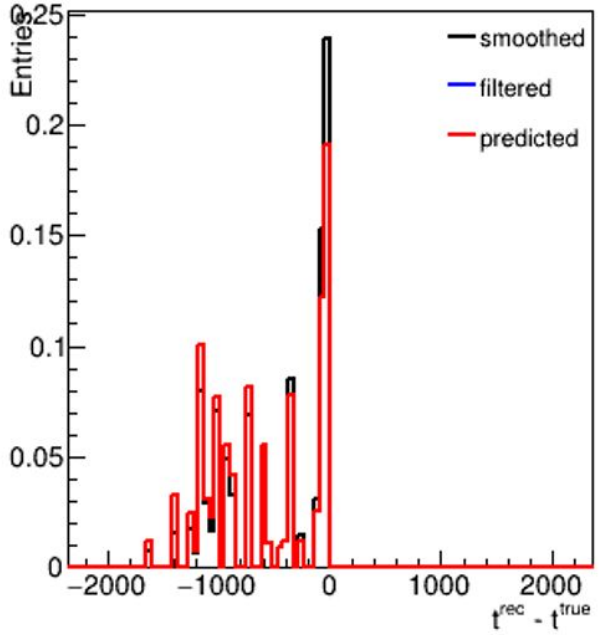
- Aimed to analyse the time residue (t_{true} - t_{rec}) before and after the smearing inclusion
- 100 events with particle gun of a single muon distributed uniformly between eta -4 and 4
- Regional plots below show the **mean time residue after smoothing** for each region

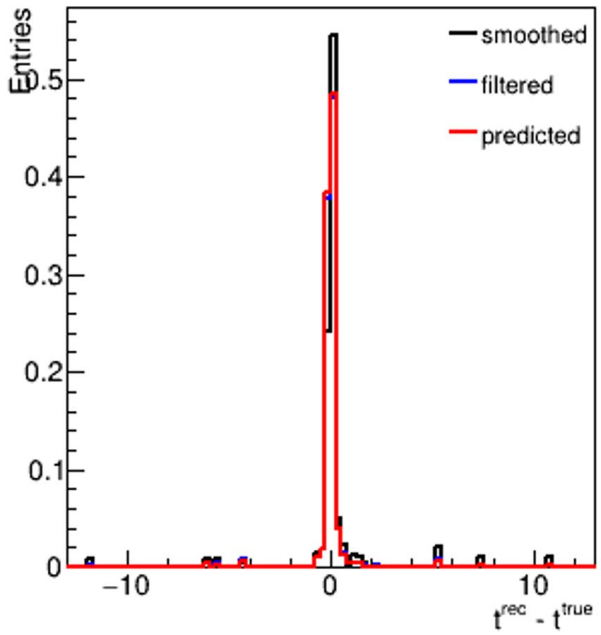




Performance of particle guns reconstruction

- For this scenario, the residue now is centred at zero with a variance lower than HGTD resolution
- Outliers happen because of HGTD resolution being way bigger than the prediction error





Simulation of ttbar events

- For a more complex scenario, we used ttbar events(Top: qq'->tt')
  - Jets (particle sprays) to the frontal region

```
addPythia8(
    s,
    hardProcess=["Top:qqbar2ttbar=on"],
    npileup=200,
    vtxGen=acts.examples.GaussianVertexGenerator(
        mean=acts.Vector4(0, 0, 0, 0),
        stddev=acts.Vector4(0.0125 * u.mm, 0.0125 * u.mm, 55.5 * u.mm, 5.0 * u.ns),
    ),
    rnd=rnd,
    outputDirRoot=outputDir,
)
```

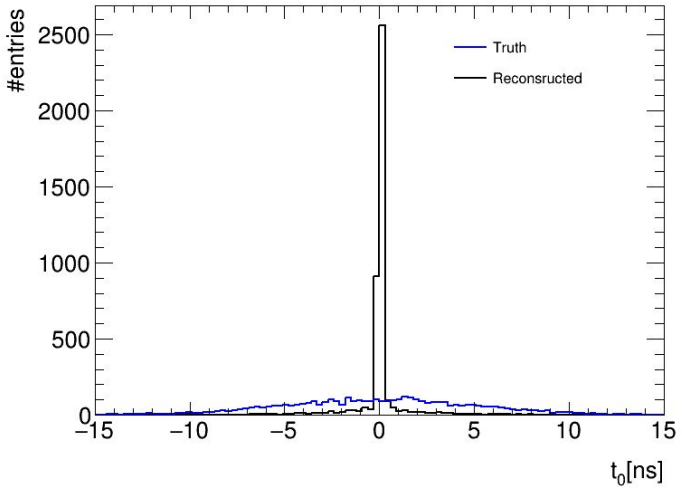
- Now particles aren't only generated at instant 0 (spread of 5 ns), making our first estimate inaccurate
  - HGTD has an important role in fixing the time prediction for tracks generated at t0 != 0

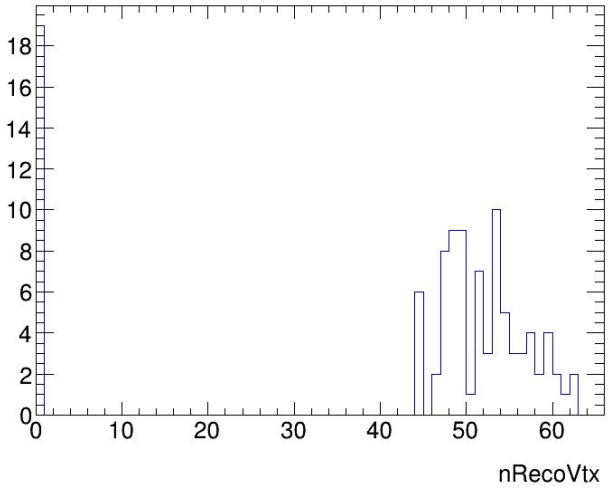
```
addVertexFitting(
    s,
    field,
    vertexFinder=VertexFinder.Iterative,
    outputDirRoot=outputDir,
)
```

Also added Vertex reconstruction

ttbar <u> = 200 vertex reconstruction

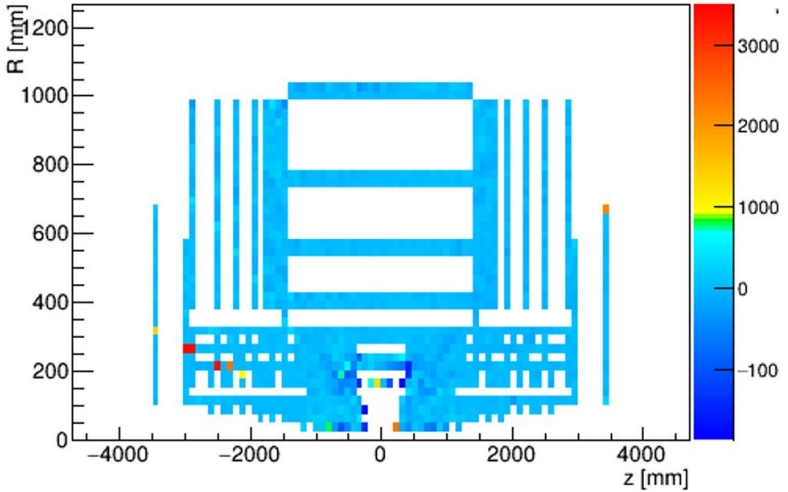
- Reconstruction fails to reconstruct vertexes
  - Error in the scale of ns
  - From 200 vertex only ~60 were reconstructed per event
- Tried to filter only particles with HGTD hits (eta > 2.4 and high pT) but the reconstruction still doesn't work

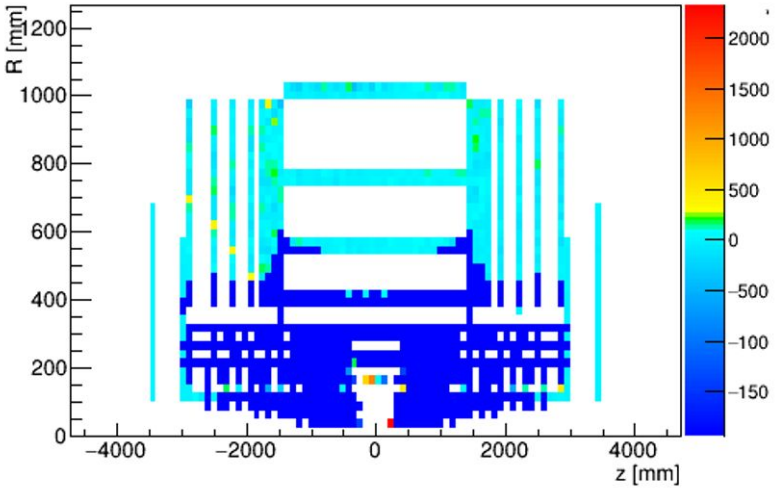




Points of improvement

- Use more accurate event generation time smearing
  - For now, vertexes are generated with t_0 ~ N(0,5ns), but that the stddev is way higher than it should be
- Improve (understand) the smoothing step of the CKF
  - Have to fix weird behaviour where smoothed samples are worse than filtered ones





- Need to understand Vertex reconstruction methods