HL-LHC ATLAS 4D tracking
ACTS CKF timing reconsustruction

Rodrigo Estevam de Paula

Marco Aurelio Lisboa Leite

Vitor Heloiz Nascimento

19 July, 2024

Quick Updates

HGTD Authorship Qualification Project

- Waiting for approval of HGTD project leaders
- We decided on the following task abstract

The first part of the QT consists of updating the raw data object representing a hit in HGTD (HGTD_RDO) to express time of arrival (TOA) as an integer value, as it will be output by the Altiroc chip. The necessary changes to downstream components (cluster building) to interpret the TOA appropriately will be included as well.

The second part consists of developing and testing a combinatorial approach for the HGTD reconstruction, within which tracks reconstructed in the Inner Tracker (ITk) will be used to extrapolate to HGTD and identify compatible hits with time information. Compared to the iterative reconstruction currently implemented, the combinatorial version should make use of the time measured in HGTD hits to identify outliers during the track extension step and reject them early. This should reduce the amount of "confusion" in the track extension, in which close-by pileup hits are accidentally chosen by the algorithm instead of primary hits left by a particle. The combinatorial approach will be implemented using ACTS based tracking tools within Athena and its performance will be studied.

Search for new (faster) reconstruction methods

- There's already another group on ATLAS working with the state of the art GNN based track reconstruction
- However, there are new methods for combinatorial optimization problems that we can explore
- Here is an example of a new method proposed this year:
  - Arxiv link

Distributed Constrained Combinatorial Optimization leveraging Hypergraph Neural Networks

Nasimeh Heydaribeni[1,*], Xinrui Zhan[1], Ruisi Zhang[1], Tina Eliassi-Rad,[2], and Farinaz Koushanfar[1]

[1]Department of Electrical and Computer Engineering, University of California, San Diego
[2]Khoury College of Computer Sciences, Northeastern University, Boston, MA, USA,
nheydaribeni, x5zhan, ruz032, fkoushanfar @ucsd.edu; t.eliassirad@northeastern.edu
[*]Corresponding Author

Abstract

Scalable addressing of high dimensional constrained combinatorial optimization problems is a challenge that arises in several science and engineering disciplines. Recent work introduced novel application of graph neural networks for solving quadratic-cost combinatorial optimization problems. However, effective utilization of models such as graph neural networks to address general problems with higher order constraints is an unresolved challenge. This paper presents a framework, HypOp, which advances the state of the art for solving combinatorial optimization problems in several aspects: (i) it generalizes the prior results to higher order constrained problems with arbitrary cost functions by leveraging hypergraph neural networks; (ii) enables scalability to larger problems by introducing a new distributed and parallel training architecture; (iii) demonstrates generalizability across different problem formulations by transferring knowledge within the same hypergraph; (iv) substantially boosts the solution accuracy compared with the prior art by suggesting a fine-tuning step using simulated annealing; (v) shows a remarkable progress on numerous benchmark examples, including hypergraph MaxCut, satisfiability, and resource allocation problems, with notable run time improvements using a combination of fine-tuning and distributed training techniques. We showcase the application of HypOp in scientific discovery by solving a hypergraph MaxCut problem on NDC drug-substance hypergraph. Through extensive experimentation on various optimization problems, HypOp demonstrates superiority over existing unsupervised learning-based solvers and generic optimization methods.

“Internal” updates

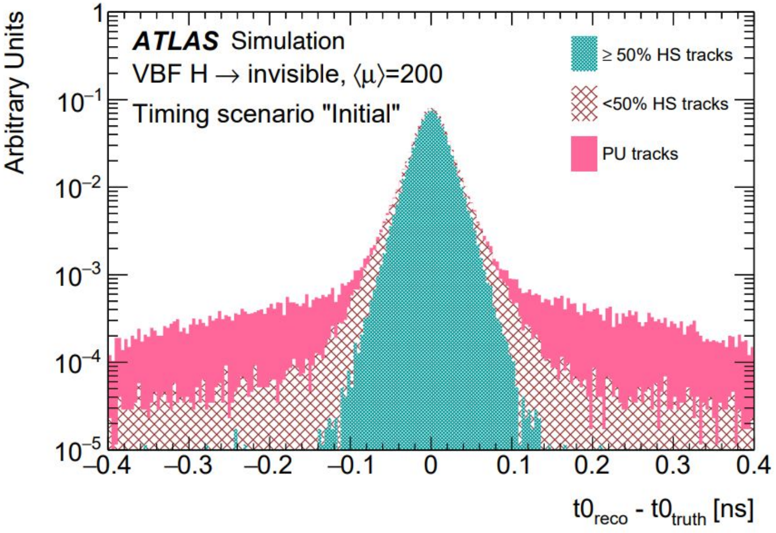
- Finished the mandatory courses 😃
- Started updating the study notes
  - The focus now is to describe the CKF and how’s implemented on ACTS
  - Will extend to the other steps of the reconstruction process (seeding, vertexing, etc)
- Will extend the systematic review to find new machine learning methods to solve high dimensional combinatorial problems (as seen on the previous slide)

ACTS Time reconstruction

Importance of primary vertex time (t0) determination

- HGTD TDR states that (pag 37):

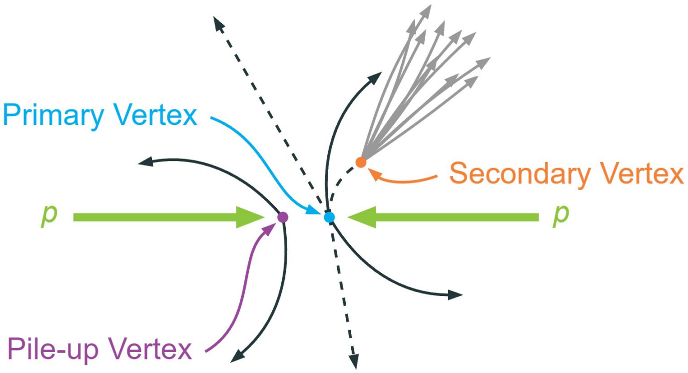
Due to the large uncertainty of the longitudinal impact parameter for tracks in the forward region (Figure 2.6), the association of tracks to nearby vertices purely based on spatial information is ambiguous in high-pileup environments, especially for low transverse momentum tracks. The ability to determine 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.

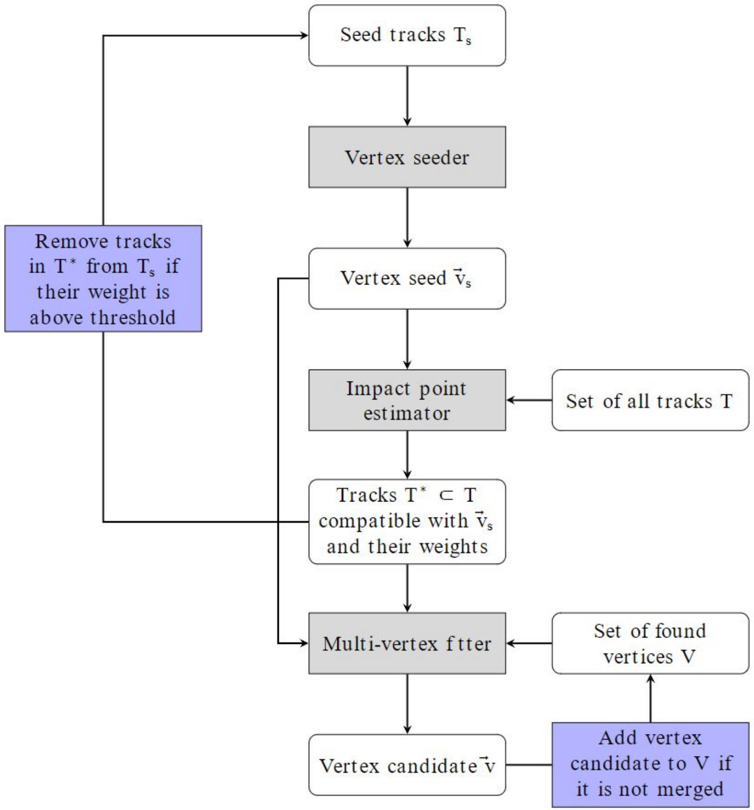


Vertex t0 resolution separately for various cases, where "HS" ("PU") stands for hardscatter (pileup). **Only track clusters tagged by HGTD were evaluated**

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

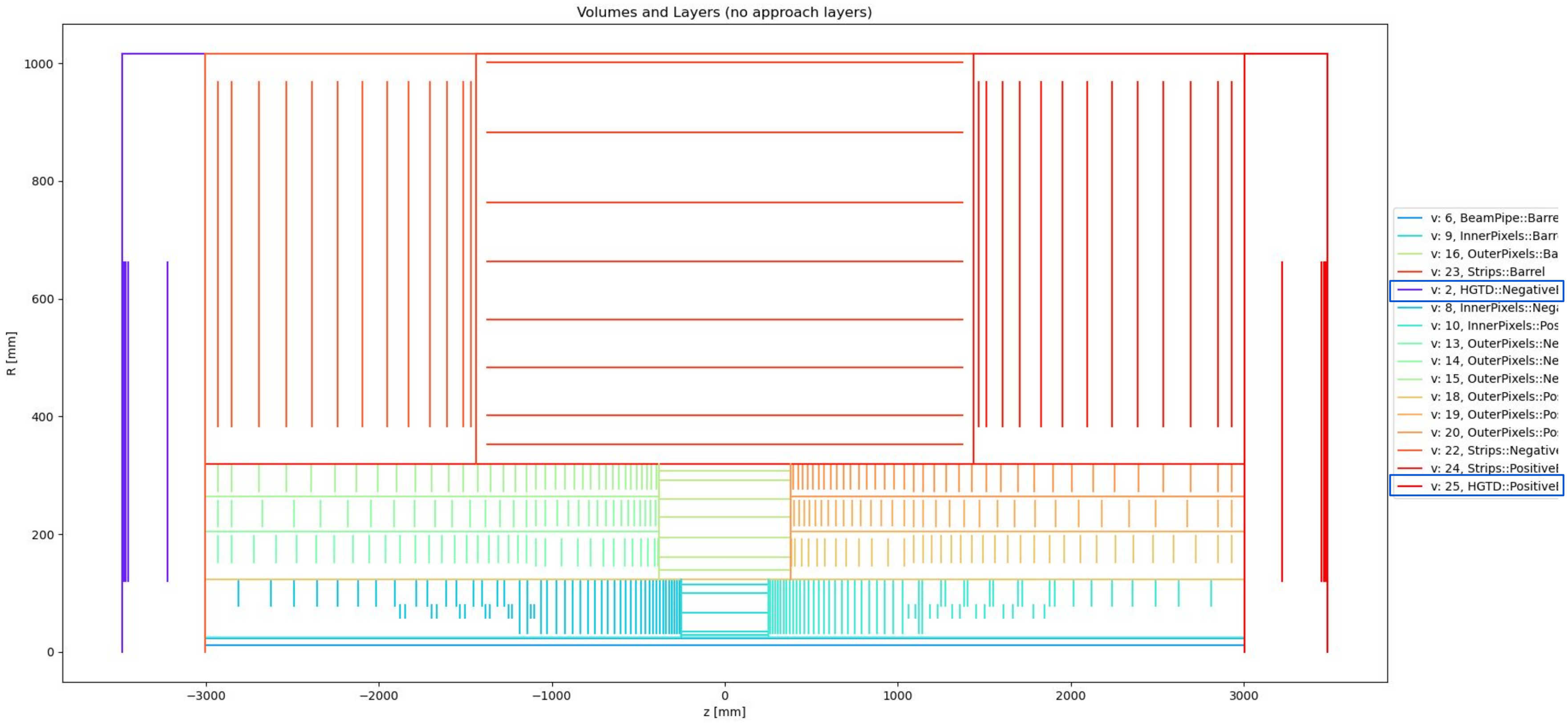
$$\sigma = 35 \text{ ps} \qquad \sigma_t = s \times \sigma$$

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

```json
{
    "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"
                    }
                ]
            }
        },
      ...
  }
```

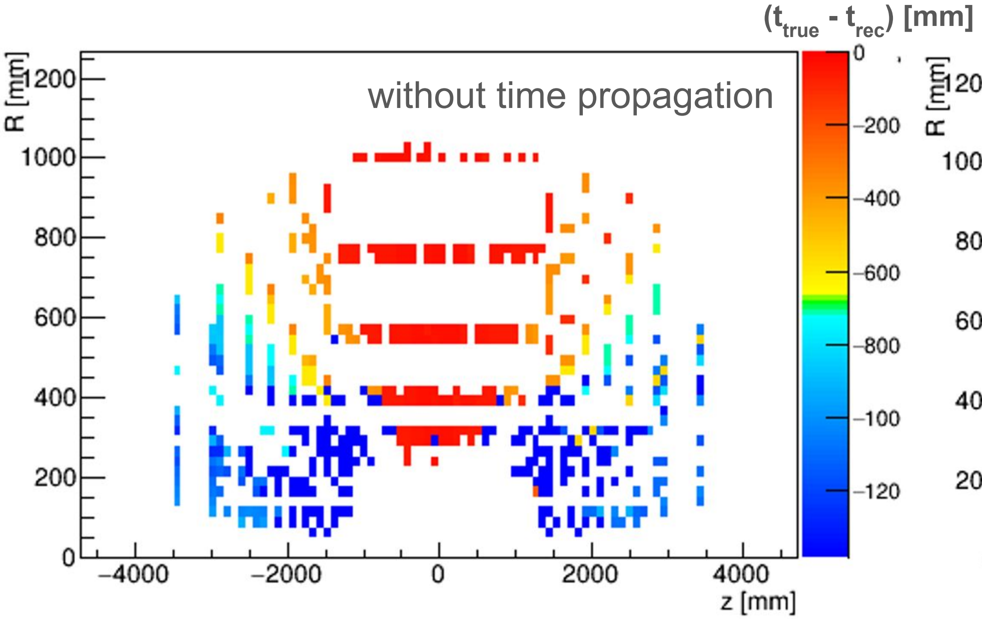
ITK + HGTD geometry at ACTS

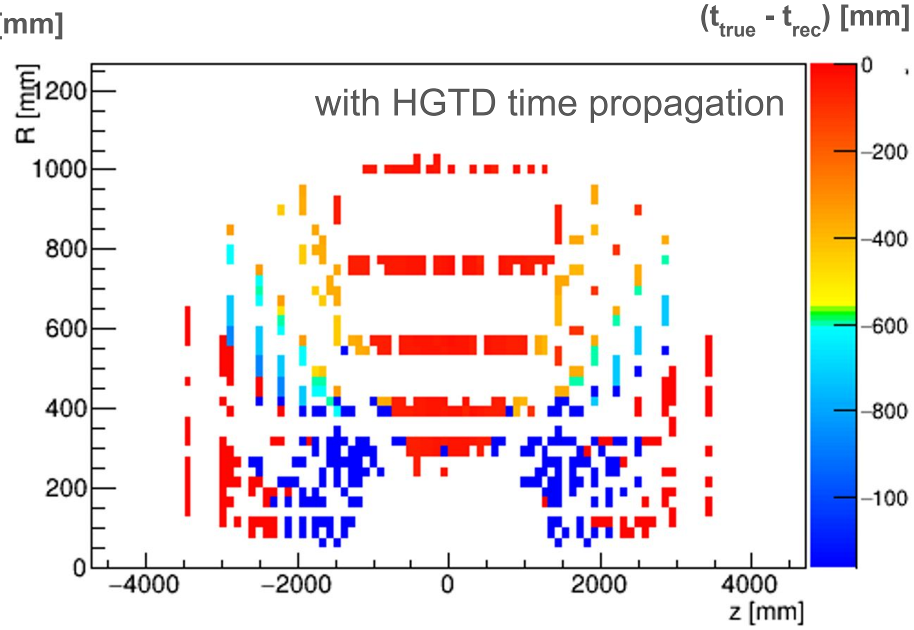
- HGTD endcaps are mapped to volume 2 and 25 of our geometry file



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

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

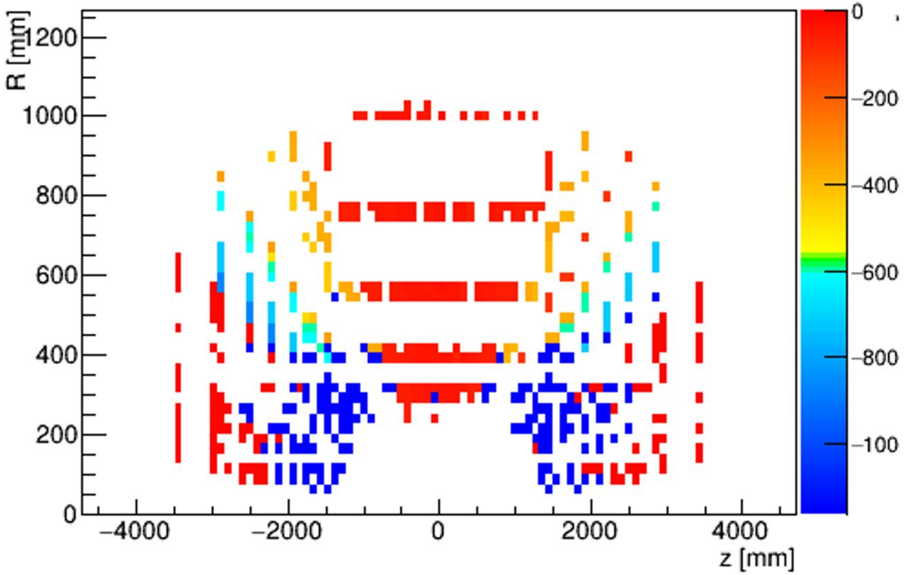
- Works well if p >> m then $t_{n+1} = t_n + h$
- And if the particle is generated at the position and time 0

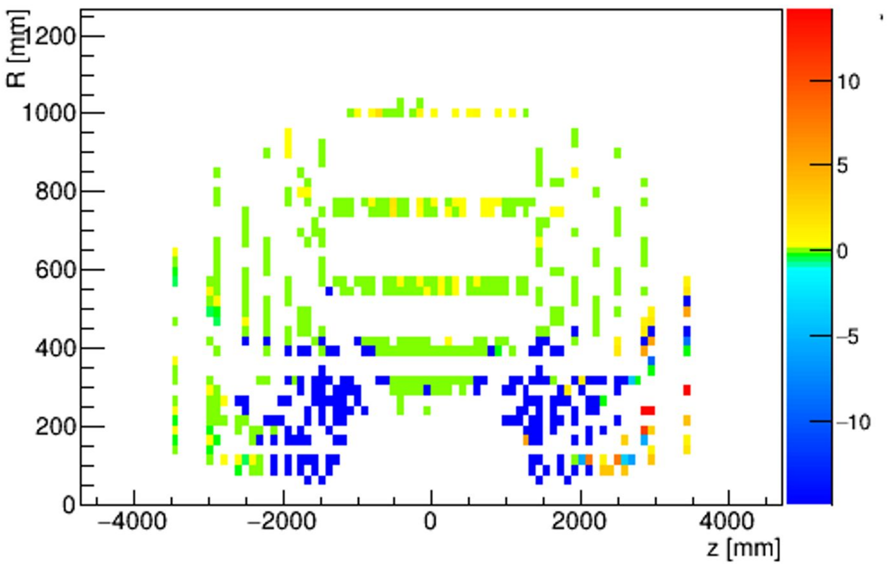
Changing first time estimate (code implementation)

```
// 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

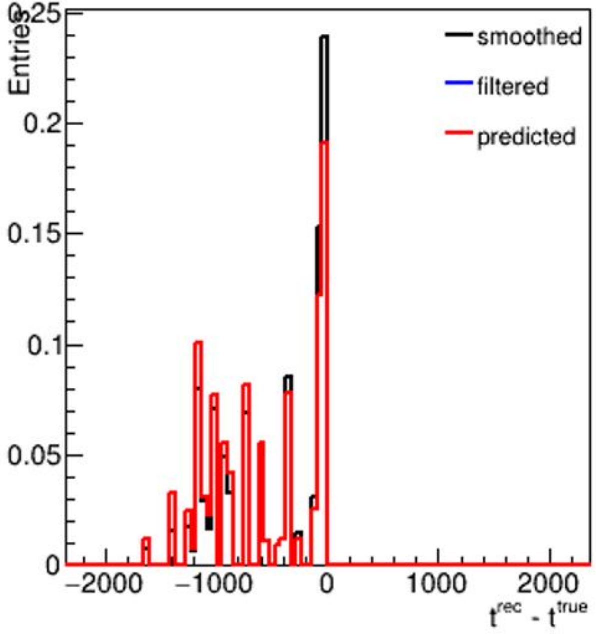
- Aimed to analyse the time residue (t_{true} - t_{rec}) before and after the smearing inclusion
- 100 events with particle gun of a single muon distributed uniformly between eta -4 and 4
- Regional plots below show the **mean time residue after smoothing** for each region

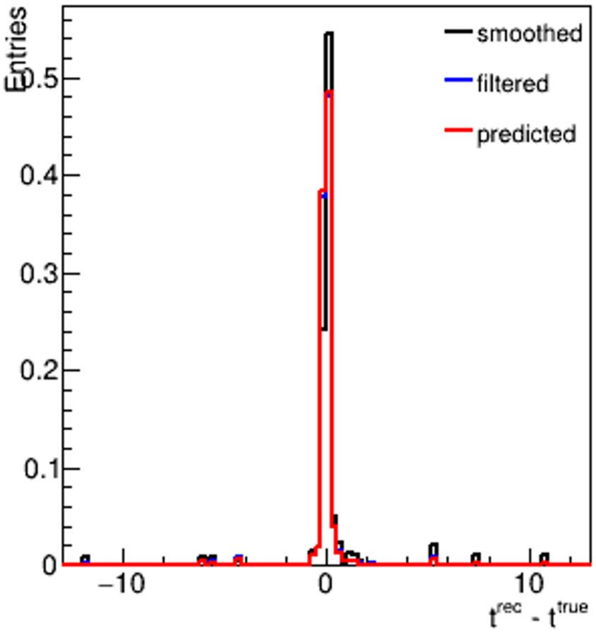




Performance of particle guns reconstruction

- 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,
)
```

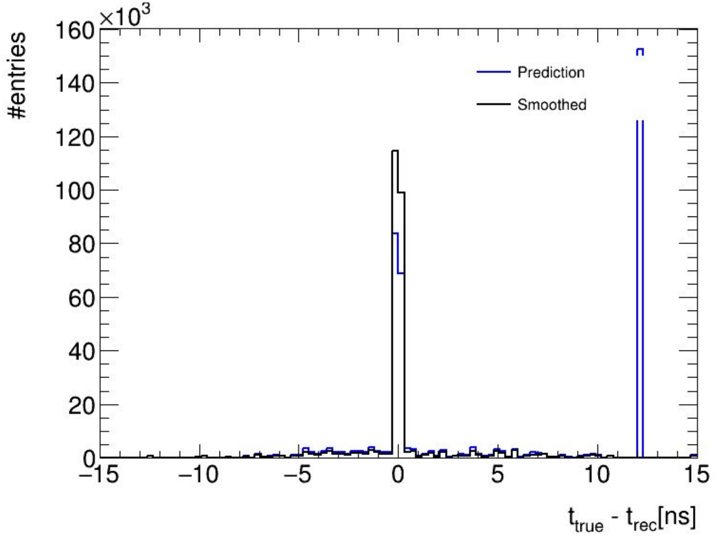
- Now particles aren't only generated at instant 0 (spread of 5 ns), making our first estimate inaccurate
  - HGTD has an important role in fixing the time prediction for tracks generated at t0 != 0

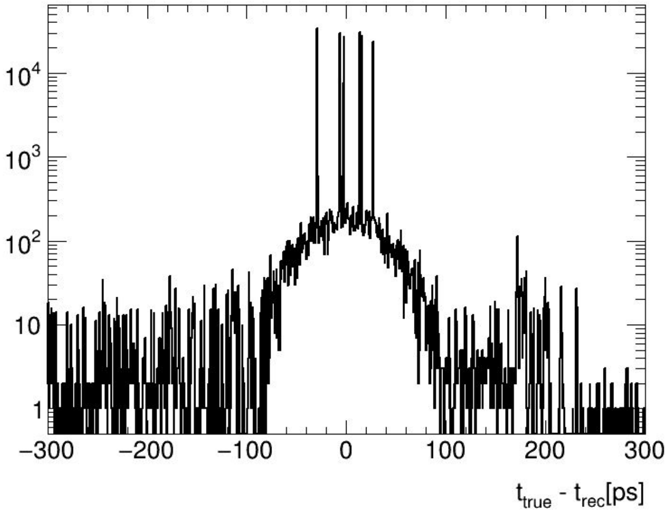
```
addVertexFitting(
    s,
    field,
    vertexFinder=VertexFinder.Iterative,
    outputDirRoot=outputDir,
)
```

Also added Vertex reconstruction

ttbar <u> = 0 residue plots

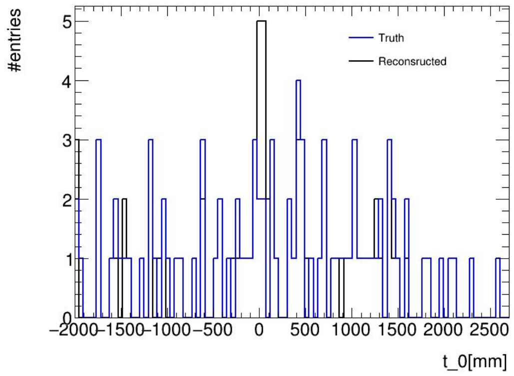
- It can be seen that HGTD filters successfully a cluster of wrong predictions
  - Prediction bin between 10 and 15 ns
- Bulk of the distribution spams from -100 to 100 ps

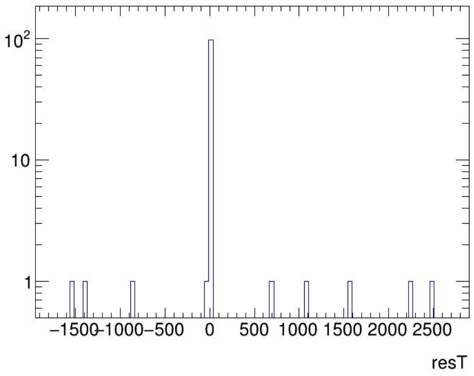


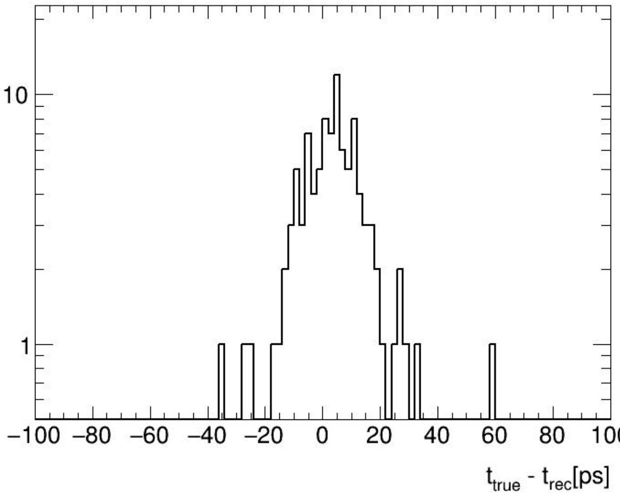


ttbar <u> = 0 vertex reconstruction

- As seen in the introduction, vertex time determination is what is crucial
- The performance of vertex reconstruction shows good acceptance
  - Most (?) vertex were reconstructed
  - Bulk of residue distribution spams from -40 to 60 ps
- Still needs to check on outliers (are they hits outside hgtd?)

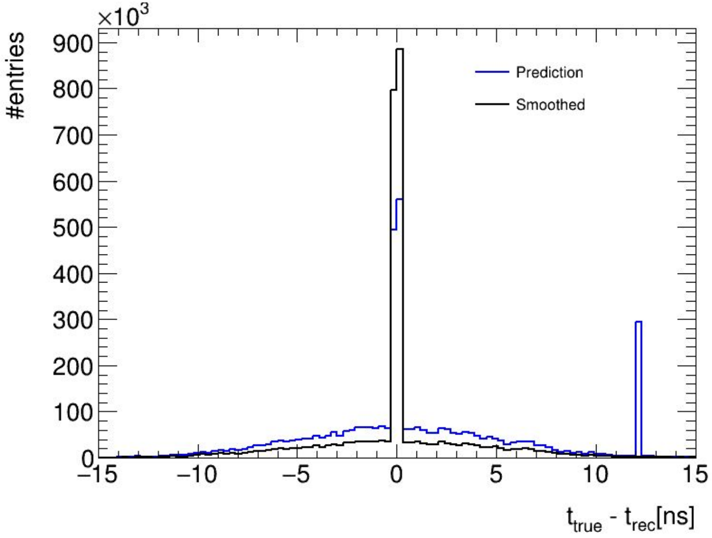


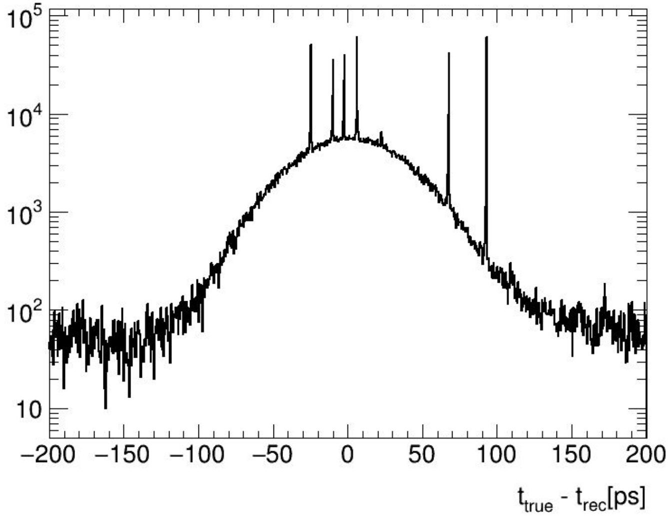




ttbar <u> = 200 residue plots

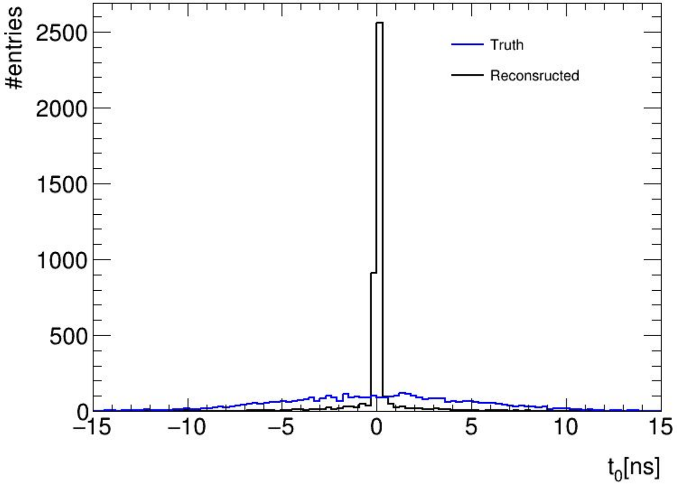
- Higher dispersion in the bulk of the distribution and the increase in outliers
- Still filters well tracks that go to the forward region

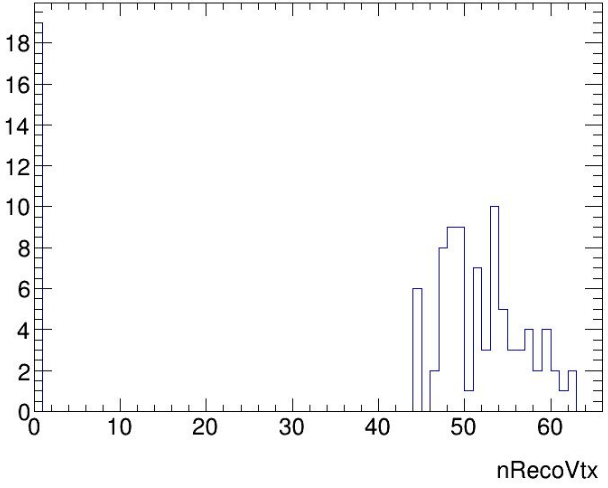




ttbar <u> = 200 vertex reconstruction

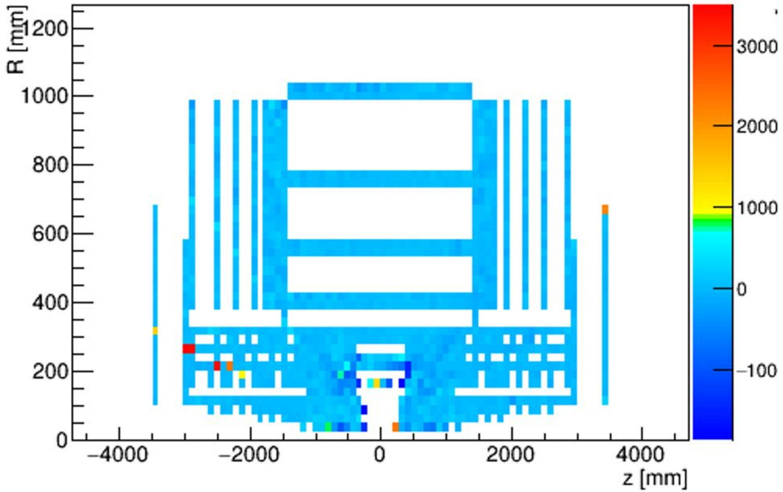
- Reconstruction fails to reconstruct vertexes
  - Error in the scale of ns
  - From 200 vertex only ~60 were reconstructed per event
- Tried to filter only particles with HGTD hits (eta > 2.4 and high pT) but the reconstruction still doesn't work

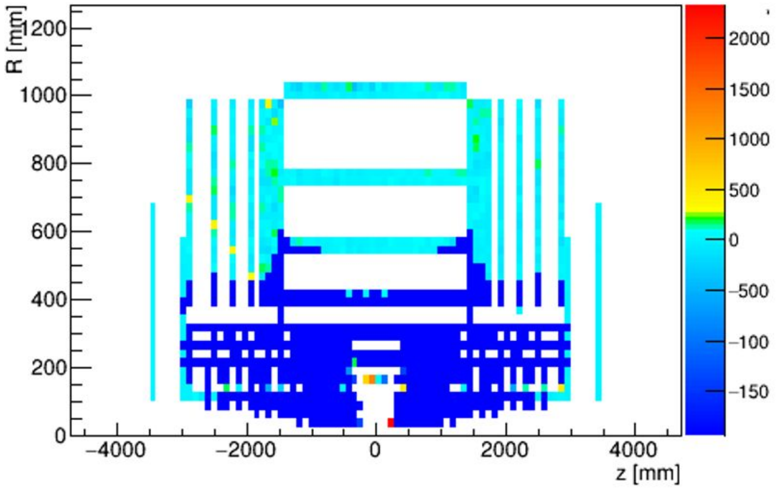




Points of improvement

- Use more accurate event generation time smearing
  - For now vertexes are generated with $t_0 \sim N(0,5ns)$, 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

Next steps

Improve CKF timing integration

- Establish value for first estimate covariance
- Understand outliers at the vertex reconstruction
- Evaluate tracking efficiency performance
- Write an article about it (?)

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