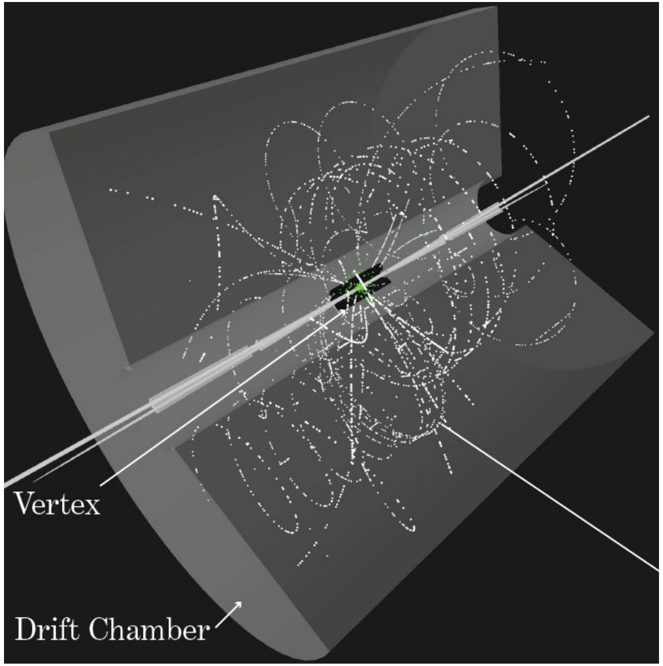
ML-based pattern recognition for CLD/IDEA

Dolores Garcia, Andrea de Vita, Brieuc Francois, Michele Selvaggi

Objective: adaptive reconstruction

- Tracking has two stages: finding and fitting
  - Track finding requires to identify groups of hits that form a track
  - Challenging because:
    - Different geometries
    - Varying number of hits
    - Missing hits in trajectory, one or multiple sub detectors
    - Abrupt changes in direction
- Classic pattern recognition methods use combinatorial optimization such as Kalman Filters
- Detector dependent and long development cycles
- The IDEA detector is particular due to the left right ambiguity in the drift → classic algorithms are not directly applicable

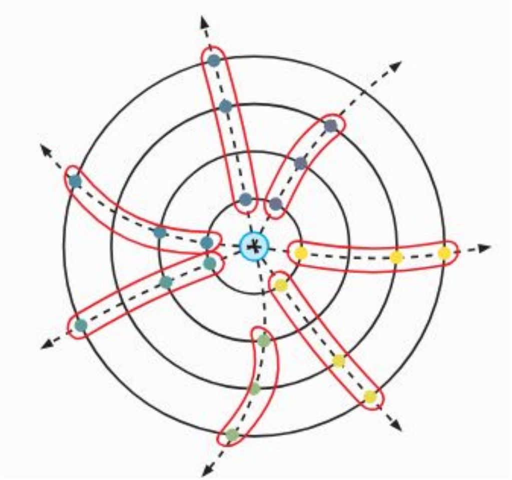


Goal

- Track finding algorithm that can cope with multiple sub-detectors and input geometries
- Is not dependent on the geometry definition and material specification
- Does not rely on analytical parametrization of the trajectories

Classical tracking approaches

- Seeding and track following:
  - **ACTS:** seeding finds triplets of points likely to belong to the same track then uses a Combinatorial Kalman Filter and takes into account material (geometry)
  - **Conformal Tracking** + Cellular automaton (**CA**) (CLD baseline) [2], coordinate transformation (circles transformed to straight lines)
    - Deviations from the circular path e.g displaced tracks are taken into account by CA
    - Creating seed cells and extrapolating along the cell direction
- Drawbacks of these methods:
  - Computationally demanding (CKF)
  - Geometry dependent
  - Do not take into account different input (hit) geometries

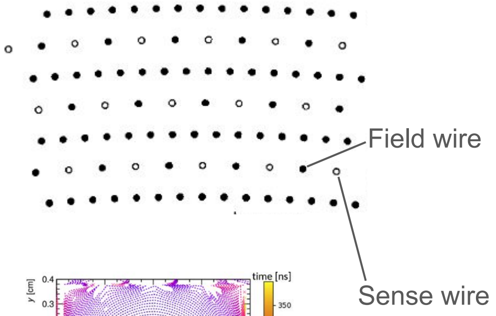


ACTS seeing approach [1]

Dataset

- Generated events of Z→qqbar 91GeV without background using Pythia + ddsim with CLD_02_v05 (key4hep 2024-05-09) + digitizer
- Store hits from
    - CLD: Vertex Barrel, Vertex Endcap, Inner Tracker Barrel, Inner Tracker Endcap, Outer Tracker Barrel, Outer Tracker Endcap
    - IDEA: Digitizer-Distance along the wire and distance to the wire → left right hit coordinates

For validation store MC association using “TrackerHitRelations”



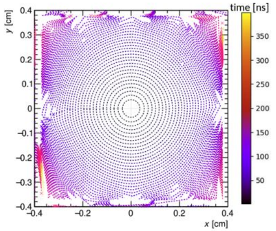
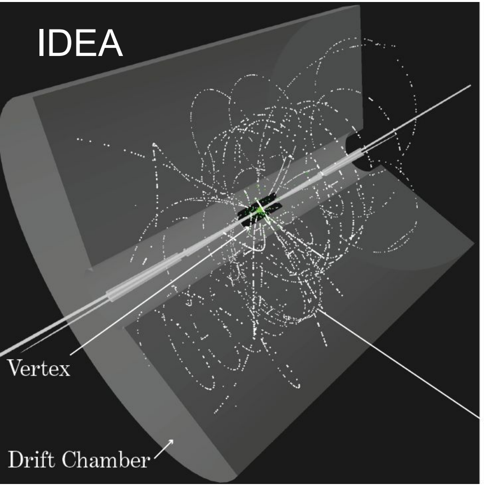
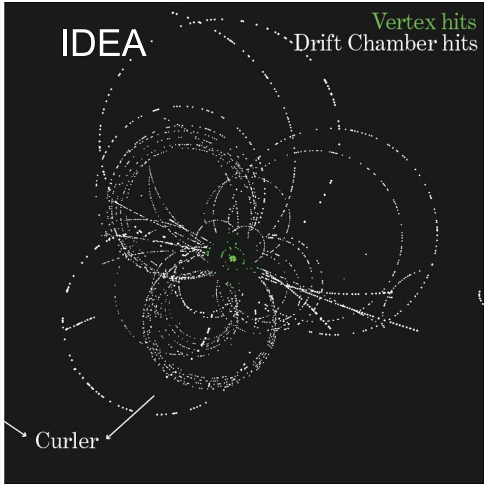
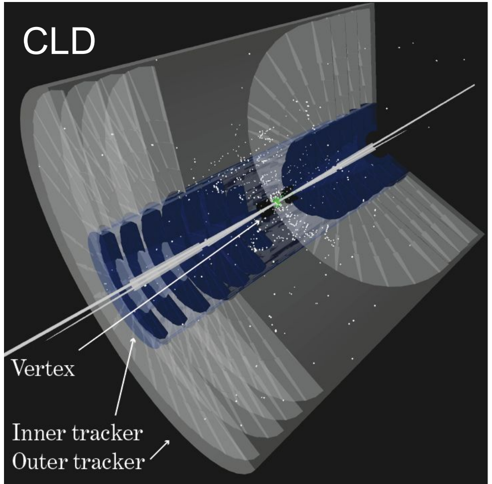


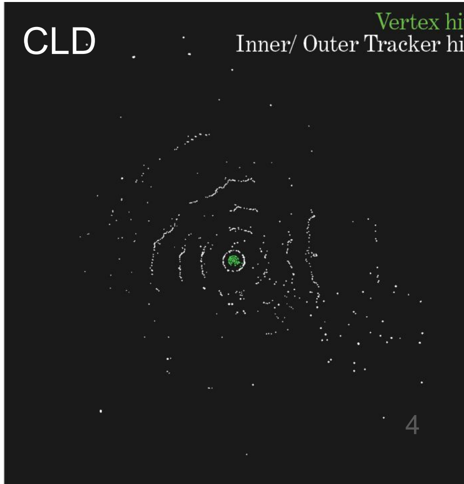
Figure 15 An example of the lines connecting points with the same drift time towards the sense wire (“isochrones”).

[1]

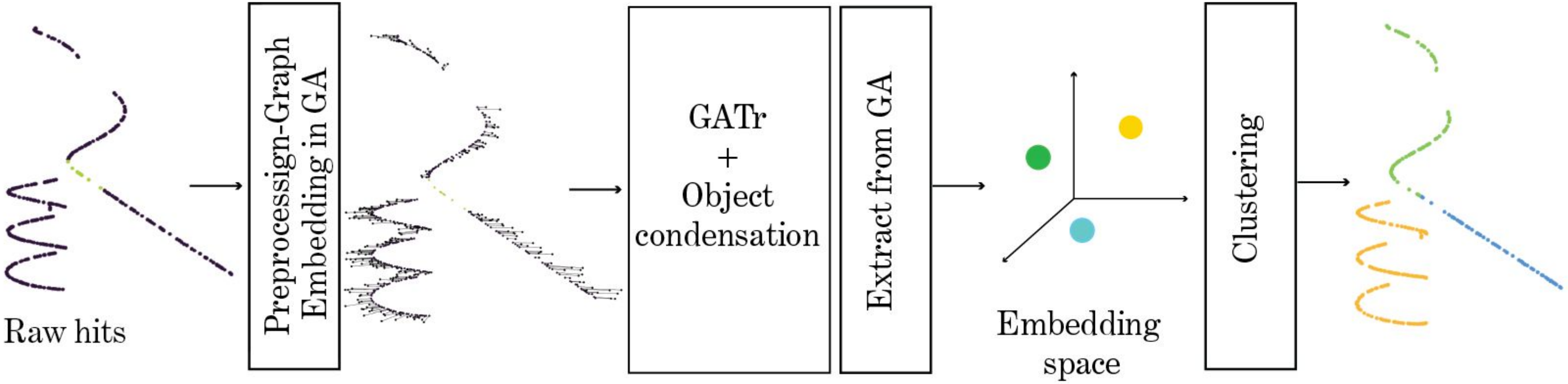








Algorithm



The algorithm is independent of the detector geometry (same pipeline for IDEA)

- Embedding of raw hits
- Graph neural network
- Clustering step → outputs are Track candidates (collection of hits)

Performance for complex events CLD: tracking efficiency

Definitions from [CLD paper]

Track hit purity: is the ratio of the number of hits in the track that belong to the MC particle and the total number of hits of the reconstructed track

Track hit efficiency: is the ratio of the number of hits in the track that belong to the MC particle and the total number of hits this particle left in the detector

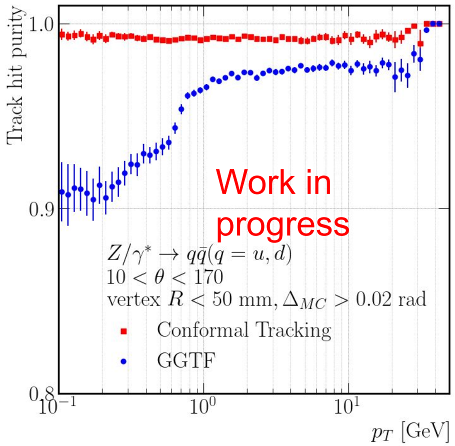
Reconstructable particle: stable at generator level, pT>100 MeV, |cosθ|<0.99 and at least 4 unique hits

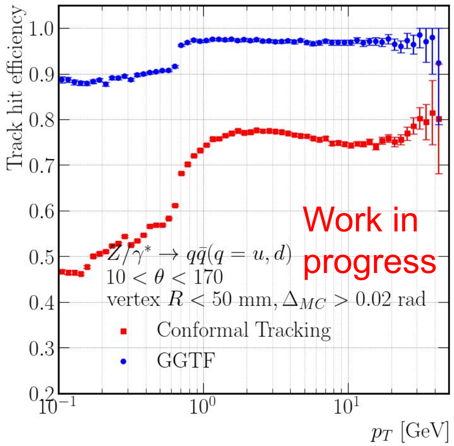
Compare with SiTracks_Refitted

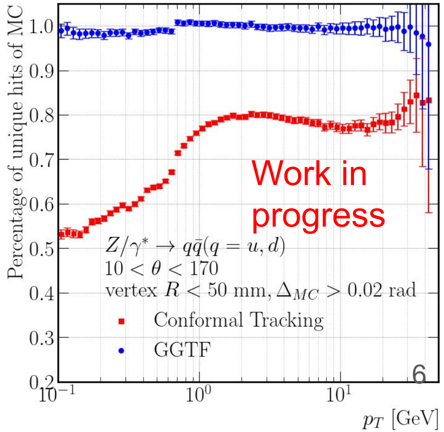
Fakes: no MC is assigned to the reconstructed track

The fakes can not be evaluated per pT bin since the track is not reconstructed but the total number of fakes is:

- ML: 4.2%
- Conformal: 4.4%





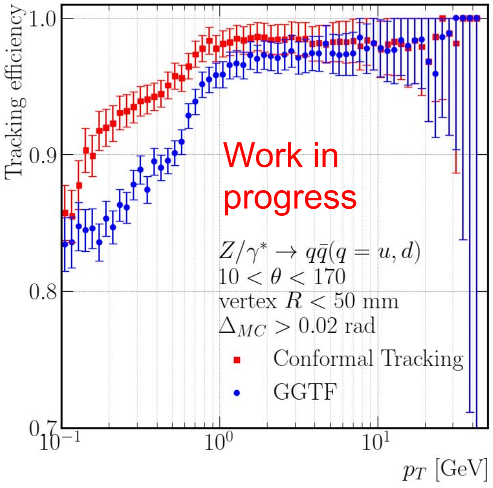


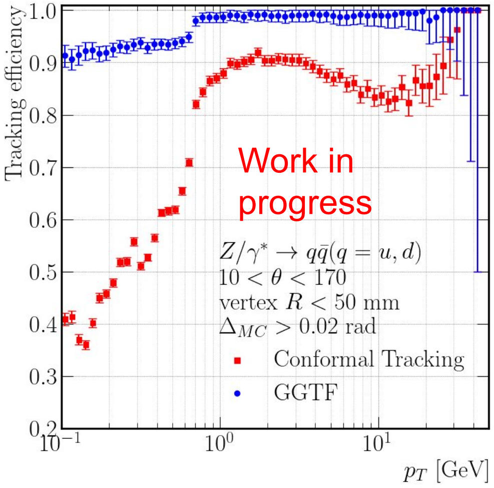
Performance for complex events CLD: tracking efficiency

Definitions from [CLD paper]

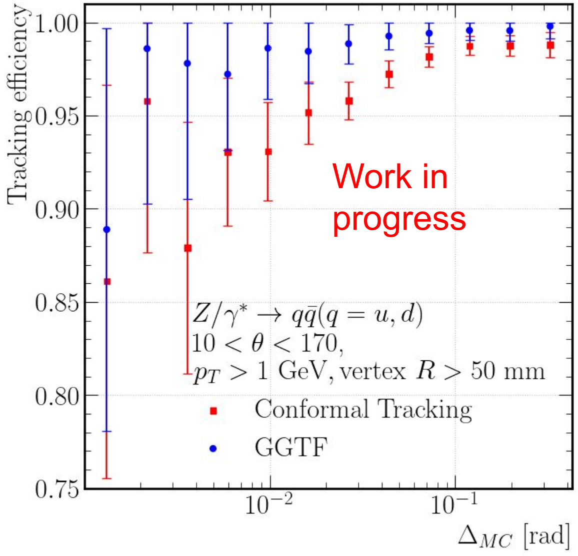
Efficiency def 1. Percentage of reconstructable particles with track hit purity >75% (track segments)

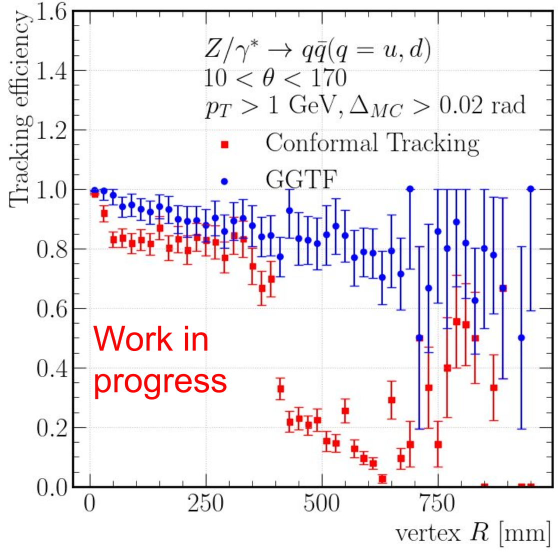
Efficiency def 2. Percentage of reconstructable particles with track hit purity >50% and track hit efficiency > 50%





Performance for complex events CLD: tracking efficiency



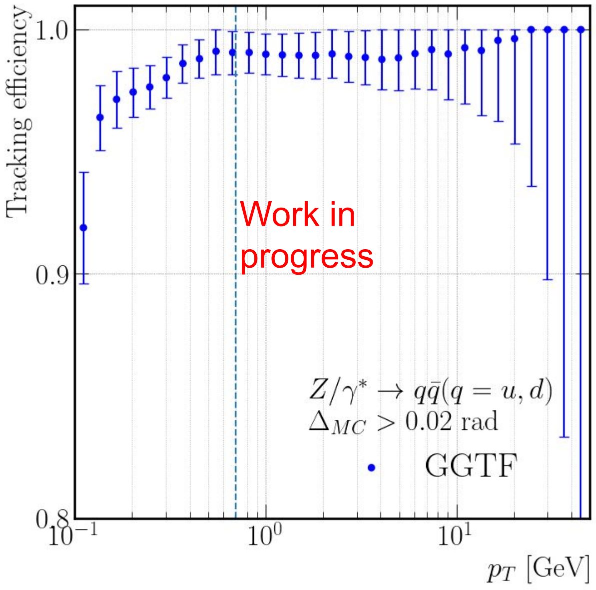


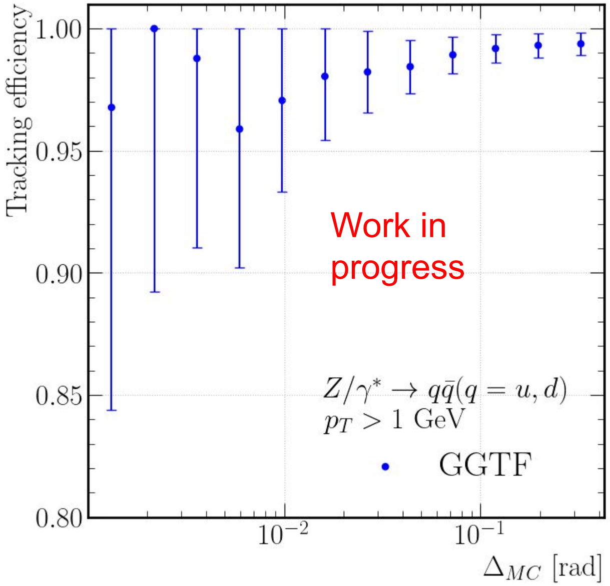
Efficiency as a function of particle proximity:

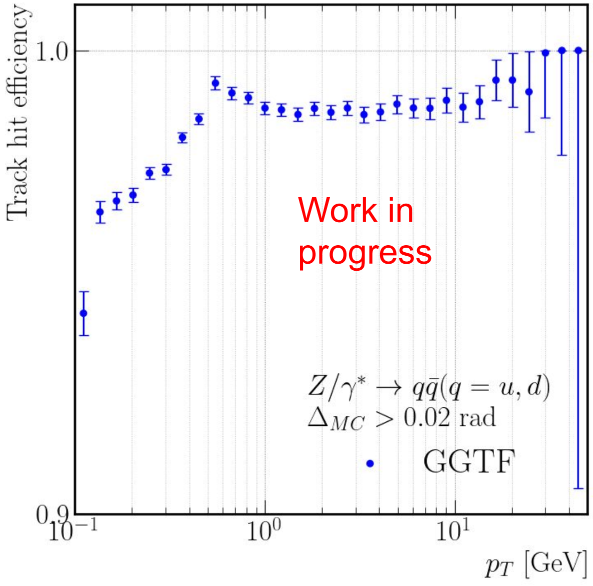
ΔMC = $\sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$

Efficiency as a function of production vertex radius

Performance for complex events IDEA: tracking efficiency







Tracking efficiency def 2)

Tracking efficiency vs ΔMC

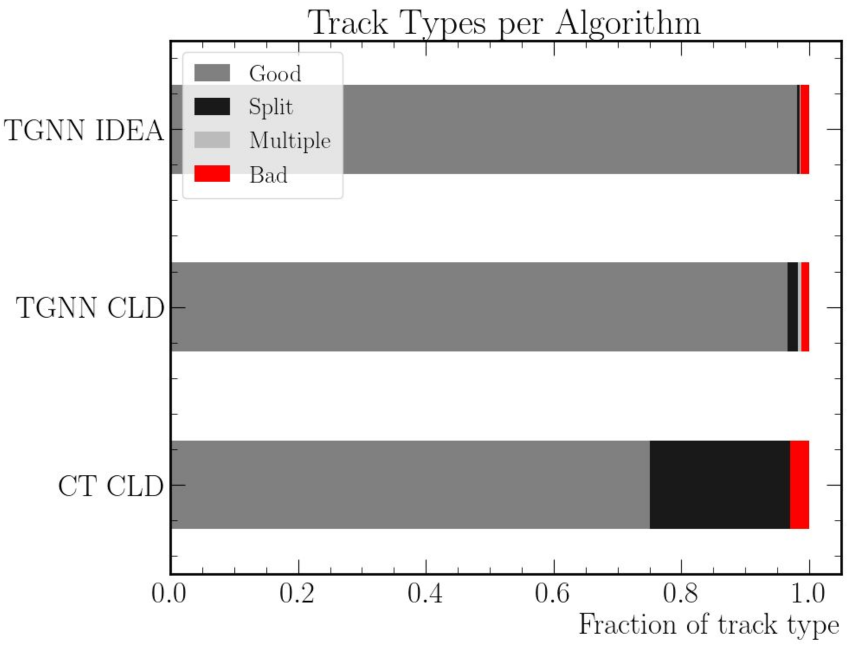
Track hit efficiency

Performance for complex events IDEA vs CLD

Track hit purity (THP) Track hit efficiency (THE)

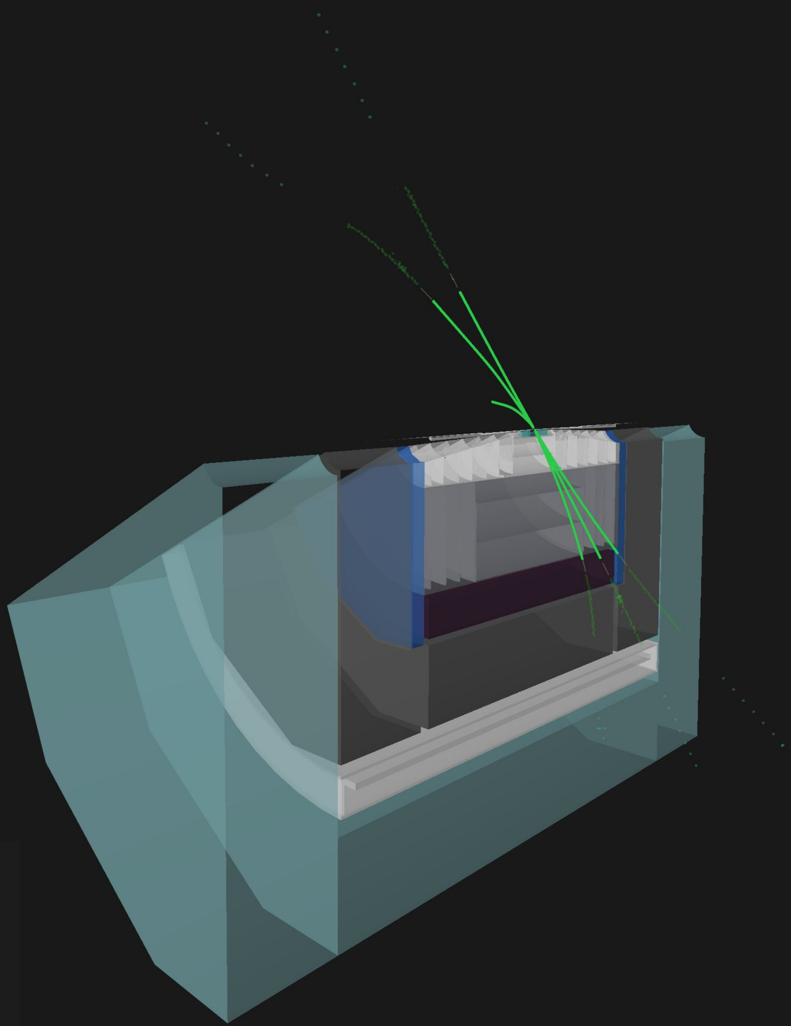
- Good: THP>50 % THE >50 %
- Split : THP>50 % THE <50 % (only a fraction of the track is reconstructed)
- Multiple: : THP<50 % THE >50 %
- Bad: THP<50 % THE <50 %

Overall, more splitted tracks are recovered using the TGNN method



Z→ ττ→(3μ)(3μ)

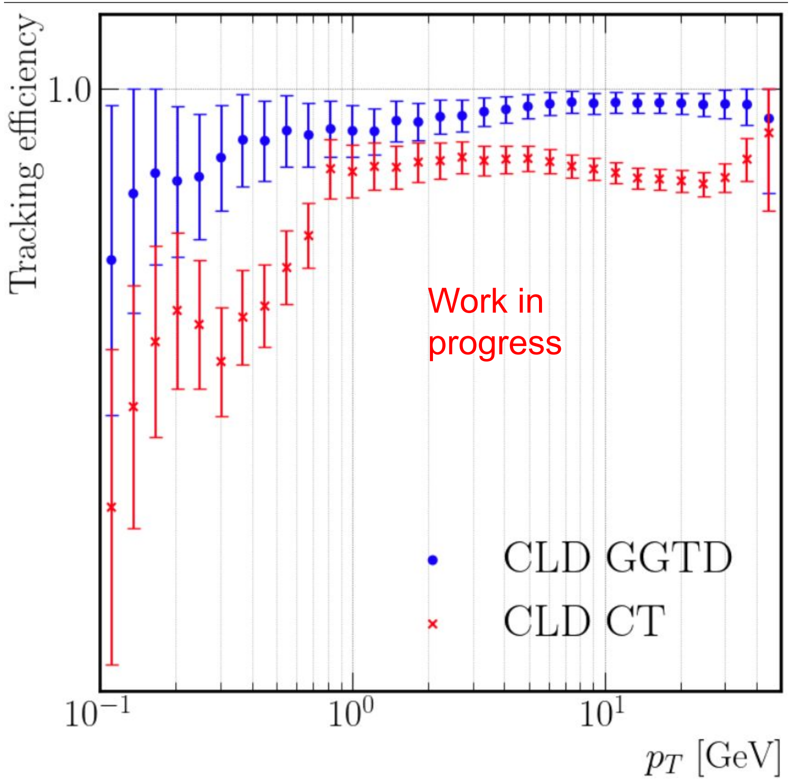
- Force pythia decay
- Same data for CLD (02_v06)
- Performance comparison



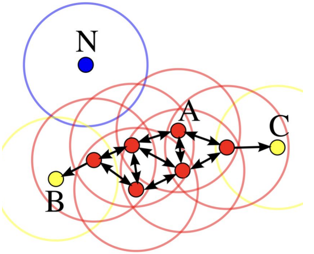
```
WeakSingleBoson:ffbar2gmZ = on
23:onMode = off
23:onIfAny = 15
15:onMode = off
15:AddChannel =  on 0.00001 0 13 13 -13 ! forced tau -> 3mu decay, pure phase space
```

Efficiency for Z→ ττ→(3μ)(3μ)

- Tracking efficiency defined as hit purity>75%
- Improved efficiency for CLD CT 'out of the box'
- However, optimization of our model done without background (next steps)



Key4hep algorithm implementation



- **Pattern recognition** is implemented as a Gaudi::functional (key4hep, v0) [Repo]
  - Returns a track collection given a collection of hits from all subdetectors
  - Model is trained in pytorch an exported as ONNX
    - Inference session in the **initialize()**
    - Execution phase **operator()**
  - DBSCAN algorithm in C++ performs the clustering
- Implementation of an **evaluation step** that returns a quick estimate of tracking efficiency and a table of parameters to calculate tracking efficiency as a function of particle properties (such as , etc…)
  - Soon merged in key4hep, more details Andreas' talk

Full pipeline in key4hep (simulation to evaluation):

1. Idea detector simulation (IDEA_o1_v02.xml)
2. Digitizer v01 (moving to Digitizer v02)
3. Generalised geometric track finding algorithm
4. Evaluation step (tracking efficiency)

Summary and next steps

- Performance is improved in terms of efficiency compared to the Conformal tracking ‘out of the box’
    - The purity is lower as the tracks include more hits but remains high
    - **Next steps:** we need to evaluate the algorithm with background overlaid (as this could explain the difference with CLD)
    - **Next steps:** We will update the model with the new geometry (v3) and new digitizer will be updated to take into account ‘circular’ ambiguity
- Key4hep implementation is ready for IDEA. **Next steps:** a similar pipeline is available in key4hep for IDEA so it could be adaptable for CLD
- **Next steps:** the effect on the track fit still needs to be evaluated
- **Next steps:** evaluate and improve inference time (important for evaluation at Z pole)