Alma Mater Studiorum - Università di Bologna

Vertex Track Perfomance Studies

Giacomo Ubaldi

Definitions for performance factors

- **Reference set: $N_{reference}$ (truth side)**
  all the tracks that an algorithm performing ideally should find and reconstruct:
  - all the tracks associated to a MC particle that crosses the FOOT apparatus at least until the last plane of the vertex (using MCRegion)
    - beam
    - primary fragment generated in the target

- **Good reconstructed set: $N_{GoodReco}$**
  all the tracks that are reconstructed by the tracking algorithm which are associated to MCparticles in the reference set .

- **Bad reconstructed set: $N_{BadReco}$**
  all the tracks that are reconstructed by the tracking algorithm but associated to MC particle that do not belong to the reference set.

Track reconstruction

For the track reconstruction I considered:
case 1: a track is reconstructed with 4 clusters (one for each VT plane)
case 2: a track is reconstructed with at least 3 clusters
case 3: a track is reconstructed with at least 3 clusters + random noise pixels are generated

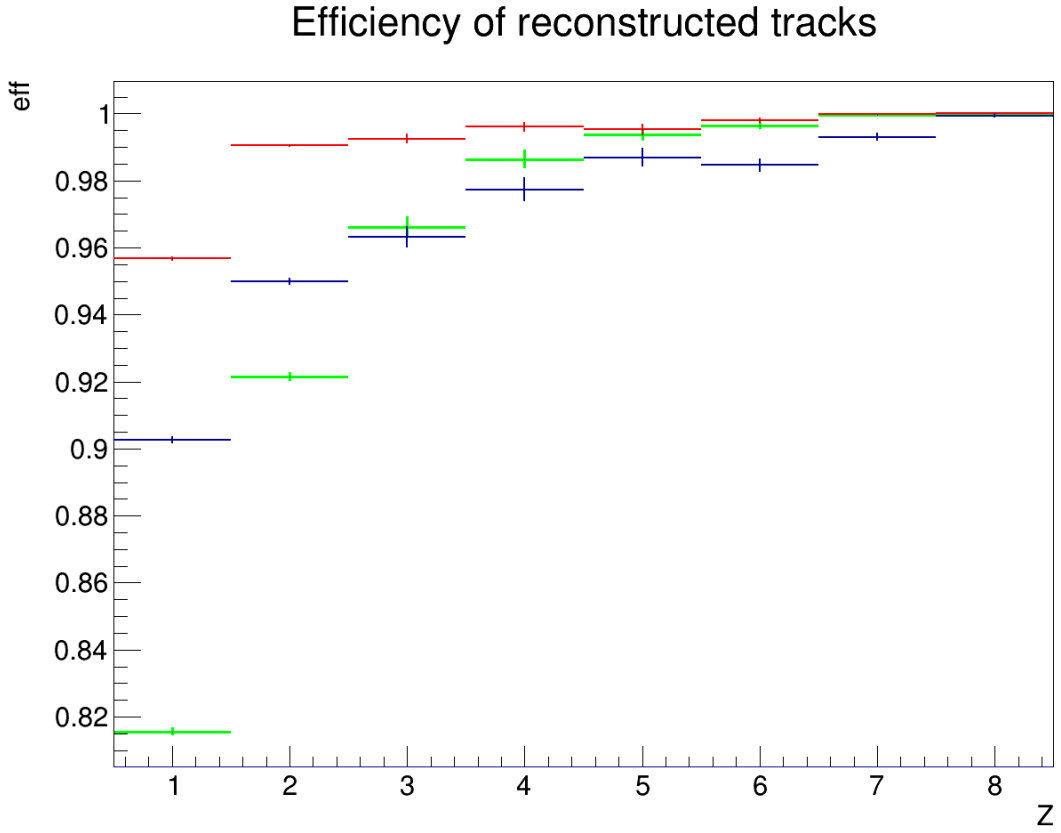
NB:
To associate a MC Particle to a reconstructed track:

- I consider the MC ID of all the clusters belonging to the track
- I take the most frequent one: this is the ID of the MC Particle matched

Definitions for performance factors

- **Reconstruction efficiency:**

$$\epsilon_{track} = \frac{N_{GoodReco}}{N_{reference}}$$

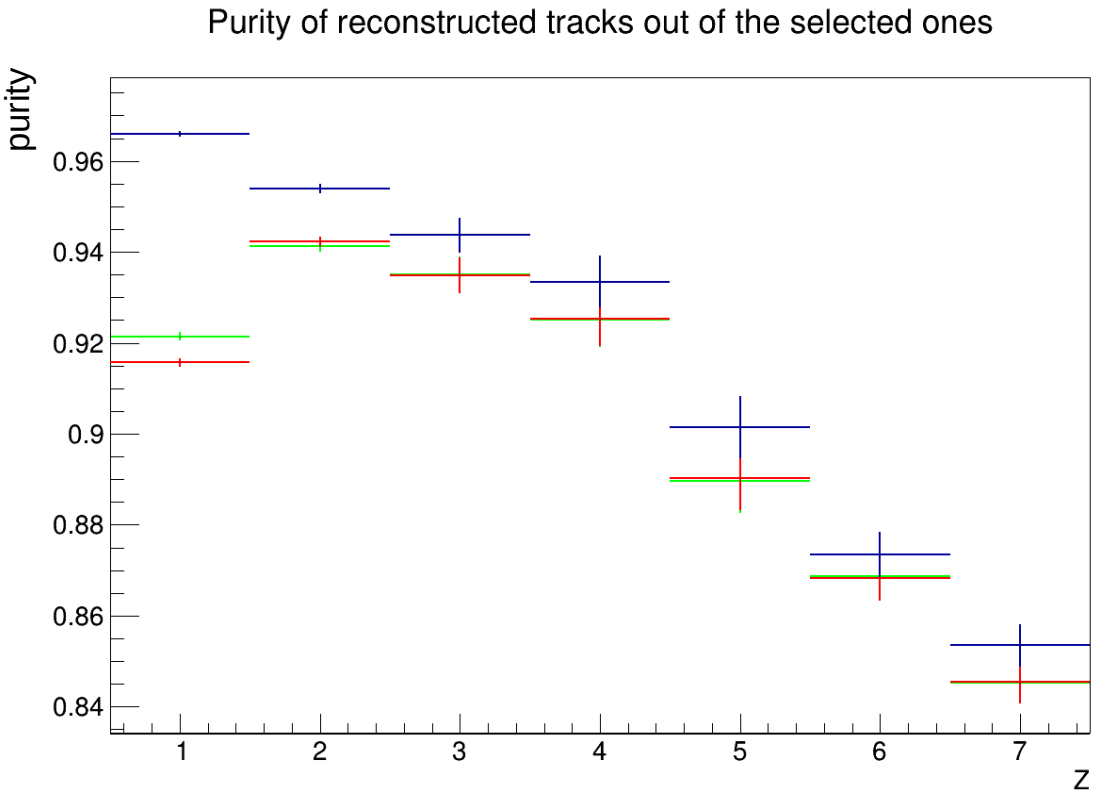


tracks with 4 clusters (case 1)
tracks with >=3 clusters (case 2)
tracks with >= 3 clus + noise (case 3)

Definitions for performance factors

- **Purity:**

$$p = \frac{N_{GoodReco}}{N_{GoodReco} + N_{BadReco}}$$

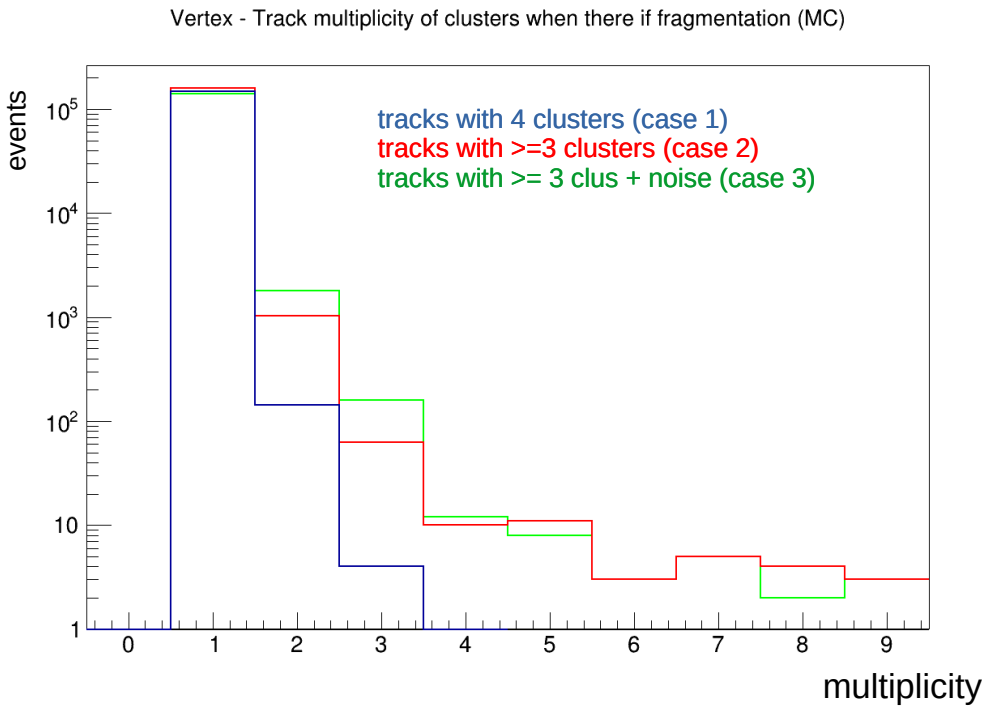


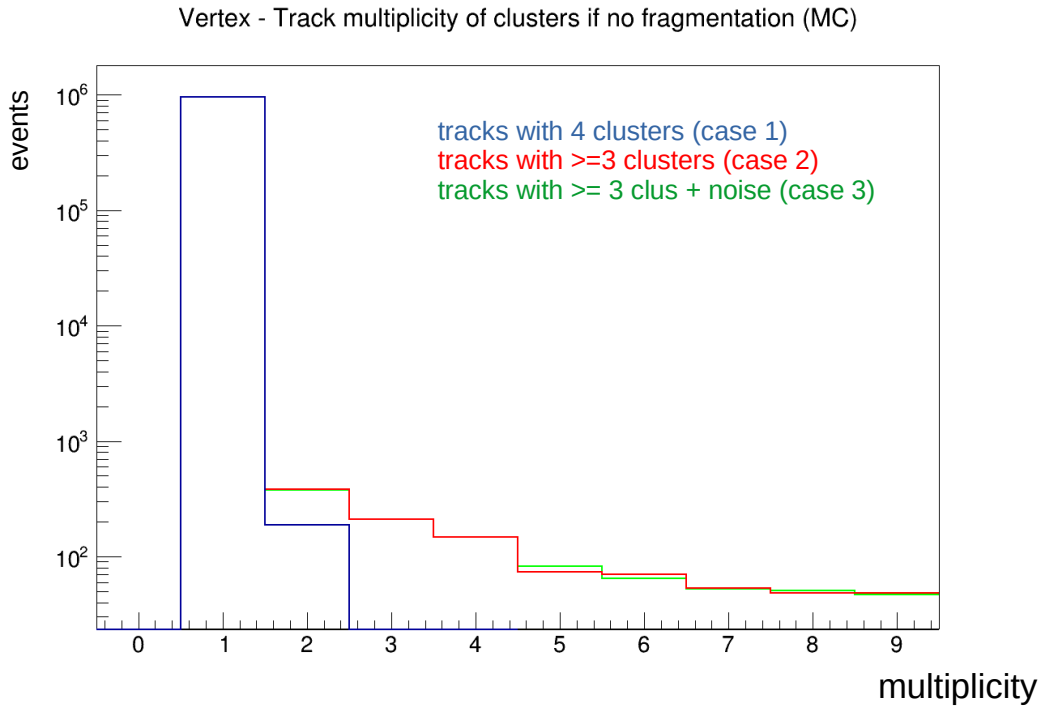
tracks with 4 clusters (case 1)
tracks with >=3 clusters (case 2)
tracks with >= 3 clus + noise (case 3)

Performance studies

- **Multiplicity**: n° of different clusters MC_ID associated to a given track
  - es: m=1 all clusters are of the same MC particle
  - es: m=2 clusters belong to two different MC particle (1-3,2-2)

NB: multiplicity can be higher than 4 (despite the clusters are at max 4) because every one can be associated to different MC particles (with different MC_ID)



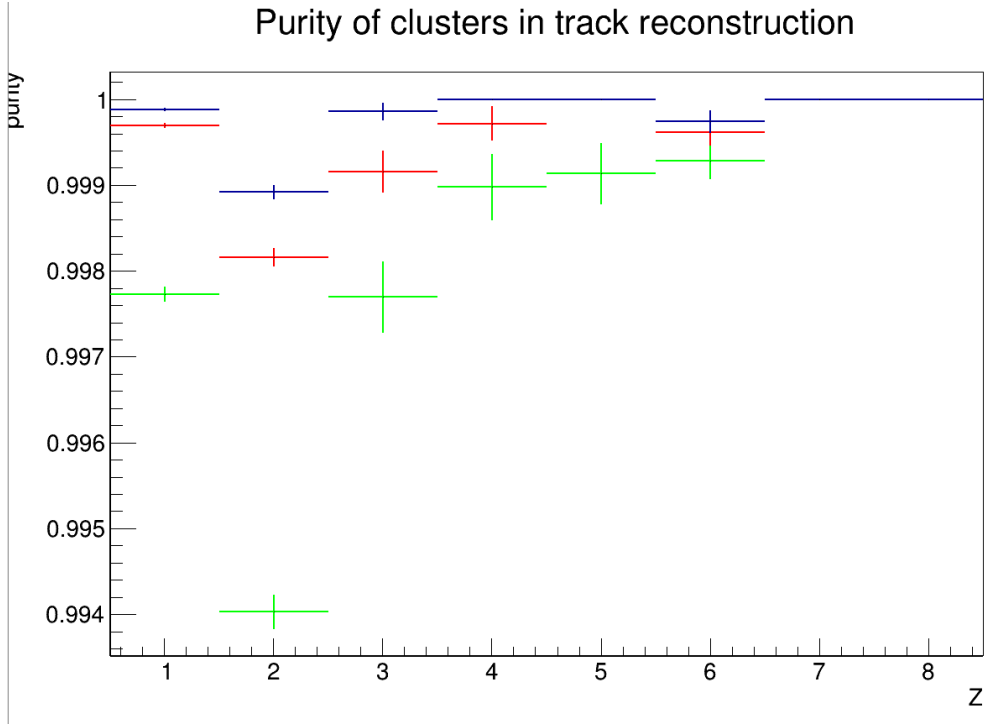


Definitions for performance factors

- **Cluster purity**

  counts of all the clusters matched well with the track MC_ID (in reference set) among all clusters of all $N_{GoodReco}$ tracks

$$\rho = \frac{\sum_{m=0}^{M} N_{correct}^{(m)}}{\sum_{m=0}^{M} N_{total}^{(m)}}$$

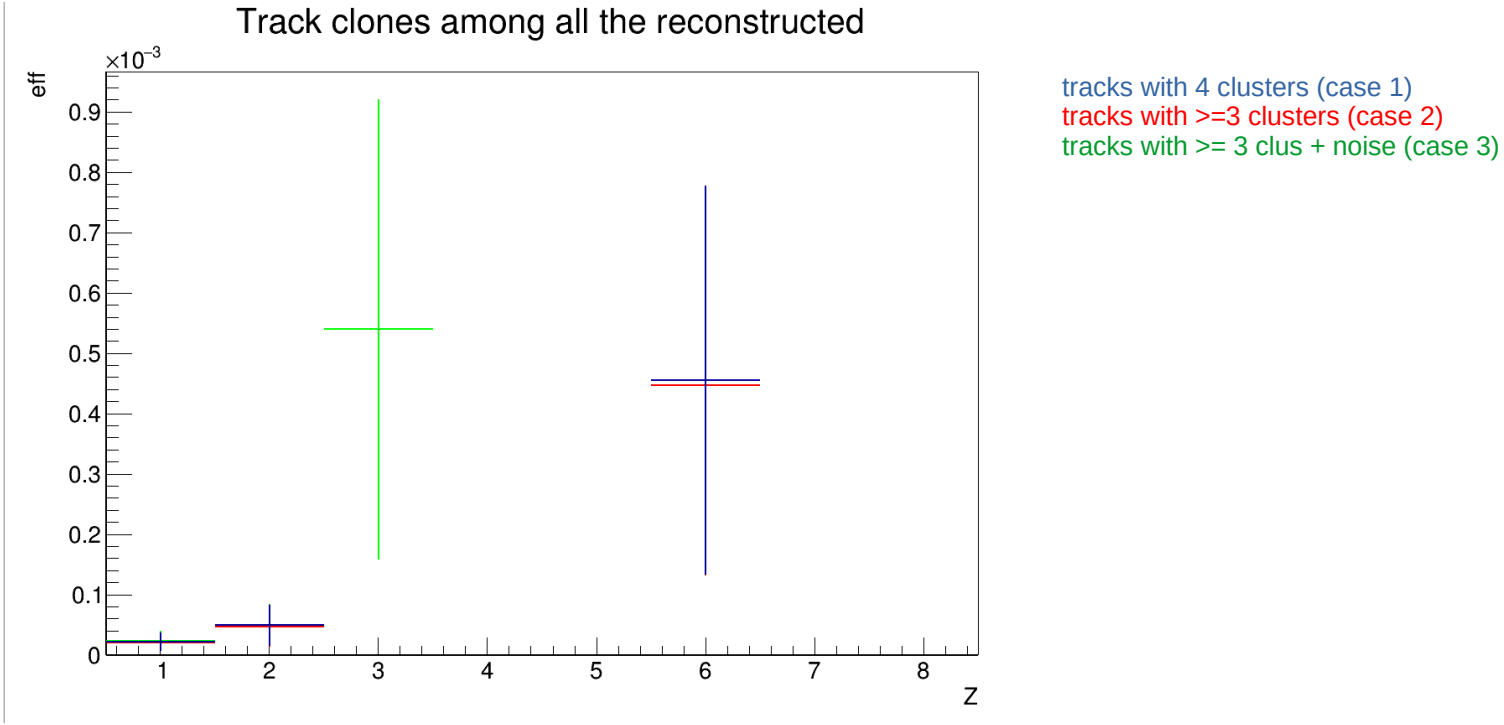


tracks with 4 clusters (case 1)
tracks with >=3 clusters (case 2)
tracks with >= 3 clus + noise (case 3)

Definitions for performance factors

- **Clone multiplicity**

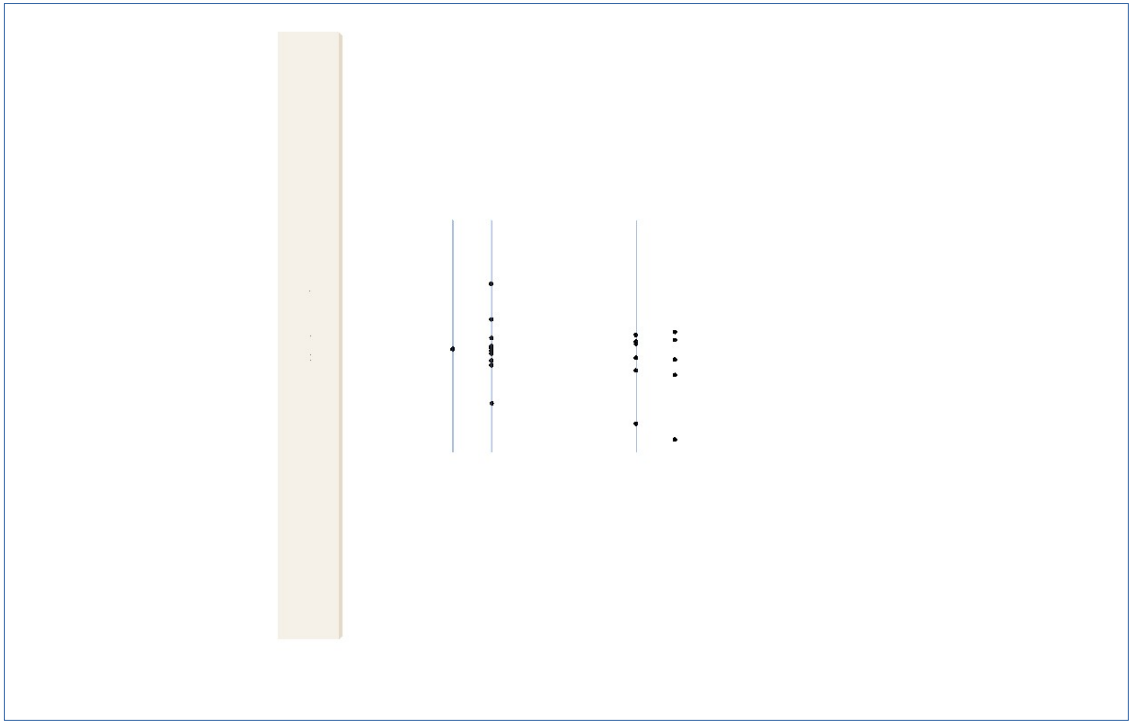
  quantification of the number of multiple cloned trajectories produced for the same MC particle matched to the track



Examples of bad reconstructed tracks

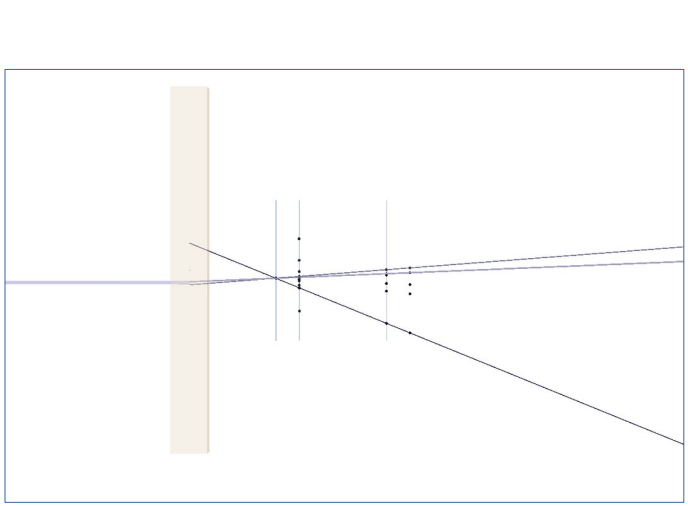
- **Bad reconstructed set: $N_{BadReco}$**
  all the tracks that are reconstructed by the tracking algorithm but associated to MC particle that do not belong to the reference set.

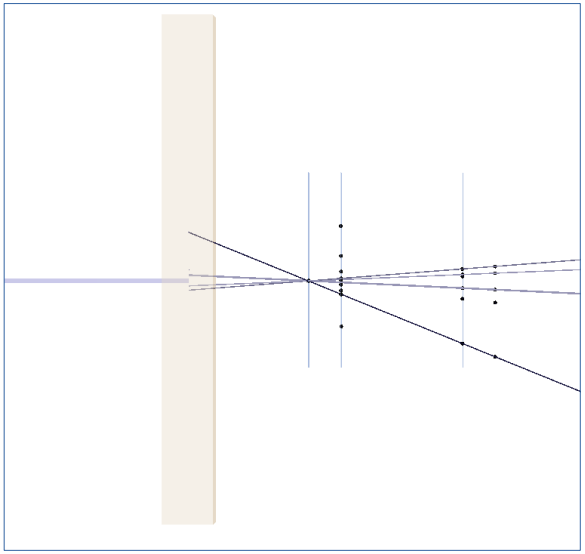
Fragmentation in the first plane of the vtx

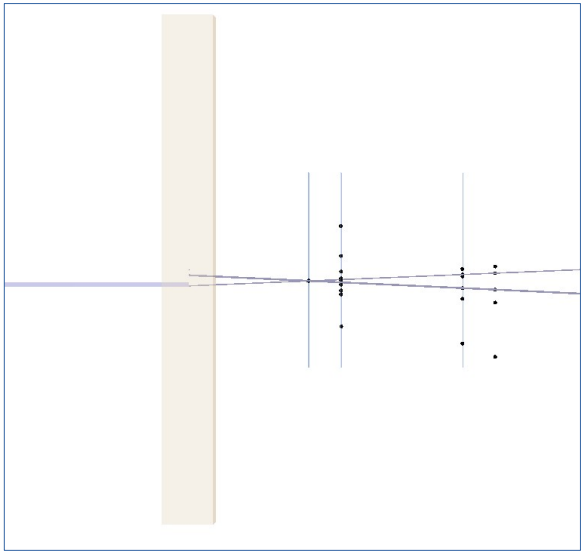


NB: the number of reconstructed tracks changes every time you analyze the same event! (of course it is very uncommon event)

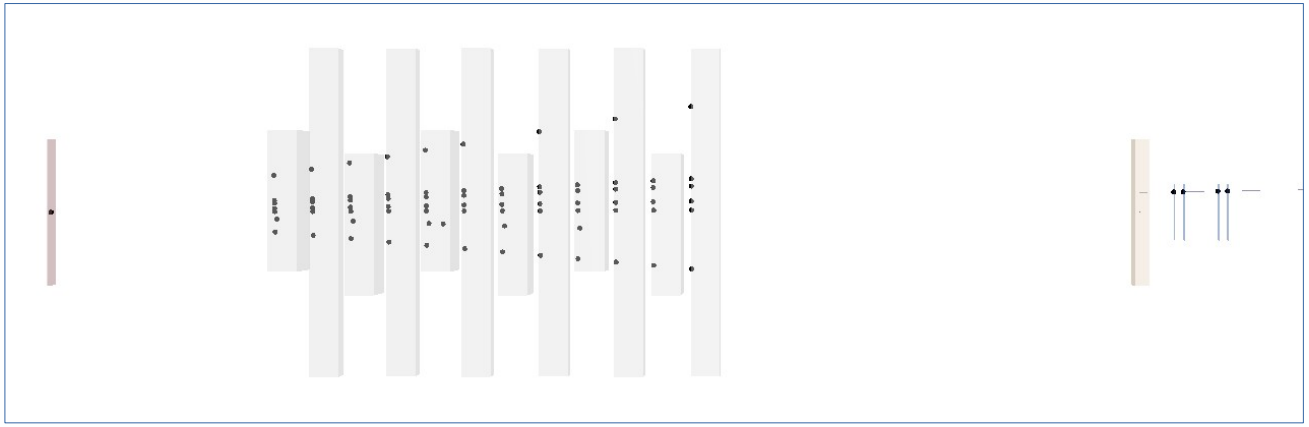
- It could depend on how random noise pixels are accounted by the track algorithm



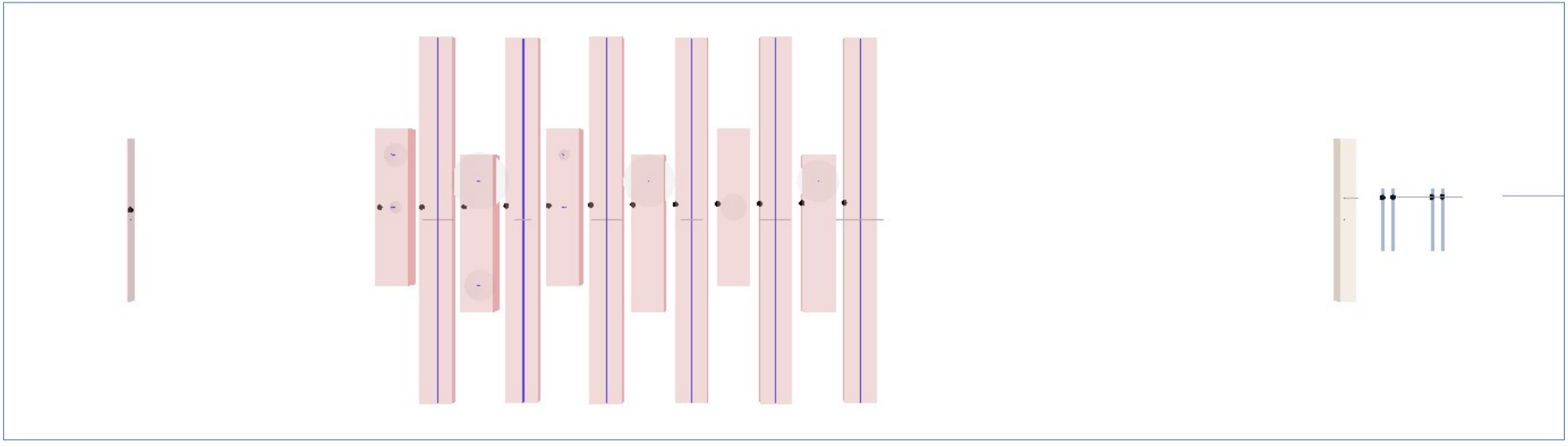




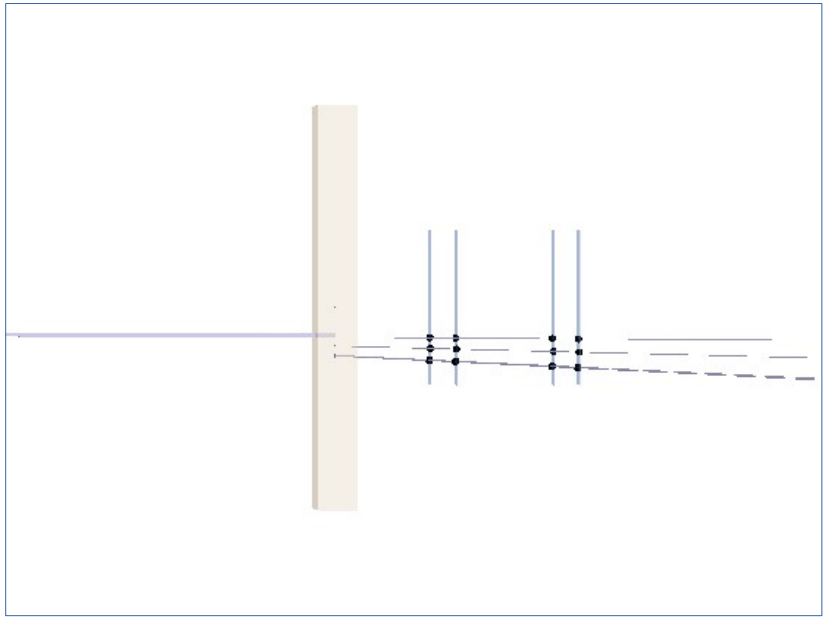
Fragmentation in the SC

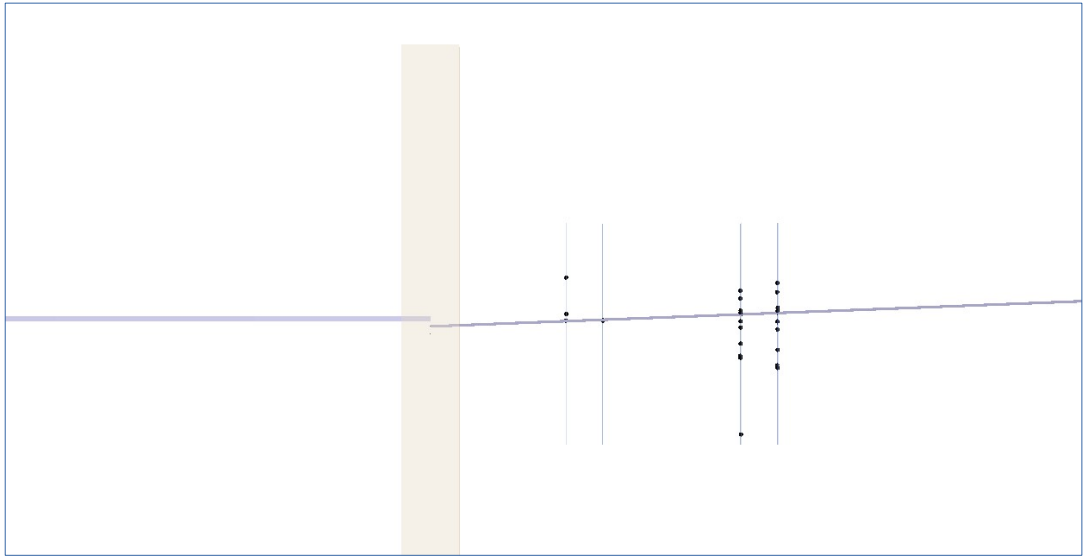


- The Oxygen changed its Mother ID (so the ID about how many times it interacted) from -1 (primary beam) to 0
- It probably emits gamma or lost a neutron



- Fragmentation in air before the target (at -5 cm) or after (at 1.5 cm)



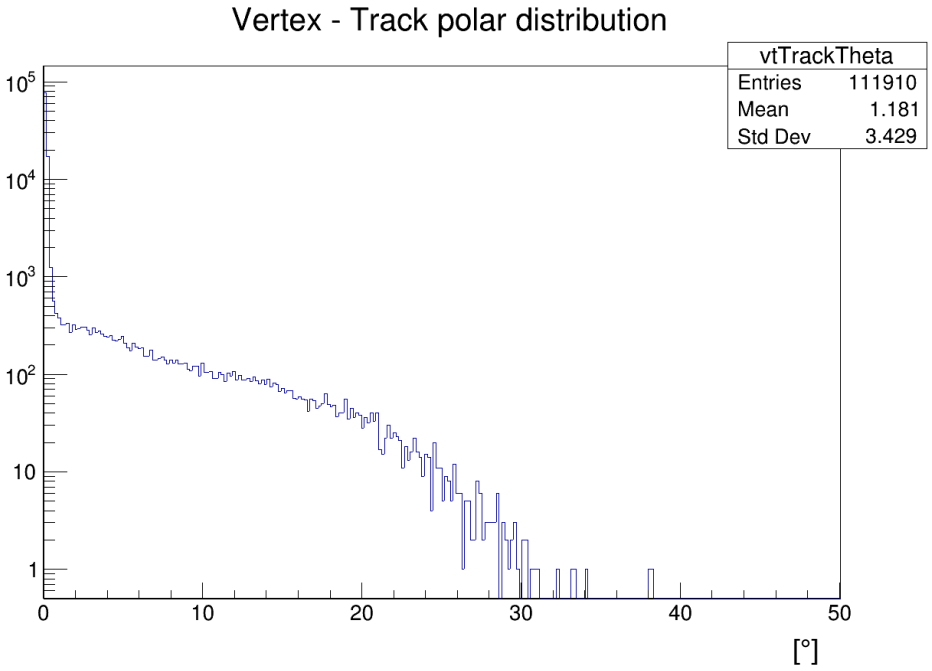


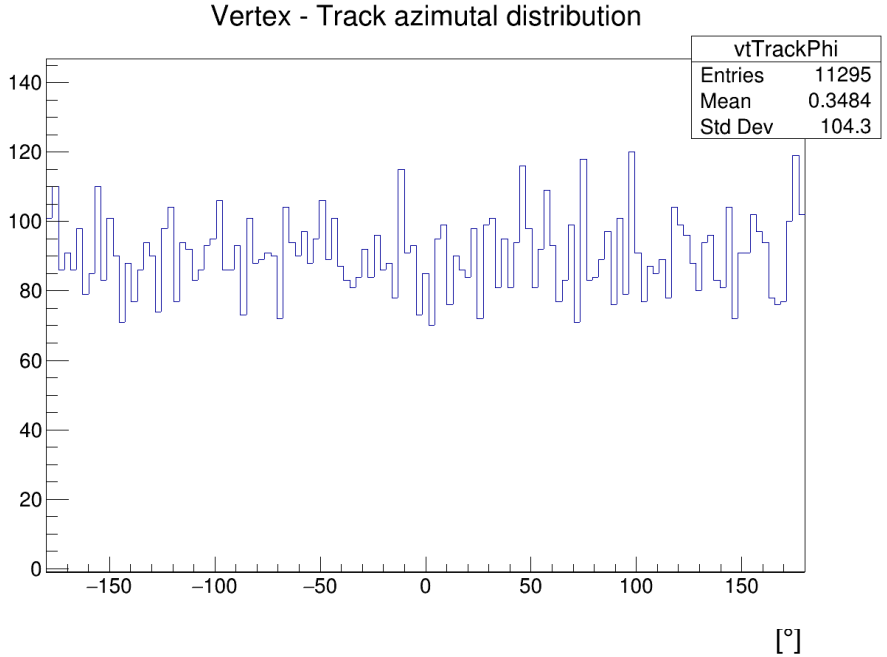
Observations

- The shown situations are such that the tracks were well reconstructed by the algorithm but in situations of fragmentation out of target. They should be considered in a detailed way.

Resolution Measurements

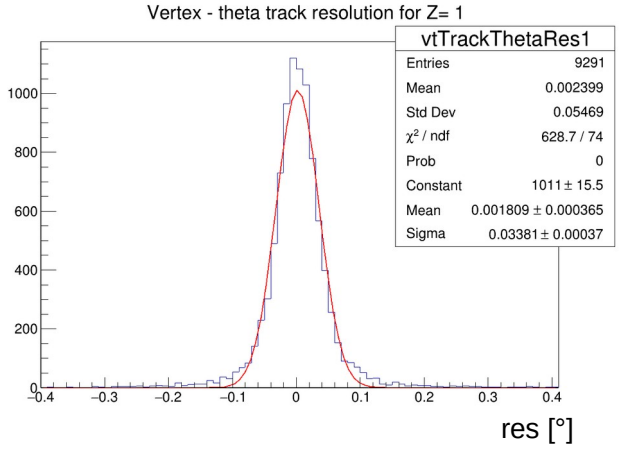
Vertex track angular distribution

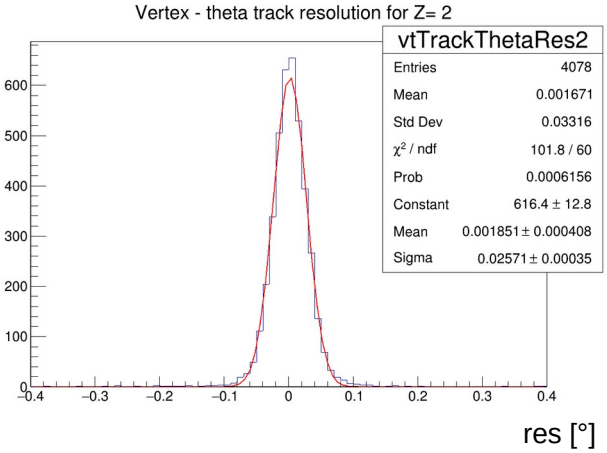


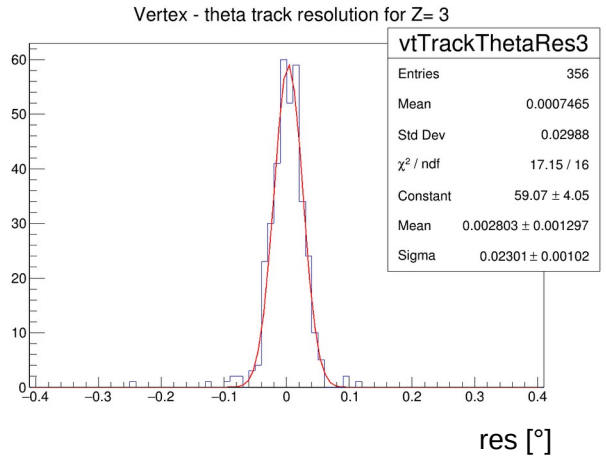


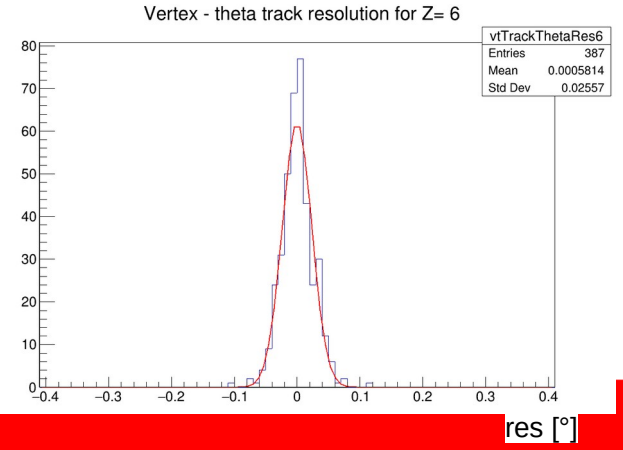
Track Theta Resolution

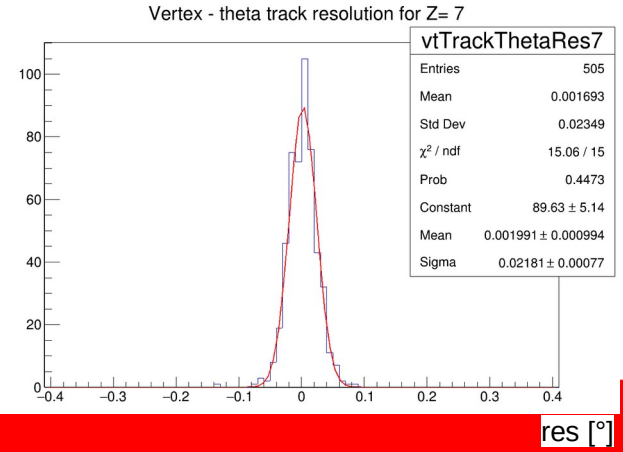
Polar angle resolution of the reconstructed track vs the mc particle: $res(\theta) = \theta_{reco} - \theta_{MC}$

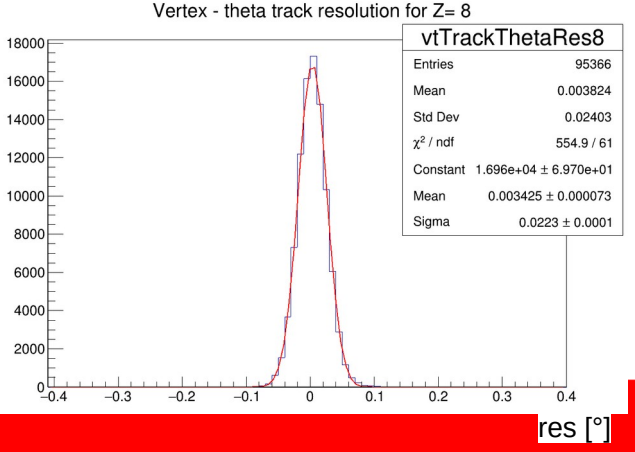






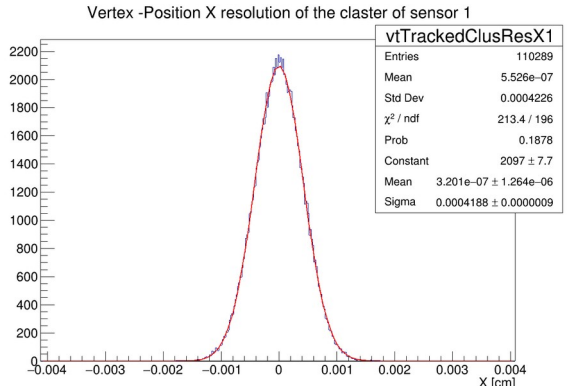


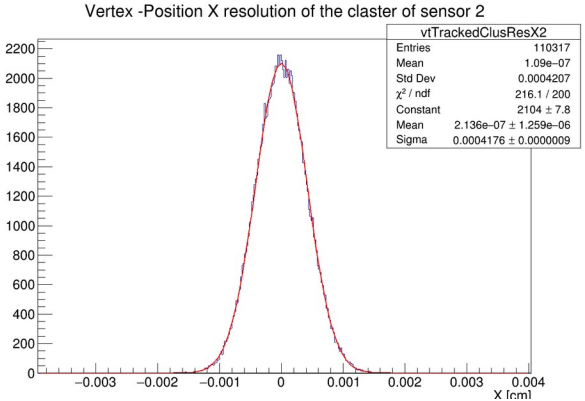




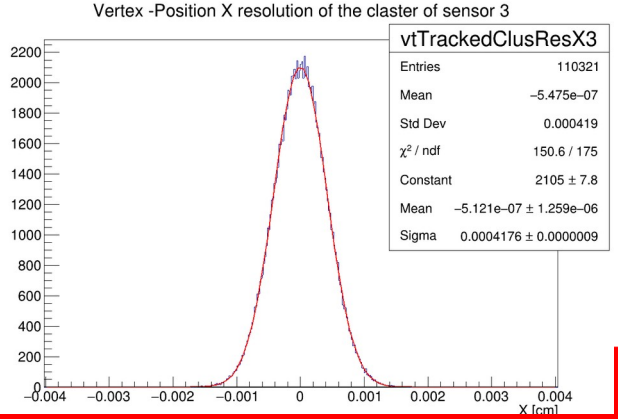
Track – Cluster position Resolution

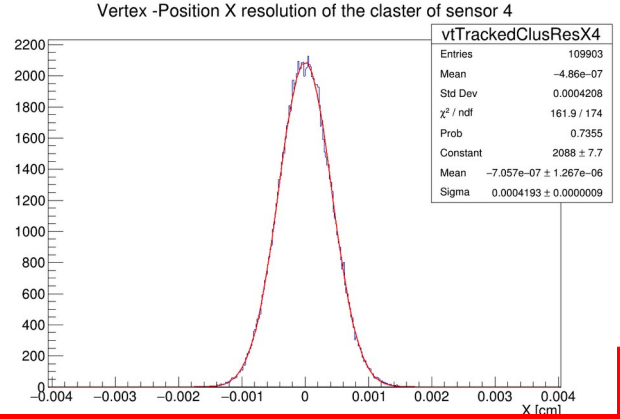
Position resolution of the reconstructed cluster of a track vs the
MC Hit (from which the cluster is generated) for every sensor of the vertex in X and Y





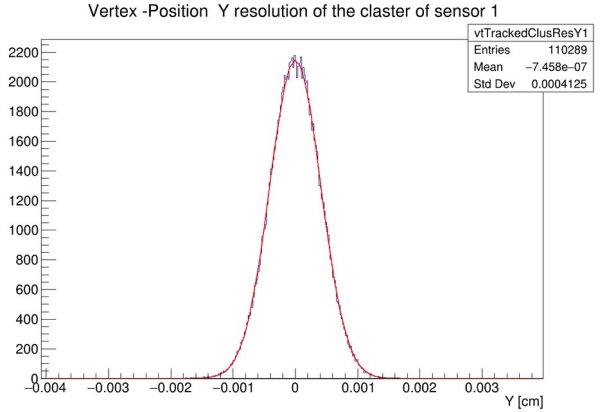
$$res(X) = X_{clus} - X_{MChit}$$

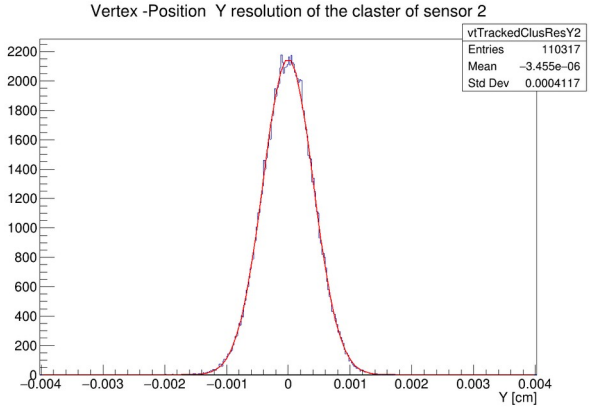




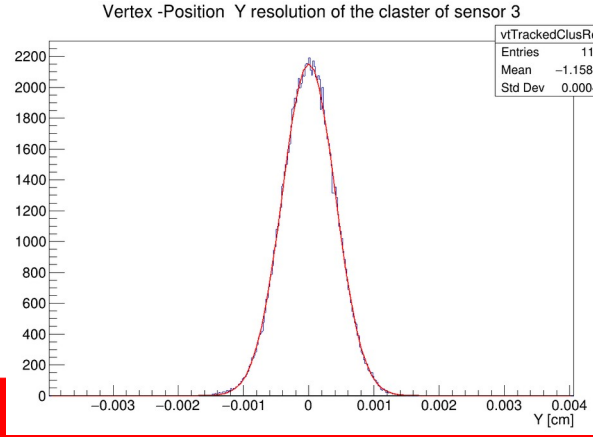
Track – Cluster position Resolution

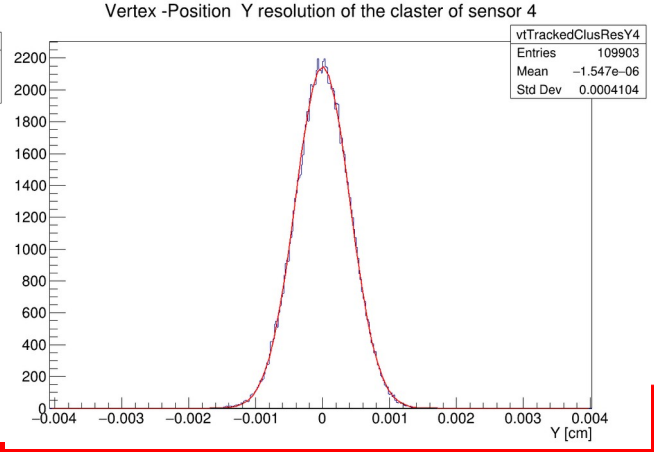
Position resolution of the reconstructed cluster of a track vs the
MC Hit (from which the cluster is generated) for every sensor of the vertex in X and Y





$$res(Y) = Y_{clus} - Y_{hit}$$





back up slides

N reference

- **Reference set: $N_{reference}$ (truth side)**
  all the tracks that an algorithm performing ideally should find and reconstruct:
  - all the tracks associated to a MC particle that crosses the FOOT apparatus at least until the last plane of the vertex (using MCRegion)
    - beam
    - primary fragment generated in the target

```
if (
    (particle_ID == 0                // if the particle is a primary OR
    || (Mid == 0 && Reg == target_region) // if the particle is a primary fragment born in the target
    ) &&  particle->GetCharge() > 0 &&
    particle->GetCharge() <= primary_charge && // with reasonable charge
    OldReg == last_vtx_plane_region && NewReg == RegAirAfterVtx // it crosses the last plane of the vertex
)
 return true;
```

Tracks algorithms

Two tracks algo, both based on a combinatorial approach.

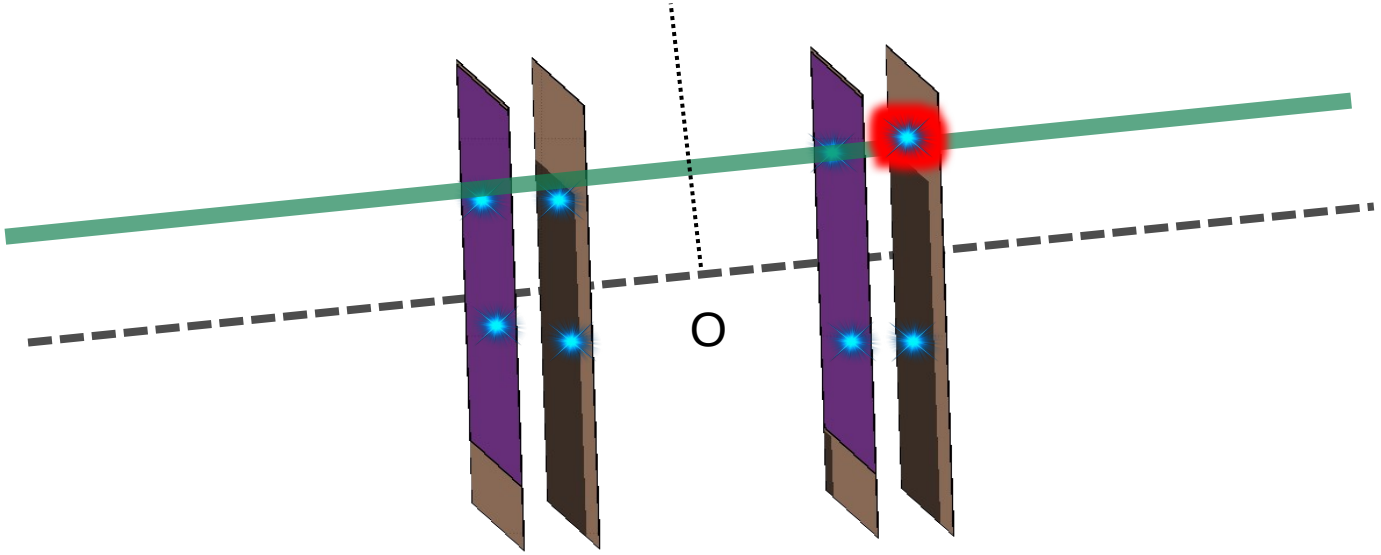
- FindStraightTracks
- FindTiltedTracks

Two different approach studied

- Already implemented: FindStraightTracks + FindTiltedTracks
- New: only fixed FindTiltedTracks (which takes both straight and tilted tracks in a while)

The following results are only for FindTiltedTracks algorithm

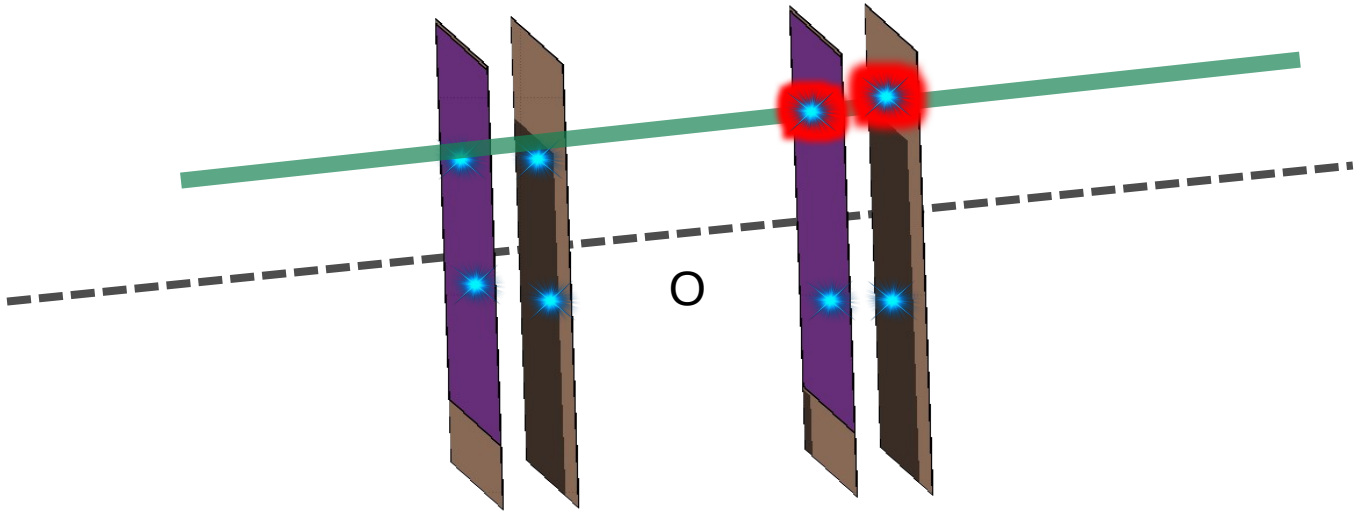
FindStraightTracks()



- loop on cluster of last plane
  - First track line: m=0; q(x,y) = position of the first cluster

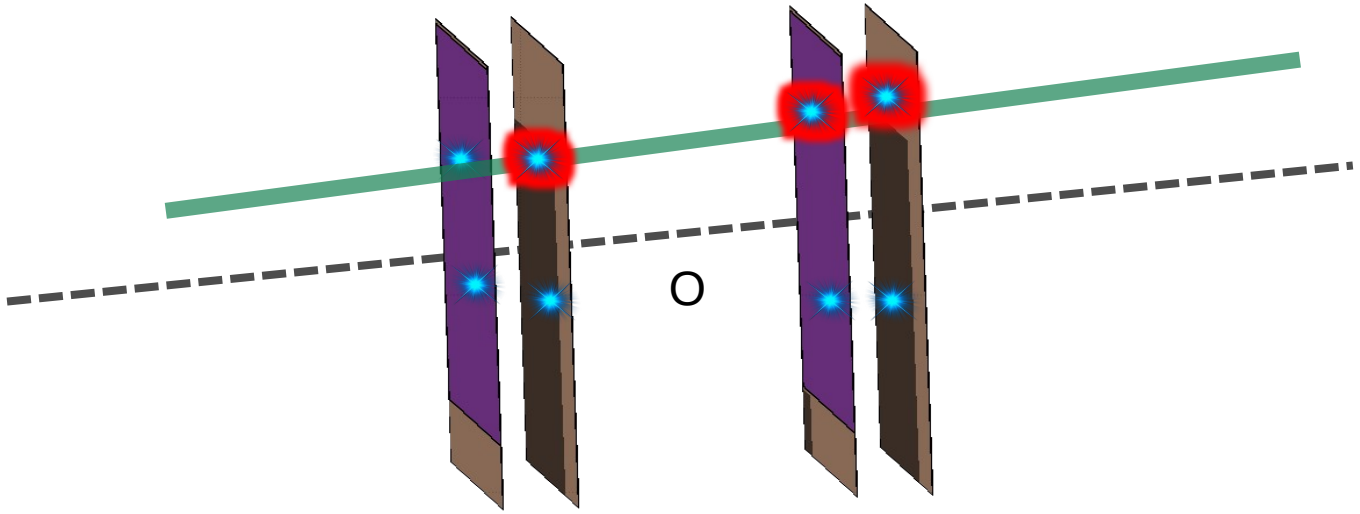
bias?

FindStraightTracks()



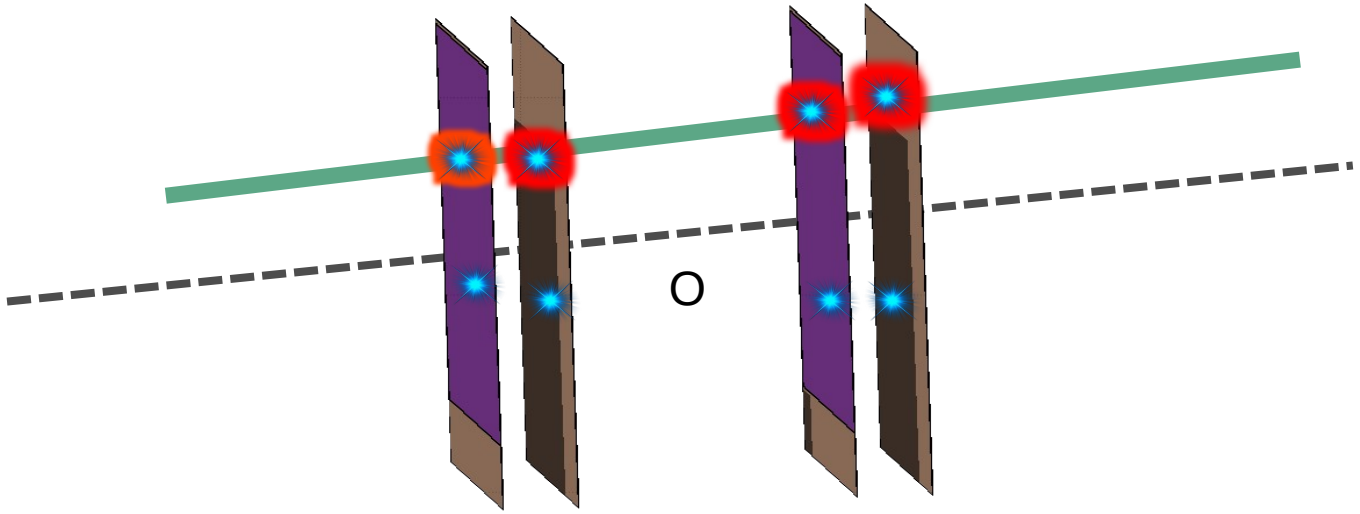
- loop on cluster of last plane
  - First track line: m=0; q(x,y) = position of the first cluster
    - loop on previous plane: I take only cluster with a distance lower than **30 micrometer**
      - **I do a linear fit with (x,y) of the clusters**

FindStraightTracks()



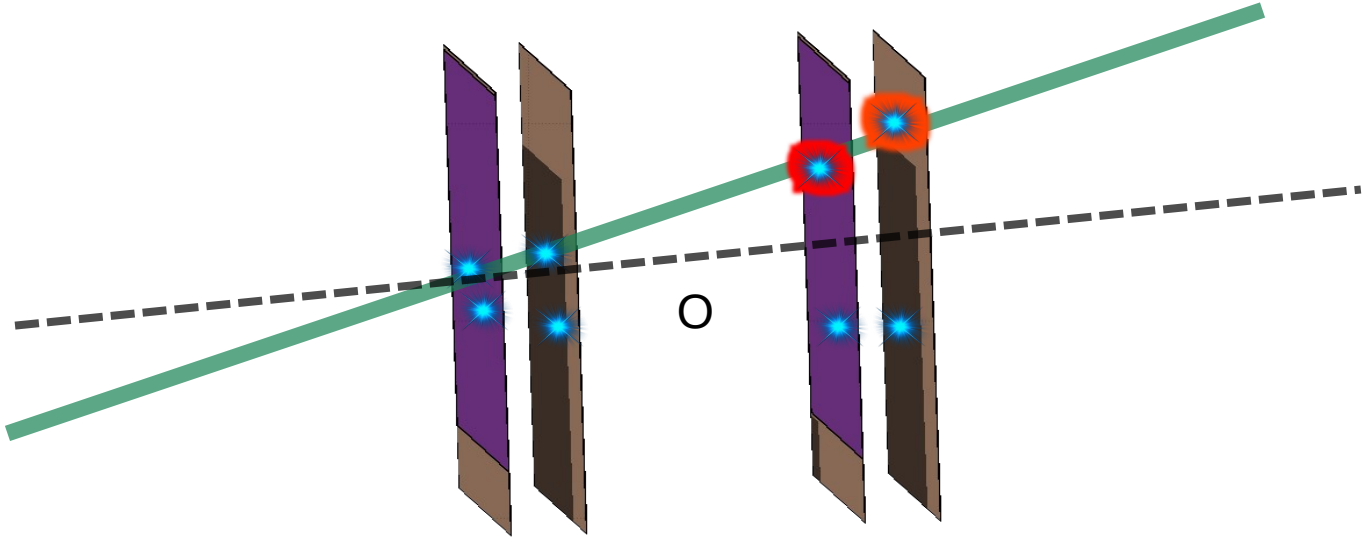
- loop on cluster of last plane
  - First track line: m=0; q(x,y) = position of the first cluster
    - loop on previous plane: I take only cluster with a distance lower than **30 micrometer**
      - **I do a linear fit with (x,y) of the clusters**

FindStraightTracks()



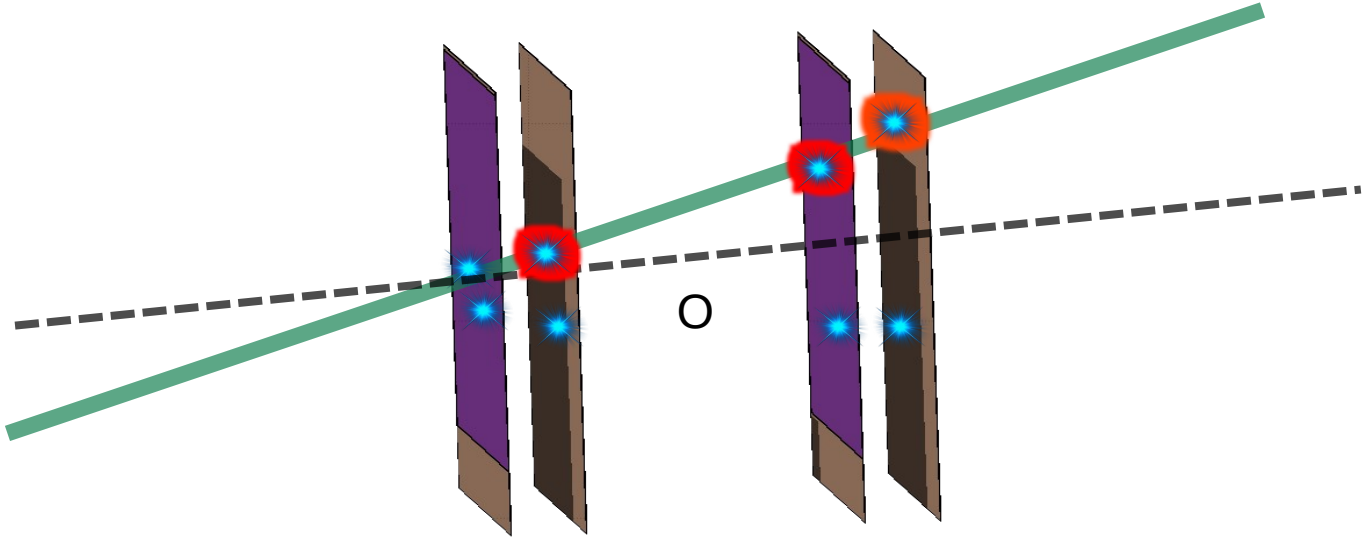
- loop on cluster of last plane
  - First track line: m=0; q(x,y) = position of the first cluster
    - loop on previous plane: I take only cluster with a distance lower than **30 micrometer**
      - **I do a linear fit with (x,y) of the clusters**

FindTiltedTracks()



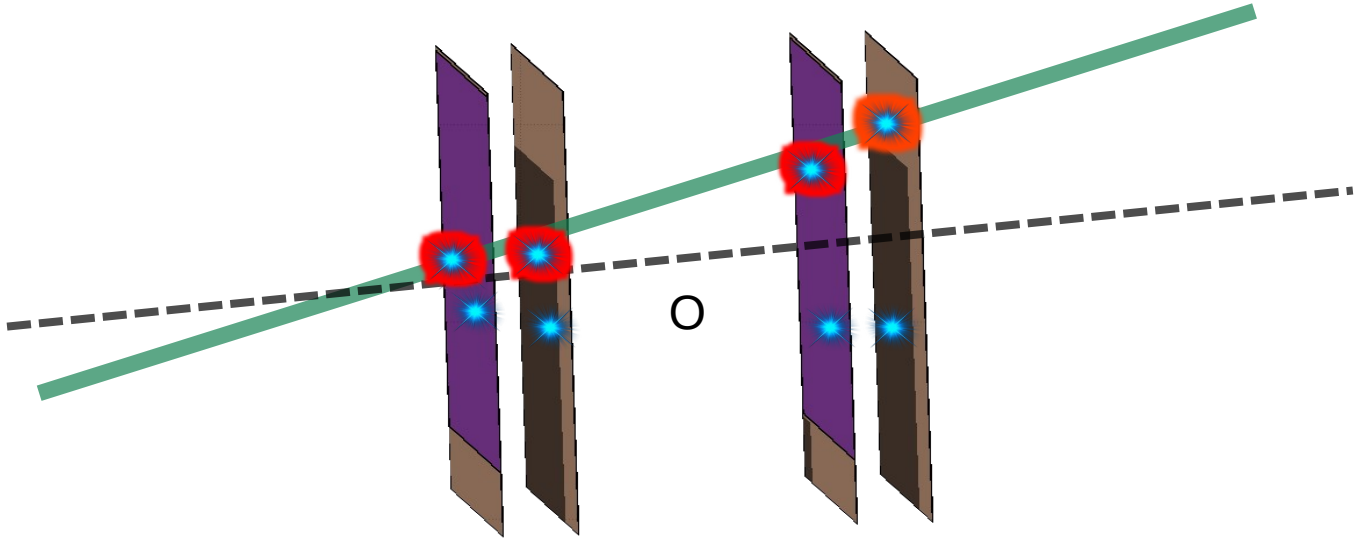
- loop on cluster of last plane
  - loop on cluster of previous plane (combinatorial of every cluster)
    - the first track is given as a fit between the first two points

FindTiltedTracks()



- loop on cluster of last plane
  - loop on cluster of previous plane (combinatorial of every cluster)
    - the first track is given as a fit between the first two points
    - loop on previous plane: I take only cluster with a distance lower than **30 micrometer**
      - **I do a linear fit with (x,y) of the clusters**

FindTiltedTracks()



- loop on cluster of last plane
  - loop on cluster of previous plane (combinatorial of every cluster)
    - the first track is given as a fit between the first two points
    - loop on previous plane: I take only cluster with a distance lower than **30 micrometer**
      - **I do a linear fit with (x,y) of the clusters**