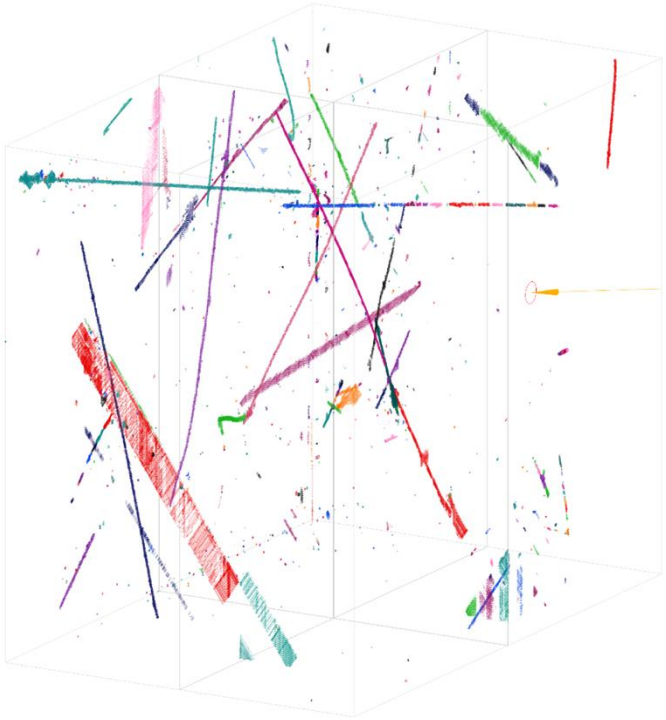
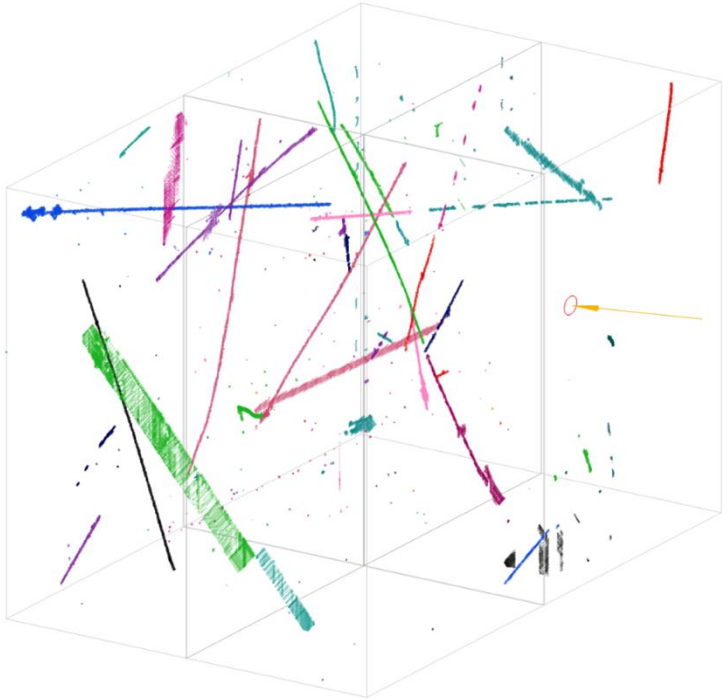
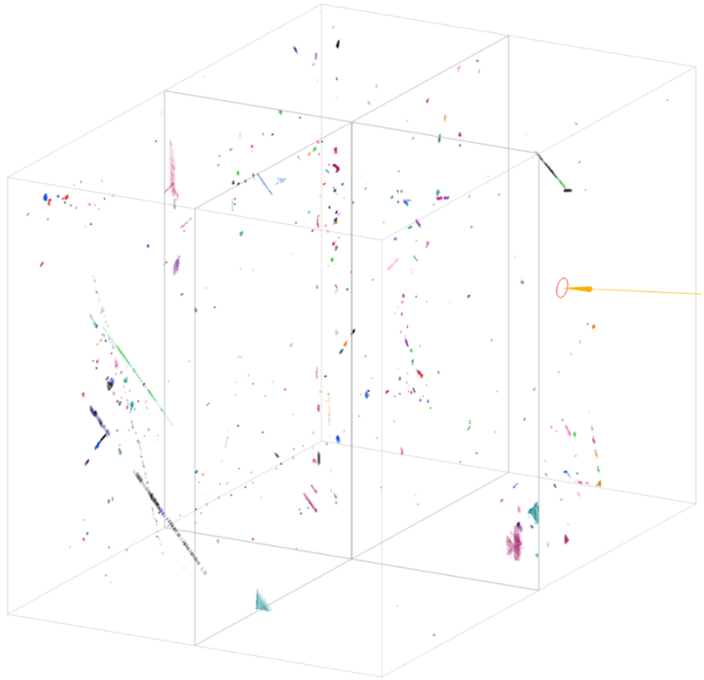
Separate ghost track



Initial imaging

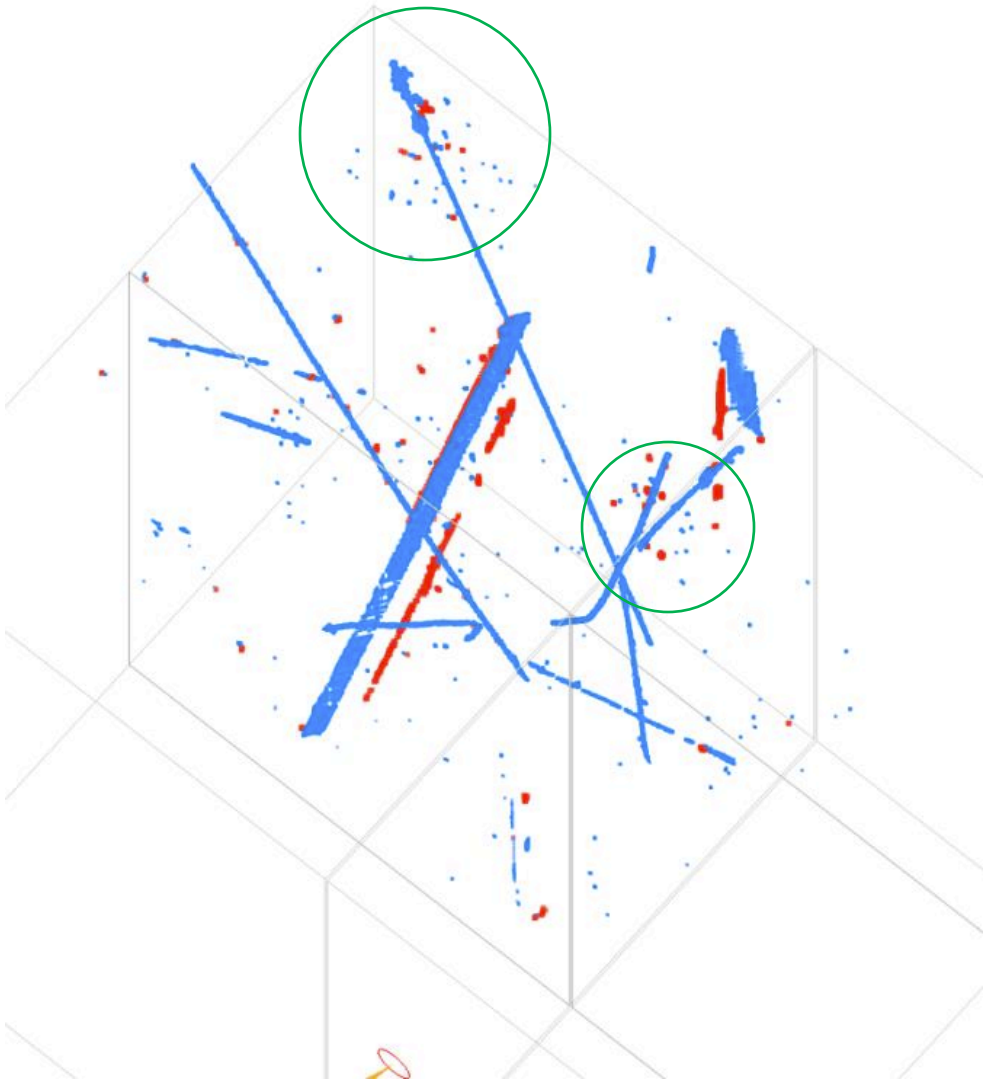


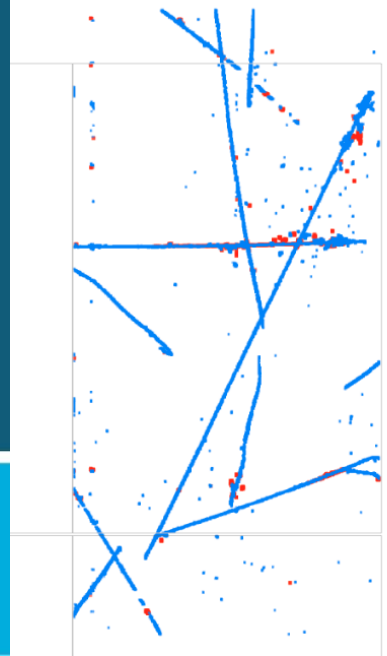
After 3-live deghosting



Ghost view

Check ghost track overlap with true tracks





- It will wrongly remove some true dots around the track
- Simply tune the parameters not perform well.
- True dots always near the track that “cover” them in 3D.
- Need to somehow separate true dots and ghost dots.

https://www.phy.bnl.gov/twister/bee/set/b4543391-954b-4a94-843c-a9200dc43285/event/0/

Deghosting in cluster; previous

Start from simple, add a *deghosting2* method.

- All clusters lined up based on its length.
- Pairwise comparison all live clusters in 3 projection view (reference vs target), reference is the longer one
  - Generate *DynamicPointCloud* of the reference cluster, it contains information of 3 projections
  - For each point in target cluster, get the nearest points in 2D view in reference cluster , then we can get a distance
    - *get_closest_2d_point_info()*
  - Define “covered”: most of the points (80%) are closer enough (5cm)to a reference cluster.

- If one cluster is covered by any others in 2view, remove it.

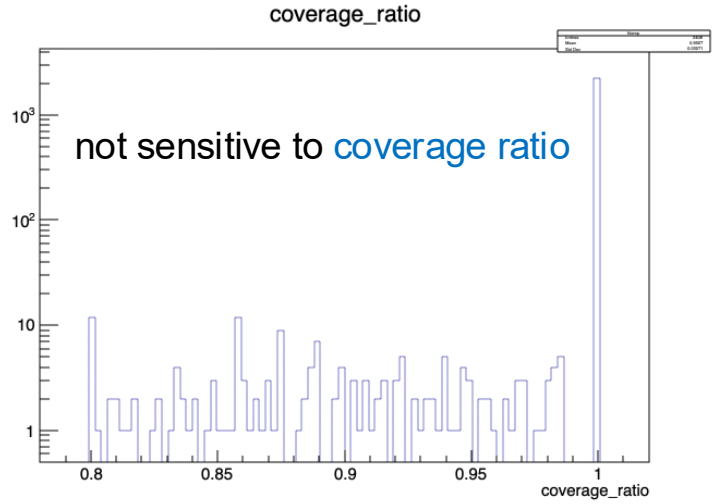
- Apply it twice, after extend and after connect1

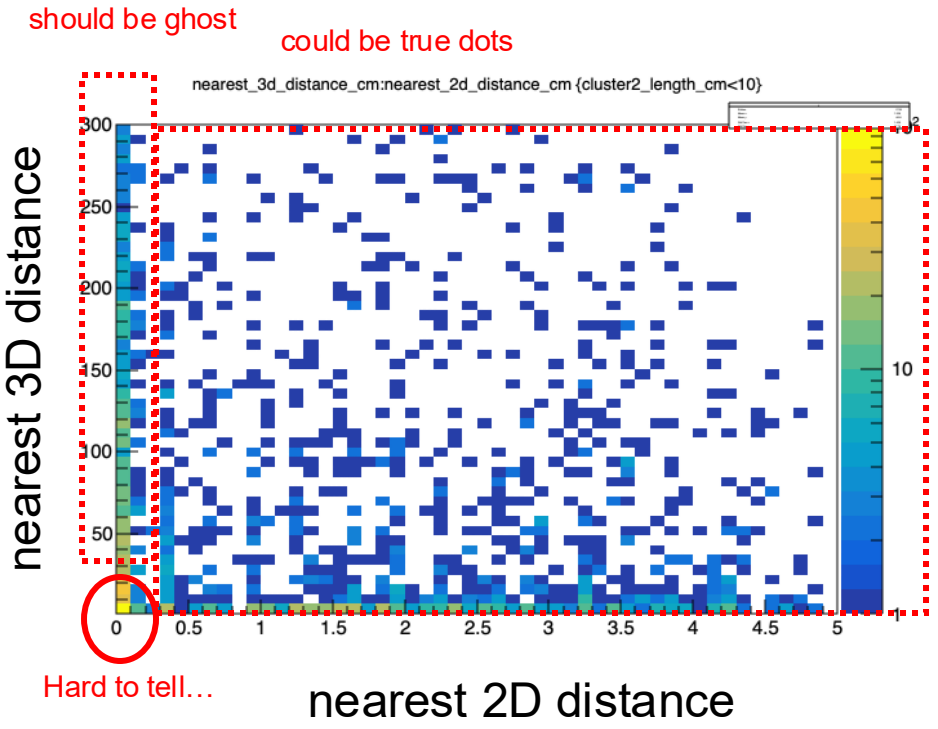
```
local cm_pipeline = [
    cm.pointed(),
    // cm.ctpointcloud(),
    cm.live_dead(dead_live_ove
    cm.extend(flag=4, length_
    cm.regular(name="-one", le
    cm.regular(name="_two", le
    cm.parallel_prolong(lengt
    cm.close(length_cut=1.2*w
    cm.extend_loop(num_try=3)
    cm.separate(use_ctpc=true
    cm.connect1(),
```

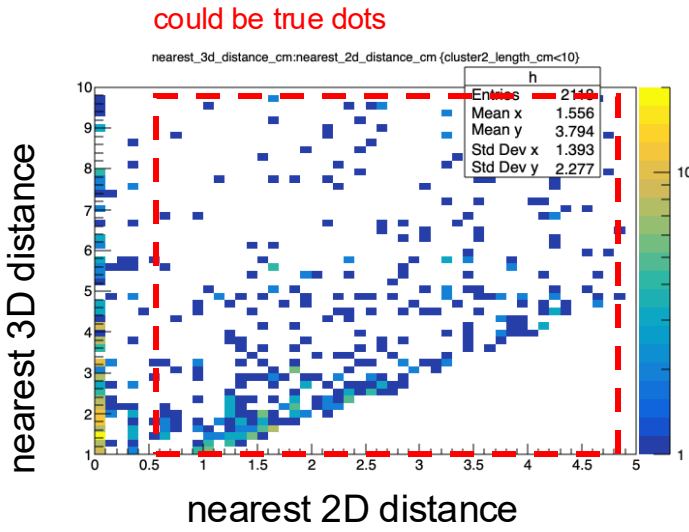
Extract more information

- "1st covered": most of the points (80%) are closer enough (5cm)to a reference cluster.
- For each covered pair, extract more parameters, save them into a TTree

```
struct CoverageInfo {
    int cluster1_id;
    int cluster2_id;
    std::string coverage_plane;
    double nearest_3d_distance;
    double nearest_2d_distance;
    double cluster1_length;
    double cluster2_length;
    int cluster1_time_min;
    int cluster1_time_max;
    int cluster2_time_min;
    int cluster2_time_max;
    int covered_points;
    int total_points;
    double coverage_ratio;
};
```





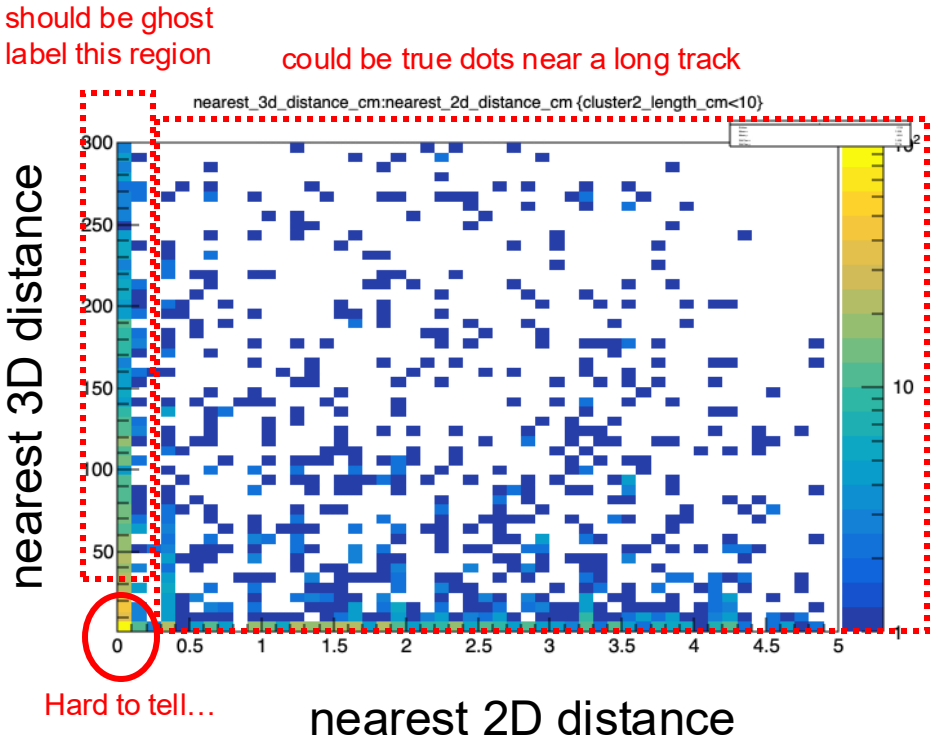


Re-Define “cover”

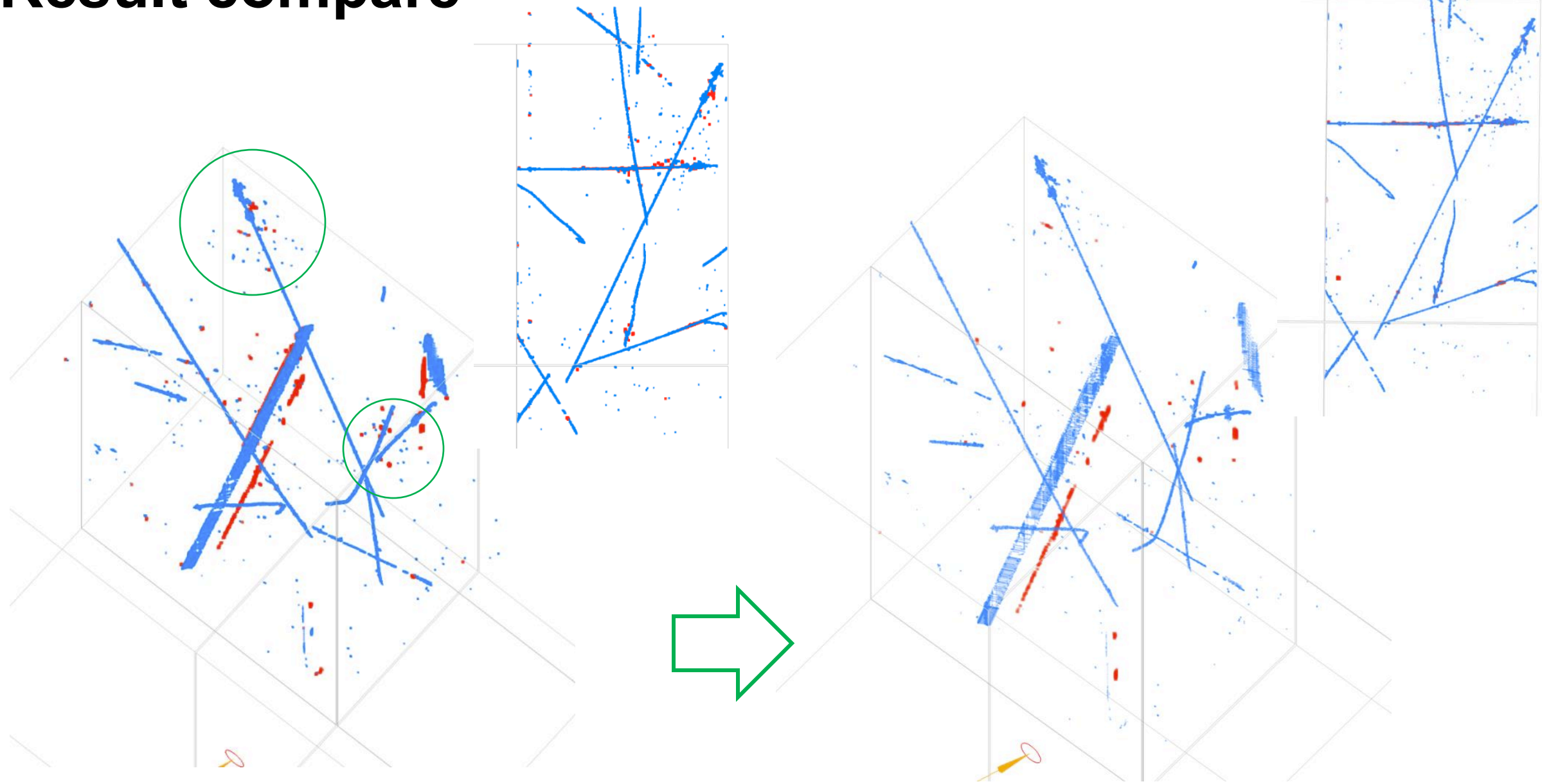
- “1st covered”:
  - most of the points (80%) from target are closer enough (5cm) to a reference cluster in a certain plane.
  - If some target is covered by more than 1 cluster in a plane, only leave the longest one as reference.

- “2nd covered”
  - 1st covered
  - nearest 3D distance >5cm && nearest 2D distance < 0.5 cm
  - A more rigorous definition.

- If it a cluster is 2nd covered in more than 2 planes, then it is labeled as a ghost.

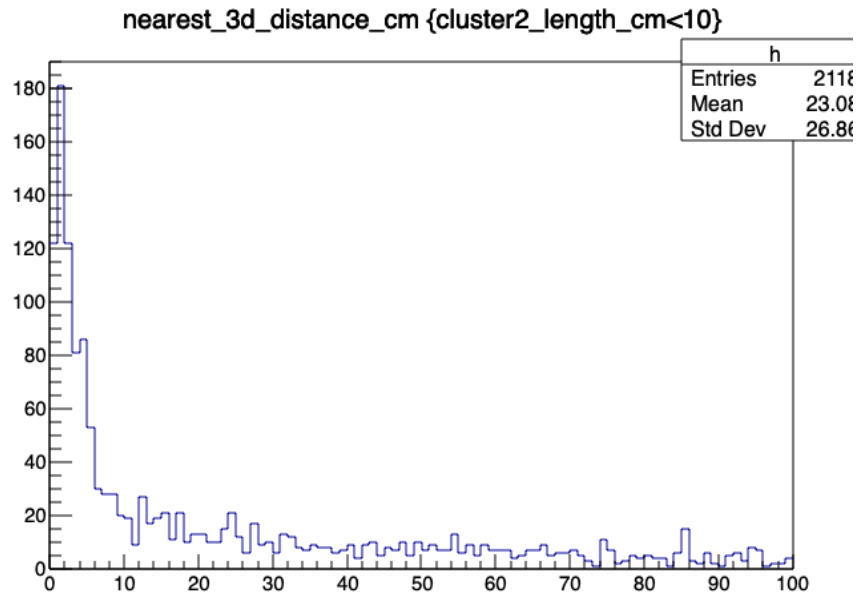


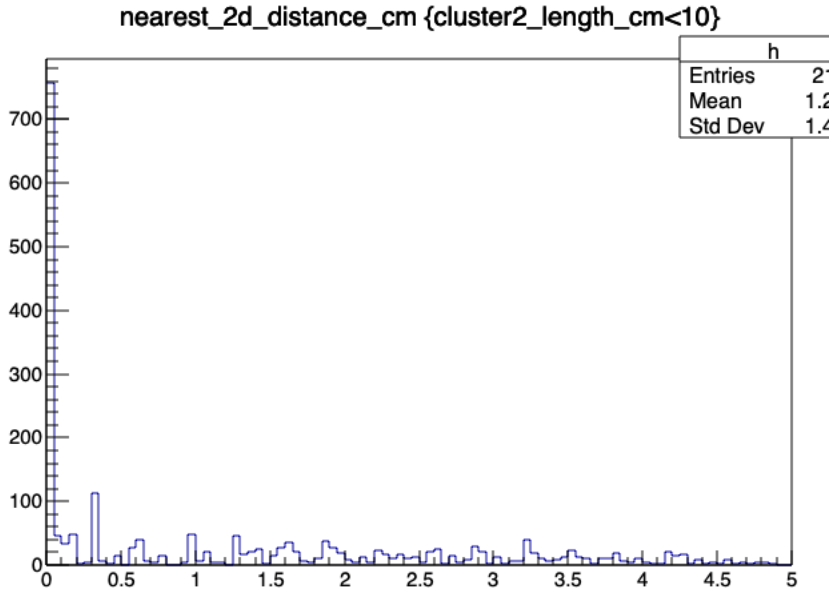
Result compare



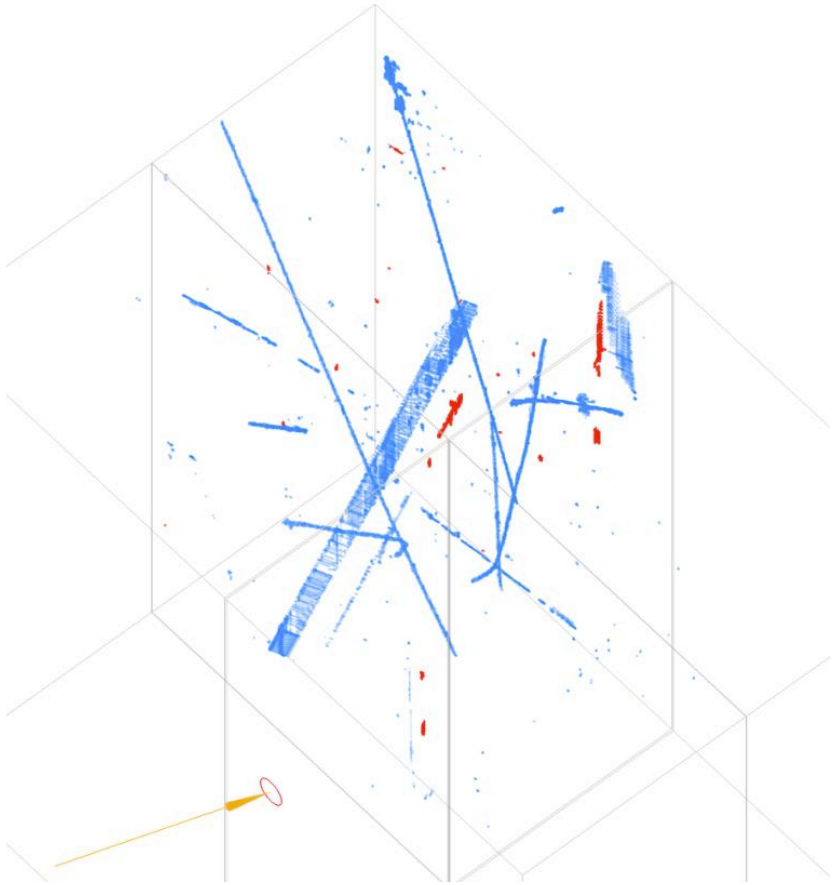
https://www.phy.bnl.gov/twister/bee/set/b25b887f-003b-41e7-b771-69a23b74182f/event/0/

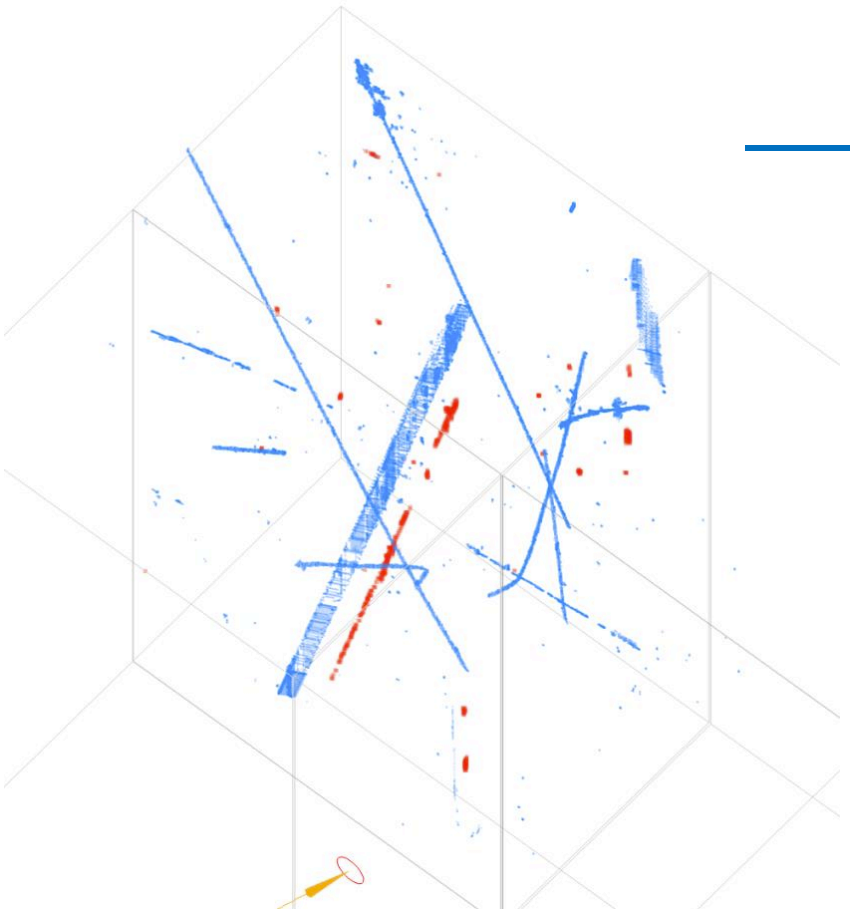
backup


nearest_3d_distance_cm {cluster2_length_cm<10}
h
Entries 2118
Mean 23.08
Std Dev 26.86
180
160
140
120
100
80
60
40
20
0
0 10 20 30 40 50 60 70 80 90 100


nearest_2d_distance_cm {cluster2_length_cm<10}
h
Entries 2
Mean 1.2
Std Dev 1.4
700
600
500
400
300
200
100
0
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5

backup





```
local cm_pipeline = [
    cm.pointed(),
    // cm.ctpointcloud(),
    cm.live_dead(dead_live_ove
    cm.extend(flag=4, length_
    cm.regular(name="-one", le
    cm.regular(name="_two", le
    cm.parallel_prolong(lengt
    cm.close(length_cut=1.2*we
    cm.extend_loop(num_try=3)
    cm.separate(use_ctpc=true
    cm.connect1(),
```