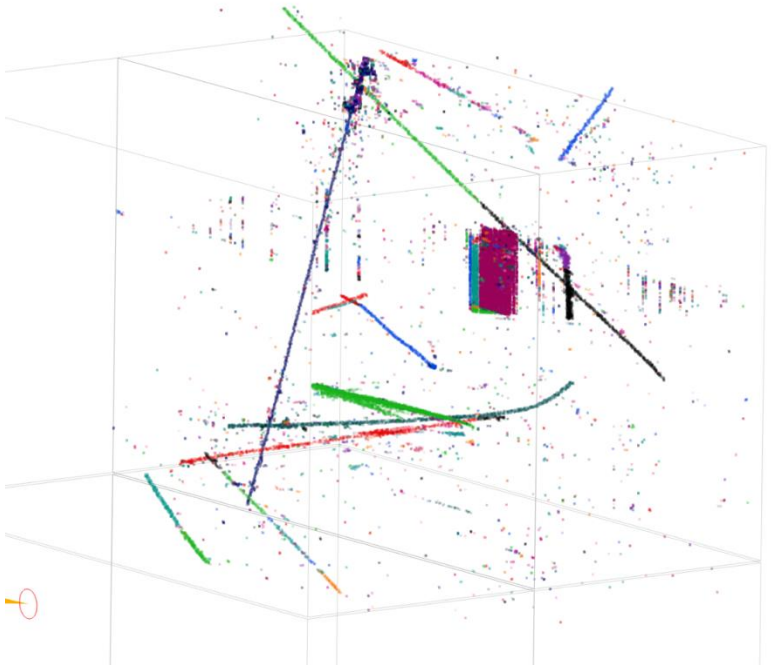
Update for cluster deghosting

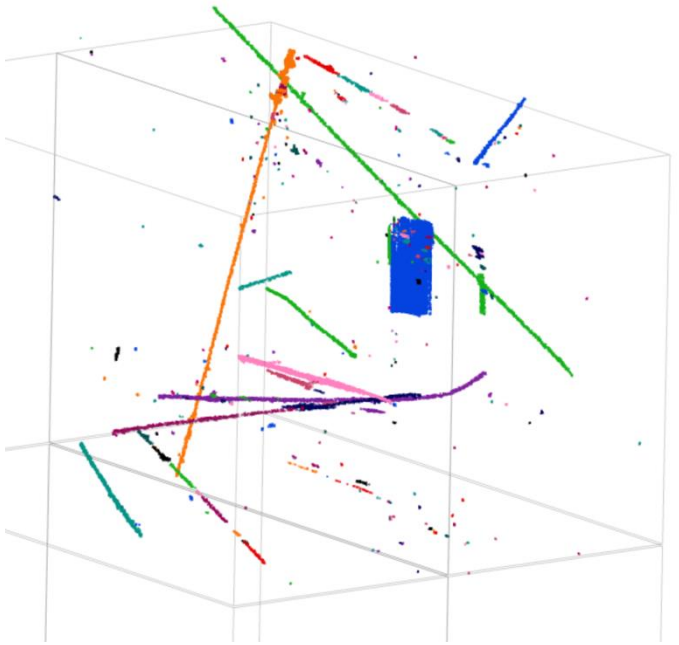
Xuyang Ning

Ghost track

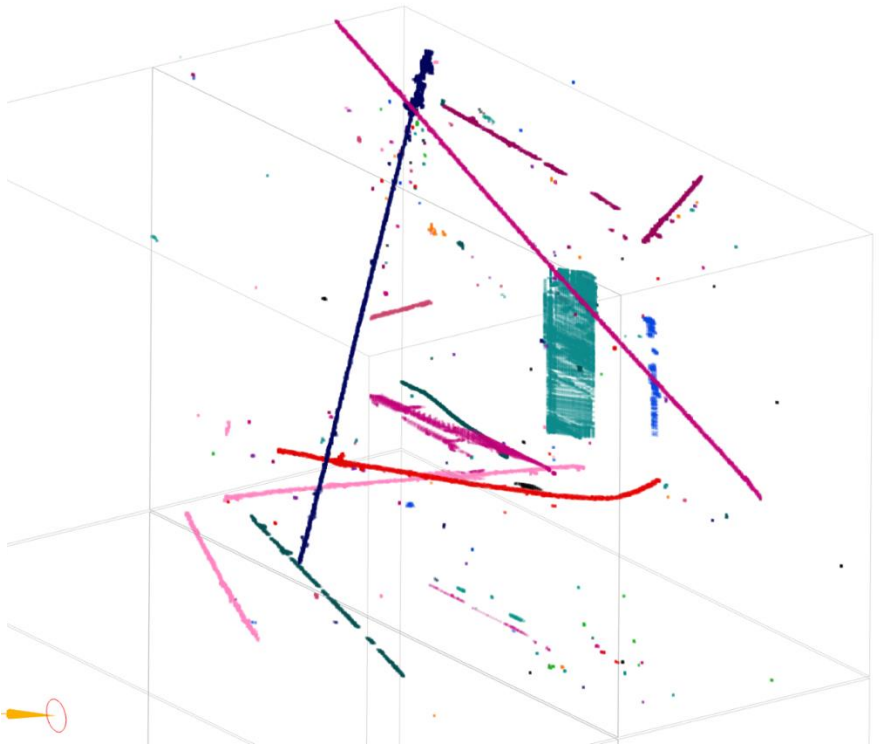
blobGrouping



Imaging



clustering



Clustering

PointTree Building

Point tree merging

Point tree merging

MultiAlgBlobClustering
Per face
only live

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

MultiAlgBlobClustering
Per APA
Include live and dead

```
local cm_pipeline = [
    cm.deghost(),
    cm.protect_overclustering(),
],
```

MultiAlgBlobClustering
All APA

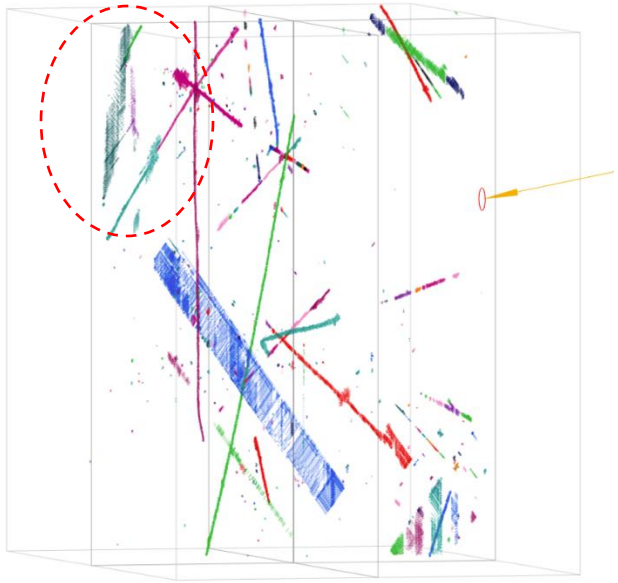
```
local cm_pipeline = [
    // cm_old.examine_x_boundary(),
    // cm_old.switch_scope(),

    cm.extend(flag=4, length_cut=60*wc.c
    cm.regular(name="1", length_cut=60*w
    cm.regular(name="2", length_cut=30*w
    cm.parallel_prolong(length_cut=35*wc
    cm.close(length_cut=1.2*wc.cm),
    cm.extend_loop(num_try=3),
    cm.separate(use_ctpc=true),
    cm.neutrino(),
    cm.isolated(),
    cm.examine_bundles(),
    cm.retile(cut_time_low=3*wc.us,
              cut_time_high=5*wc.us,
              anodes=anodes,
              samplers=[…
              ]),
],
```

- Result of individual step can be shown in Bee.
  - Easily check performance of each step, controlled by clus.jsonnet
- The target ghost tracks are live in 3 planes.
  - Add deghosting algorithm in MultiAlgBlobClustering Per face

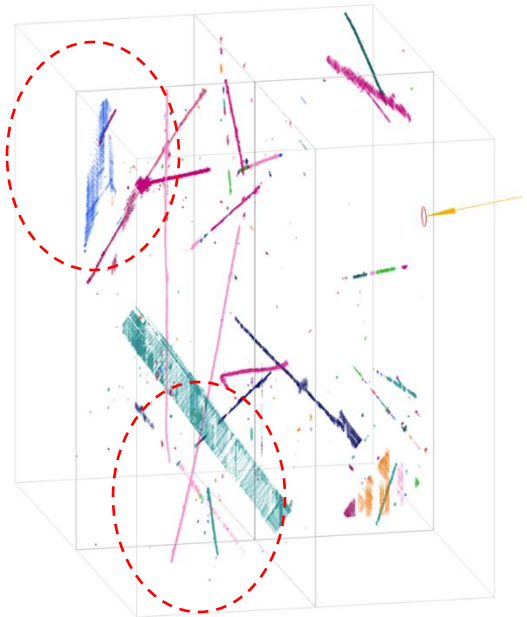
File sink

Step by step check

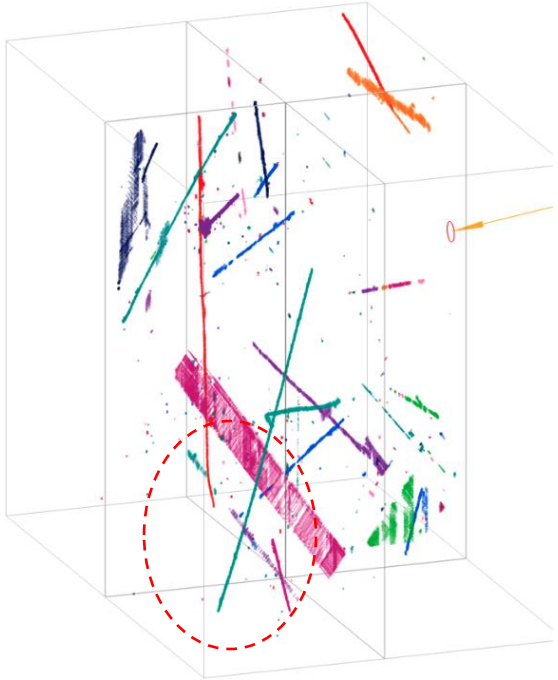
```
local cm_pipeline = [
    cm.pointed(),
    // cm.ctpointcloud(),
    cm.live_dead(dead_live_ov
    cm.extend(flag=4, length_
    cm.regular(name="-one", l
    cm.regular(name="_two", l
    cm.parallel_prolong(lengt
    cm.close(length_cut=1.2*w
    cm.extend_loop(num_try=3)
    cm.separate(use_ctpc=true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



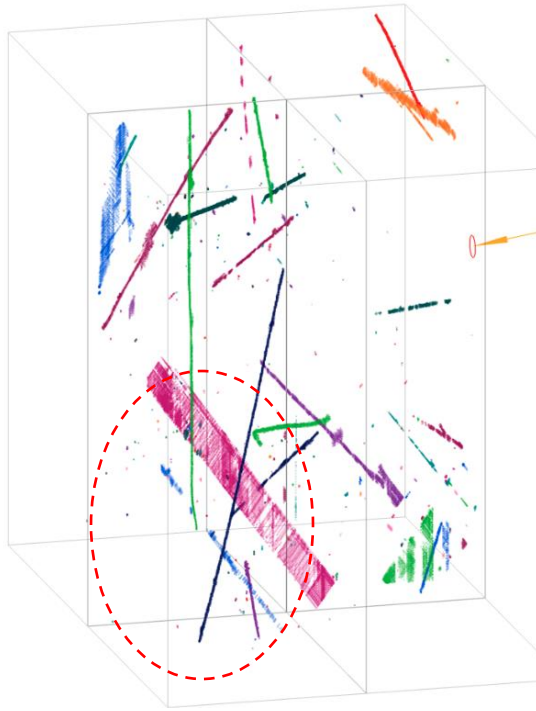
Initial imaging



After regular-one



After separate



After connect1

- Need deghosting several times at different place

Thoughts of the deghosting

```
local cm_pipeline = [
    cm.pointed(),
    // cm.ctpointcloud(),
    cm.live_dead(dead_live_overlap_off
    cm.extend(flag=4, length_cut=60*wc
    cm.regular(name="-one", length_cut
    cm.regular(name="_two", length_cut
    cm.parallel_prolong(length_cut=35*
    cm.close(length_cut=1.2*wc.cm),
    cm.extend_loop(num_try=3),
    cm.separate(use_ctpc=true),
    cm.connect1(),
    cm.deghost2(),
    // cm.isolated(),
    // cm.retile(cut_time_low=3*wc.us,
],
```

Start from simple, add a deghosting method after connect1.

On going (debugging…..)

- All clusters lined up based on its length
- Pairwise comparison all live clusters in 3 projection view (reference vs target), reference is the larger 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 (90%) are closer enough (5cm? To be tested…) to a reference cluster.

- If one cluster is covered by others in 2(or 3?) view, remove it.
  - Might need more detailed determination… to be tested.