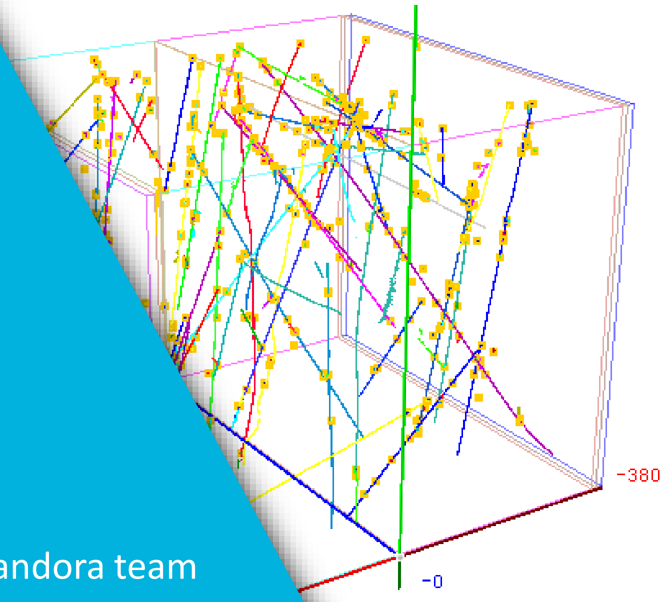
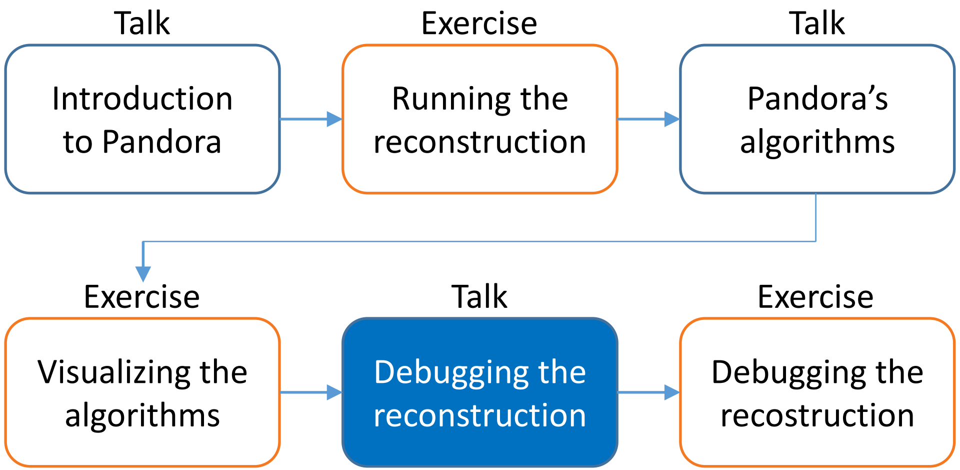
Debugging reconstruction

Maria Brigida Brunetti, Andy Chappell and Steve Dennis for the Pandora team

25/10/2023 8th UK LArTPC Software and Analysis Workshop



Reconstruction session



Credit: These slides are based on previous LArSoft workshop slides by Andrew Smith

Key references: [Pandora ProtoDUNE paper](#)
[Pandora MicroBooNE paper](#)

Reconstruction is hard

- The events you've looked at so far have been relatively simple
- Two track-like trajectories emerging from a common vertex
- Nonetheless, you may have seen some surprising reconstruction results, e.g.:

Split tracks

Holes in tracks

Wavy tracks

Merged tracks

Incorrect vertex

- Most neutrino interactions will be **more complex** than this:
  - Track-like and shower-like topologies
  - Re-interactions
  - High particle multiplicity

You will come across mis-reconstructed events!

What can go wrong?

- Any given reconstruction failure can depend on a **wide variety of minute details** of the event under consideration

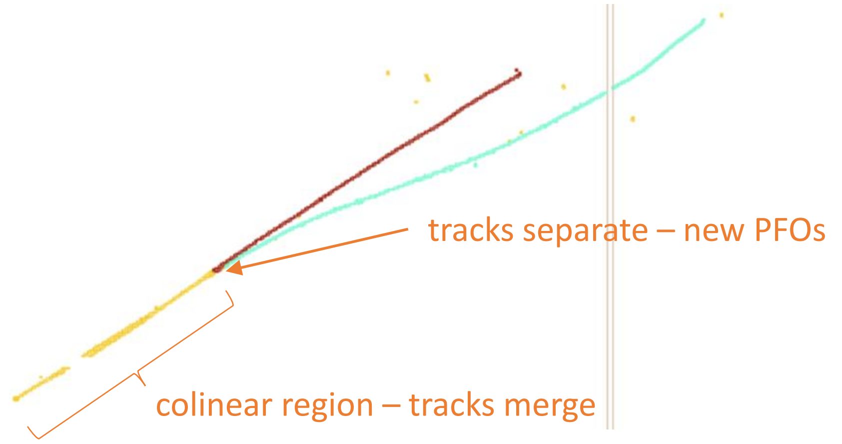
- However, a number of circumstances exist that more reliably cause problems
  - We'll give some **examples** in the next few slides!

- Combinations of these issues can lead to some bizarre reconstruction errors that make no sense unless you **walk through the sequence of (often small) mistakes** that produced the final outcome

Overlapping and back-to-back trajectories

- Neutrino interactions happen in 3D, but we have (typically) three, 2D projections of the interaction
- Trajectories that are clearly distinct in 3D can appear indistinguishable in a 2D projection
- If the trajectories can be distinguished in two of the projections, it is still possible to effectively reconstruct the 3D trajectories
- If overlap occurs in multiple views however, you'll likely lose a particle

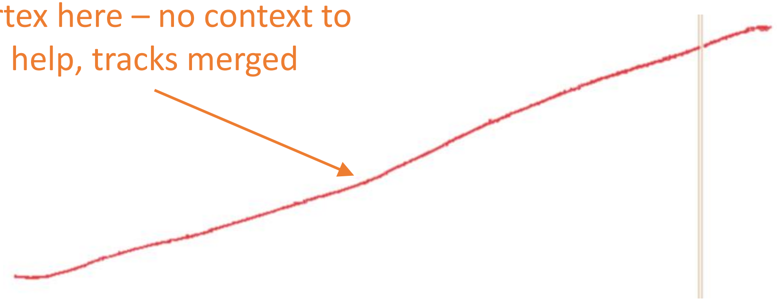
Example:
Tracks merged due to collinearity



Overlapping and back-to-back trajectories (2)

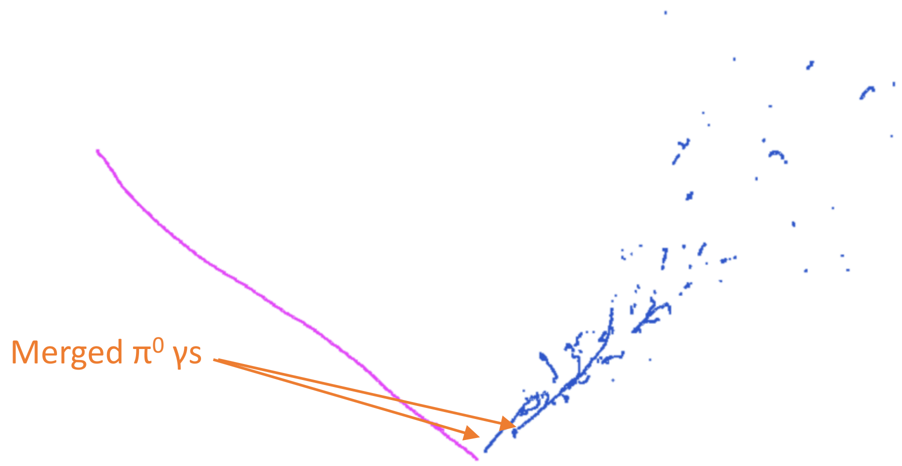
- Neutrino interactions happen in 3D, but we have (typically) three, 2D projections of the interaction
- Trajectories that are clearly distinct in 3D can appear indistinguishable in a 2D projection
- If the trajectories can be distinguished in two of the projections, it is still possible to effectively reconstruct the 3D trajectories

Example:
If one particle has a direction exactly opposite another, it's very likely the resultant straight-line will be reconstructed as a single particle



Small opening angles and sparse showers

- For showers, in particular, a small opening angle between two showers can make it challenging to determine to which shower the hits belong
- This can result in an incorrect distribution of hits among the showers, or to a complete merging of the two showers
- This is a common failure mode

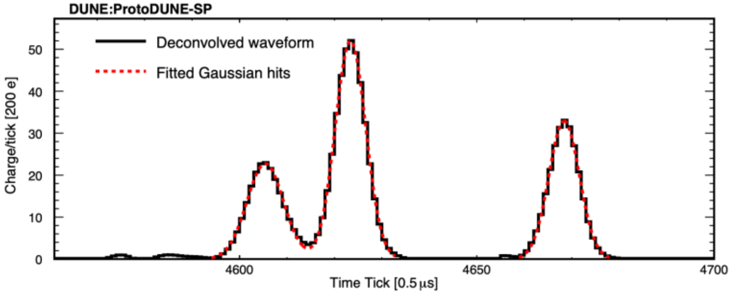


Small opening angles and sparse showers (2)

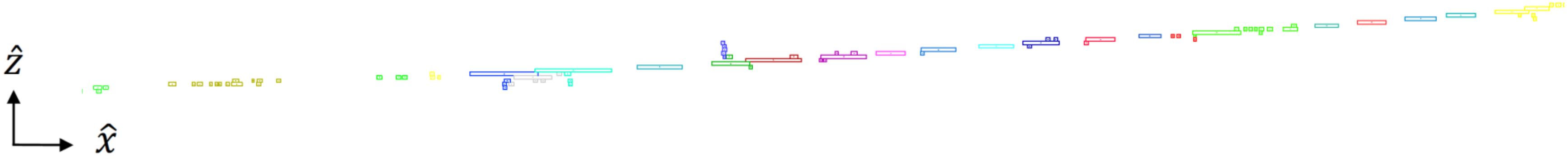
- The opposite of this problem is shower fragmentation
- If there are large gaps between the hits belonging to a given shower, it can be difficult to merge them together and so showers can be broken into multiple reconstructed particles

Awkward trajectories

- Pandora runs on discrete hits that it receives from signal processing steps that run before it
- However those hits are extracted from continuous waveforms produced by the drift electrons as they pass by induction planes and get deposited on the collection plane

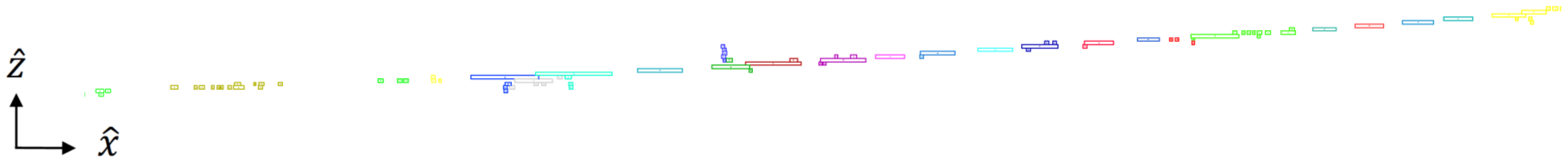


- If a particle trajectory is perpendicular to the wire planes, its drift electrons interact with a single wire/strip, from which it is challenging to extract hits
  - The result is **a small number of wide hits** (i.e. a high uncertainty in the position along the drift direction)
  - Such hits can be difficult to cluster



Awkward trajectories (2)

- This can also occur if the component parallel to the planes aligns with a wire/strip (though this will only affect one view)
- The final awkward trajectory is what we call the **isochronous case**, where each (most) points along a particle trajectory have a common x coordinate
  - This is not a problem for reconstruction within a single view, but matching clusters between views uses the common x coordinate as a means to relate the clusters and so having all of the hits sharing a common x coordinate can be very unhelpful



Identifying a misbehaving algorithm

- Different algorithms target different topologies and so use different criteria for decision making, which sometimes will be inappropriate

- If you see a reconstruction problem in your fully reconstructed event, it can be very useful to intercept the reconstruction at intermediate points to understand where things started to go wrong

- You can do this using the techniques from the previous exercise

  1) Add visualization algorithms at various points in the XML configuration
  2) Look to see if the clusters/PFOs at each point appear well reconstructed
  3) Make a judgment – for example, highly fragmented trajectories are often fine if the algorithm that targets these fragments hasn’t run yet

Identifying a misbehaving algorithm (2)

- If you find an algorithm that broke your event, you have two broad choices:

1. Tune the algorithm

to modify its decision making to avoid the mistake

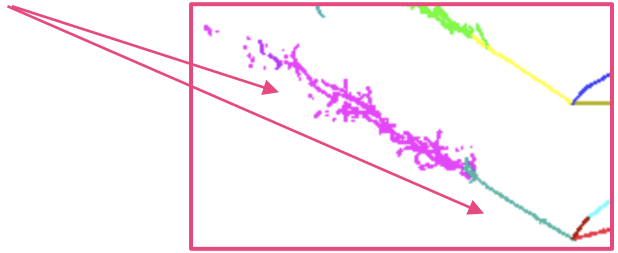
2. Develop a new algorithm

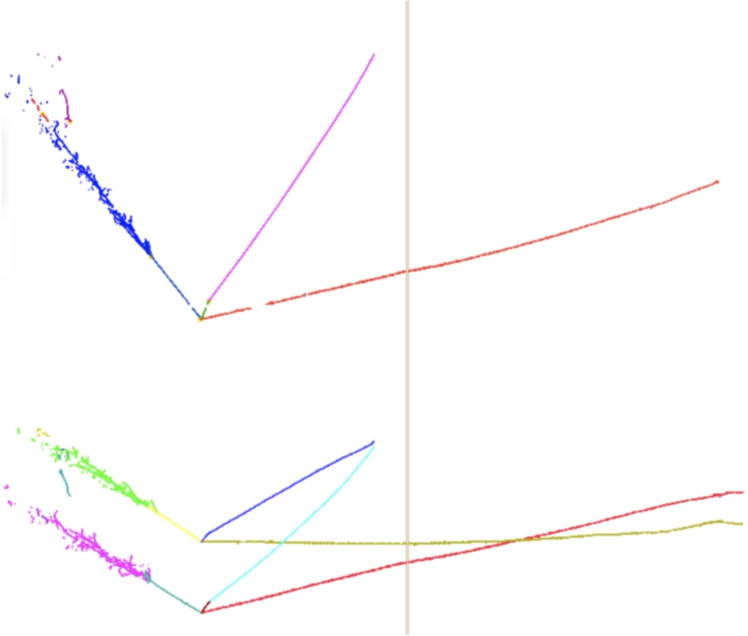
specifically designed to fix the kind of mistake you've found

- Today, we'll look at tuning, because you don't have time to develop a new algorithm, but keep in mind, if you tune things too much for one event, you'll break others

A mis-reconstructed primary electron

- The event we'll be looking at today mis-reconstructs a primary electron
  - The MIP-like stub connecting to the interaction vertex and the downstream cascade are reconstructed as separate particles



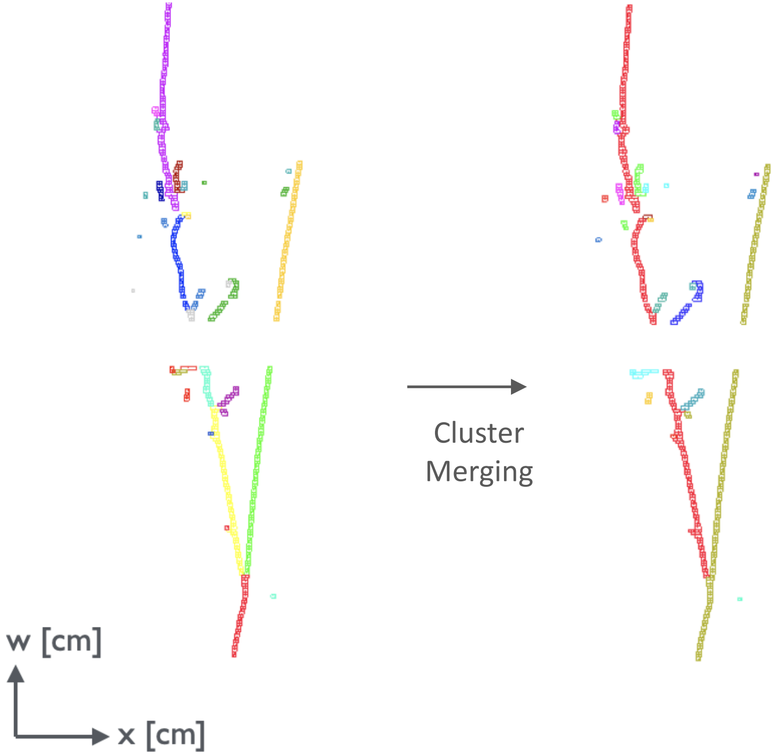


- The 2D algorithms we want to investigate are Longitudinal and Transverse association and extension algorithms
- We'll spend the next couple of slides getting an idea of what these algorithms do and then move onto the exercise to try to fix them

Longitudinal algorithms

- Given two relatively long (i.e. having well-defined directions) clusters, the longitudinal algorithms will check to see that they are end-to-end
    - Small, partially configurable, tolerances for gaps between clusters or overlap of clusters are allowed
- During association fits are performed of the upstream cluster and the downstream cluster
    - Directions must align to within a specified angular tolerance
    - Extremal points of the fits must be within specified longitudinal and transverse displacements
    - If these checks pass, clusters can be merged

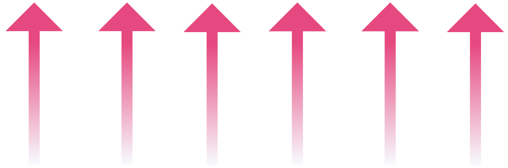
Example: Longitudinal Association algorithm



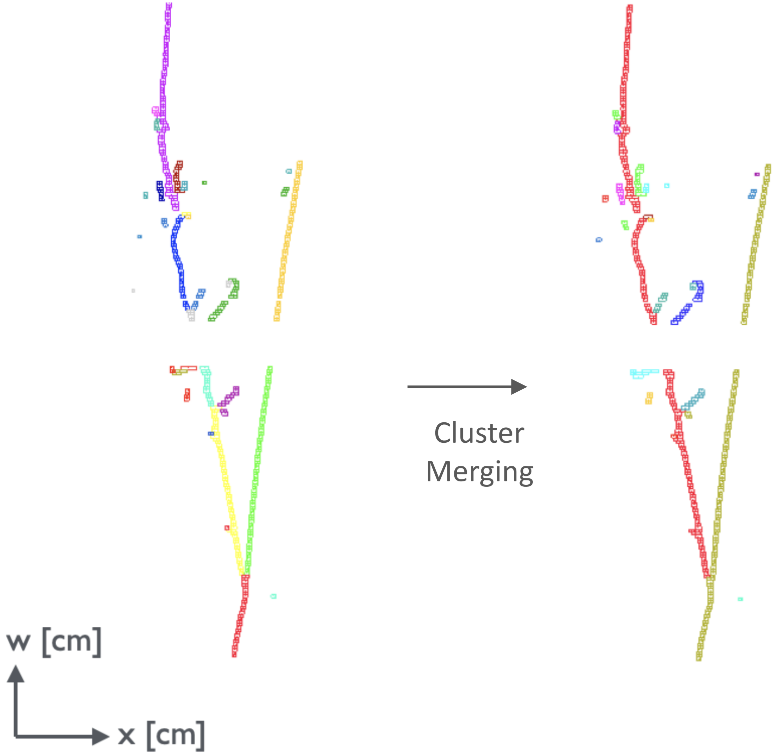
Longitudinal algorithms

- During extension fits are performed that consider local direction at the extrema of clusters
  - Vertices must be within specified distances
  - Local directions should align within specified angular tolerances and impact parameters must be within specified displacements
  - If these checks pass, clusters can be merged

You can influence these decisions by tweaking allowed displacements and angular tolerances!



Example: Longitudinal Association algorithm



Transverse algorithms

- Transverse tracks tend to be somewhat shorter and more fragmented in early clustering, so direction tends to be less well-defined
  - Clusters are grouped into short, medium and long groups
  - Try to establish associations between short clusters (look for proximity of closest approach, allowing for some specified overlap)
  - An association must be between the two closest clusters, “jumping over” clusters not allowed
  - Attempt to define a direction by fitting a set of associated transverse clusters
  - Attempt to associate these groups of clusters based on proximity and direction

Transverse algorithms (2)

- During transverse extension, fits are performed that consider local direction at the extrema of clusters, much as in the longitudinal case

  - Vertices must be within specified distances
  - Local directions should align within specified angular tolerances and impact parameters must be within specified displacements
  - If these checks pass, clusters can be merged

- You can influence these decisions by tweaking allowed displacements and angular tolerances!

We will now try to fix a broken event