Event Reconstruction in LArTPC

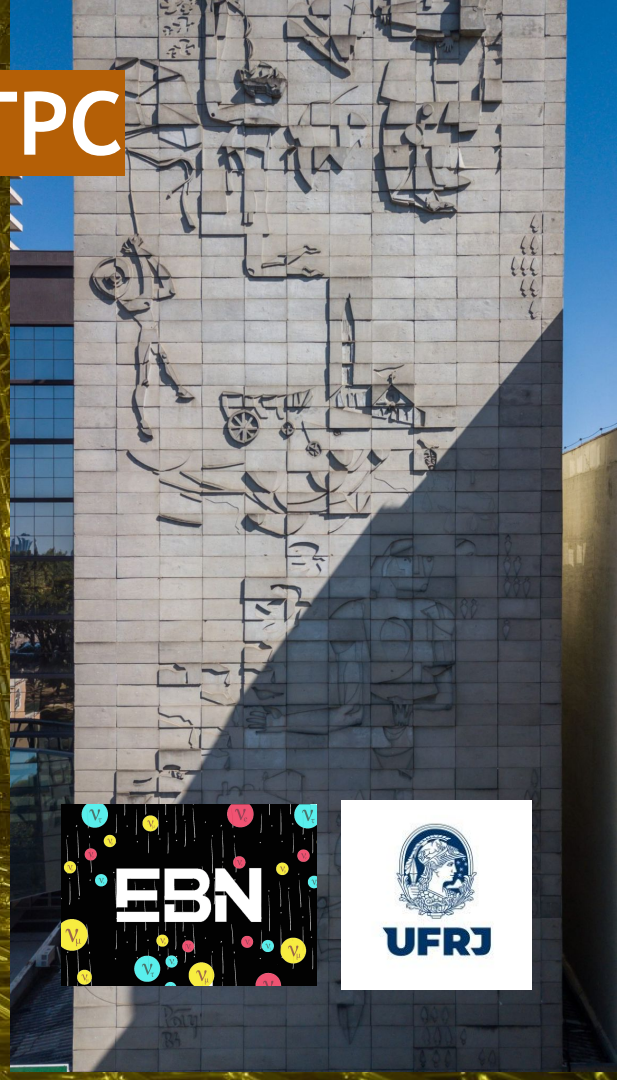
August 7th 2025

Escola Brasileira de Neutrinos - Curitiba

Leonardo Peres (UFRJ)

leoperes@if.ufrj.br

Based on: 2025 CERN LArTPC Software and Analysis Workshop
Many thanks to Andy Chappell, Dom Brailsford, Isobel Mawby, Andrew Smith-Jones and UK team!



LArTPC:
Full simulation and reconstruction chain in this workshop

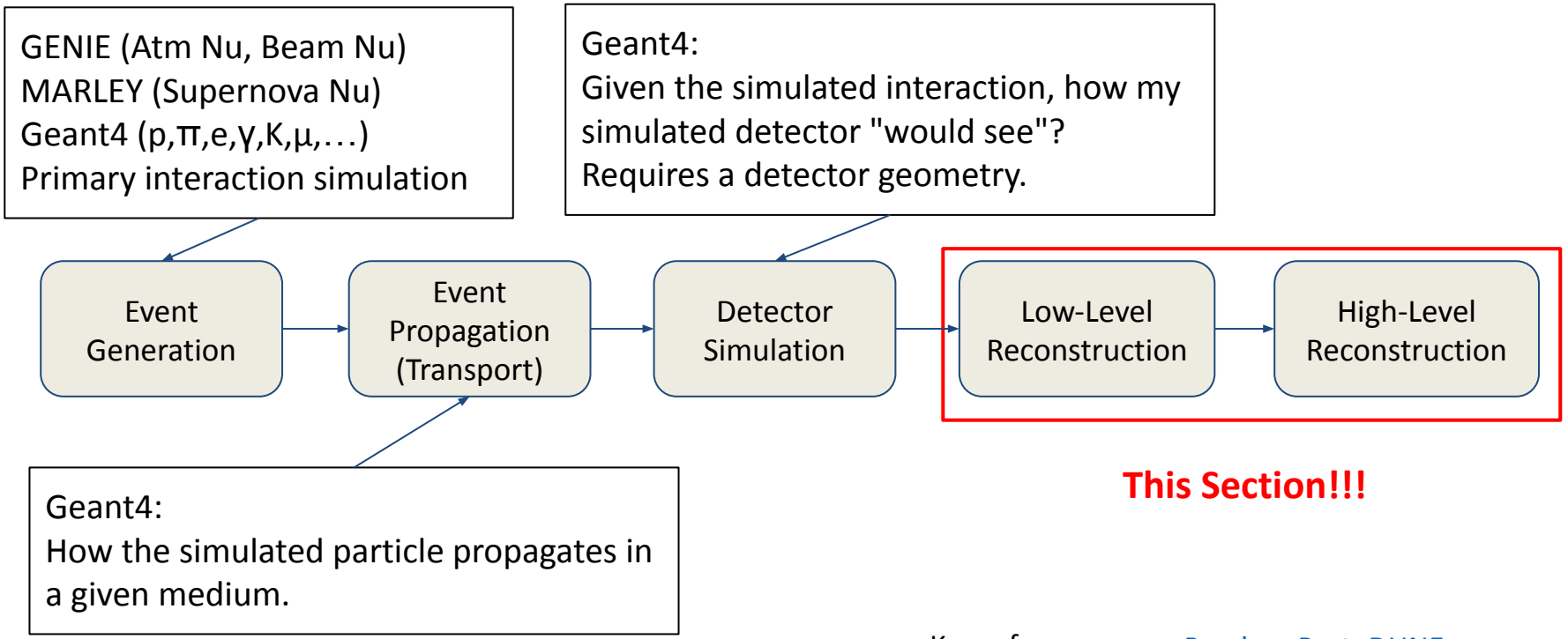
GENIE (Atm Nu, Beam Nu)
MARLEY (Supernova Nu)
Geant4 (p,π,e,γ,K,μ,…)
Primary interaction simulation

Geant4:
Given the simulated interaction, how my simulated detector "would see"?
Requires a detector geometry.

Event Generation → Event Propagation (Transport) → Detector Simulation → Low-Level Reconstruction → High-Level Reconstruction

Geant4:
How the simulated particle propagates in a given medium.

LArTPC:
Full simulation and reconstruction chain in this workshop



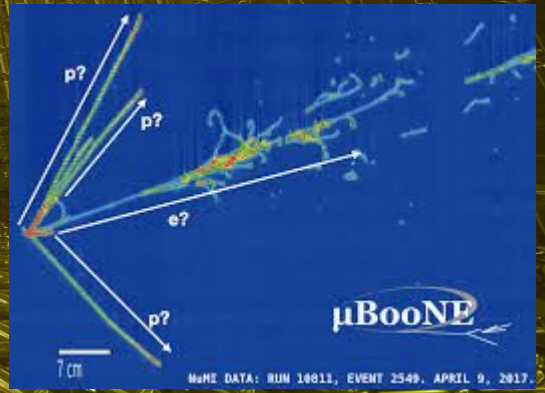
Credit: These slides build on previous LArSoft workshop slides by Andrew Smith-Jones

Key references: Pandora ProtoDUNE paper
Pandora MicroBooNE paper

Session Overview

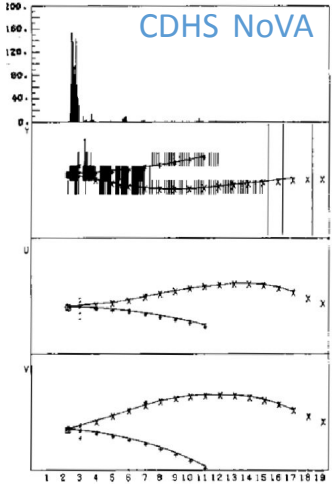
1. **Introduction to Pandora**
2. **Running the Reconstruction**
3. **Pandora Algorithms**
4. **Studying/Debugging the Reconstruction**

1. Introduction to Pandora

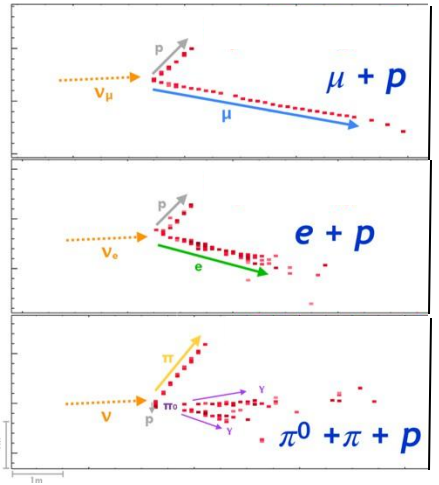


Neutrino detectors

New Frontiers in High-Energy Physics, pp227-261

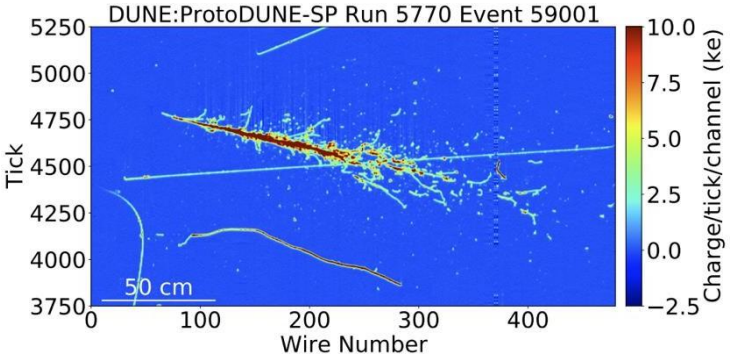


DOI: 10.5281/zenodo.1286758



B. Abi et al 2020 JINST 15 P12004

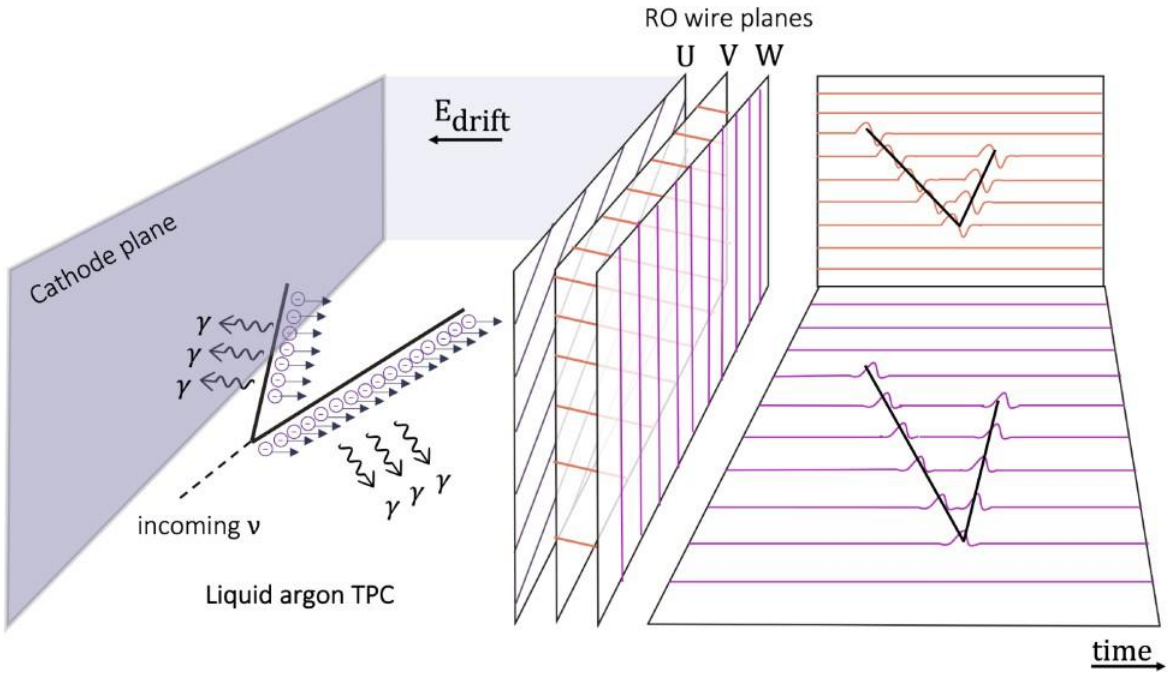
ProtoDUNE-SP



- Evolving detector technologies, with a general trend towards imaging neutrino interactions
  - Emphasis on identifying and characterizing individual visible particles
- Physics sensitivity now depends critically on both hardware and software
  - Need a sophisticated event reconstruction to harness information in the images
- Aim to reconstruct hierarchy of particles of identified types, with measured four-momenta
  - “Particle flow” reconstruction

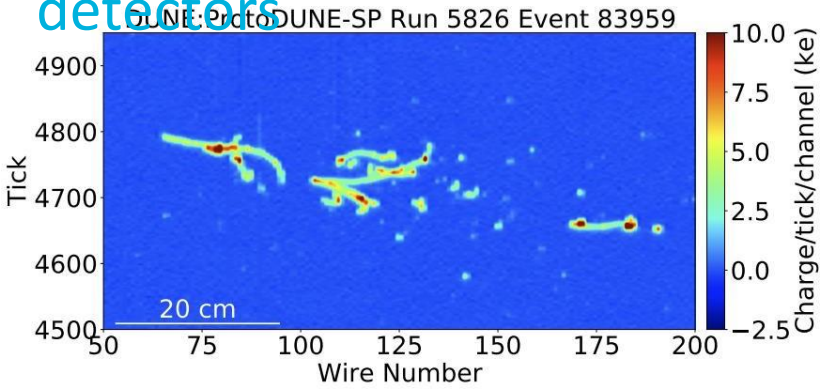
LArTPC detectors

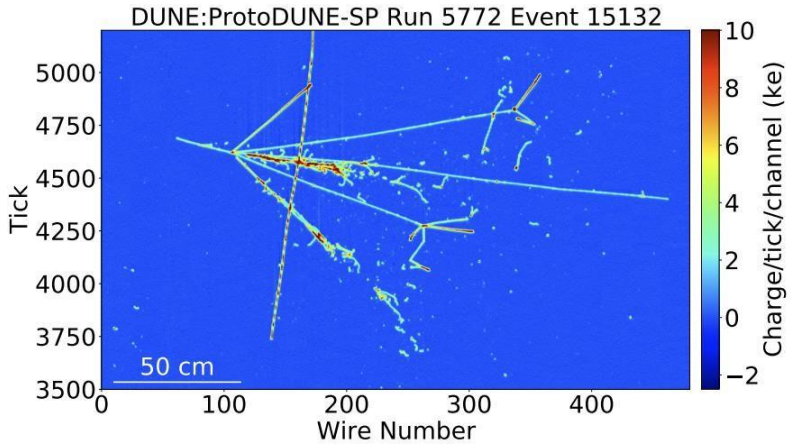
- Charged particles deposit ionization trails in liquid argon
- Ionization electrons drift in an applied electric field
- Electrons are detected by a series of readout planes (wire planes in this example)
- LArTPC detectors:
  - Past: ICARUS, ArgoNeuT, ProtoDUNE-SP/DP, MicroBooNE
  - Current: ProtoDUNE-HD, ICARUS@SBN, ICARUS
  - Coming soon: SBND, ProtoDUNE-VD



arxiv:1612.05824

LArTPC detectors



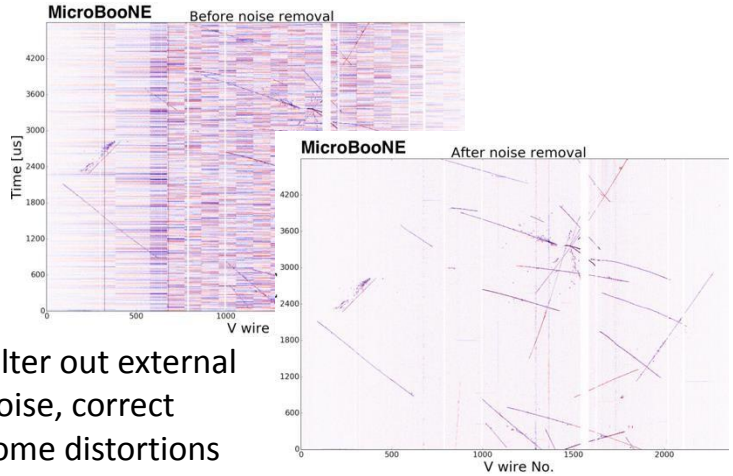


The conversion of raw LArTPC images into analysis-level quantities:

- Low-level steps:
  - Noise filtering
  - Signal processing
- Pattern recognition:
  - The bit you do by eye!
  - Turn images into sparse 2D hits
  - Assign 2D hits to clusters
  - Match features between planes
  - Output a hierarchy of 3D particles
- High-level characterisation:
  - Particle identification
  - Neutrino flavour and interaction type
  - Neutrino energy, etc…

Event reconstruction – low-level

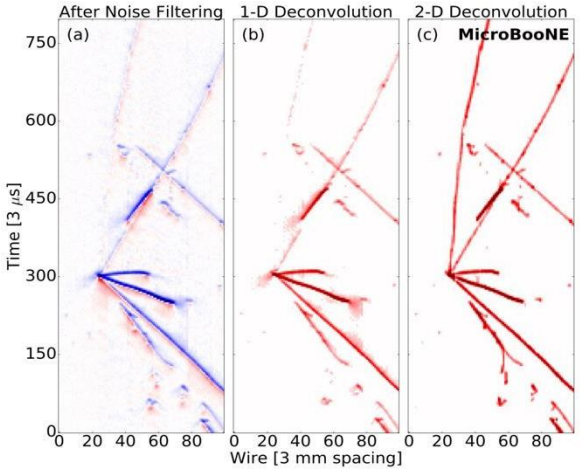
Noise filtering



Filter out external noise, correct some distortions

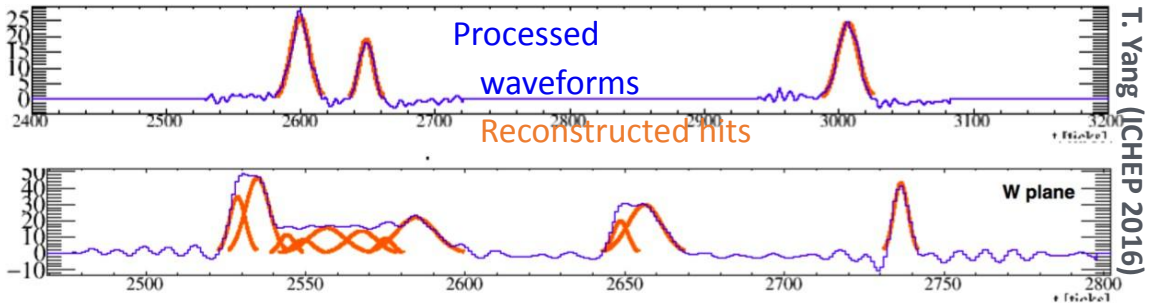
Signal processing

Convert digitized TPC waveform to number of ionization electrons passing through a wire plane at a given time (via deconvolution)



Hit finding

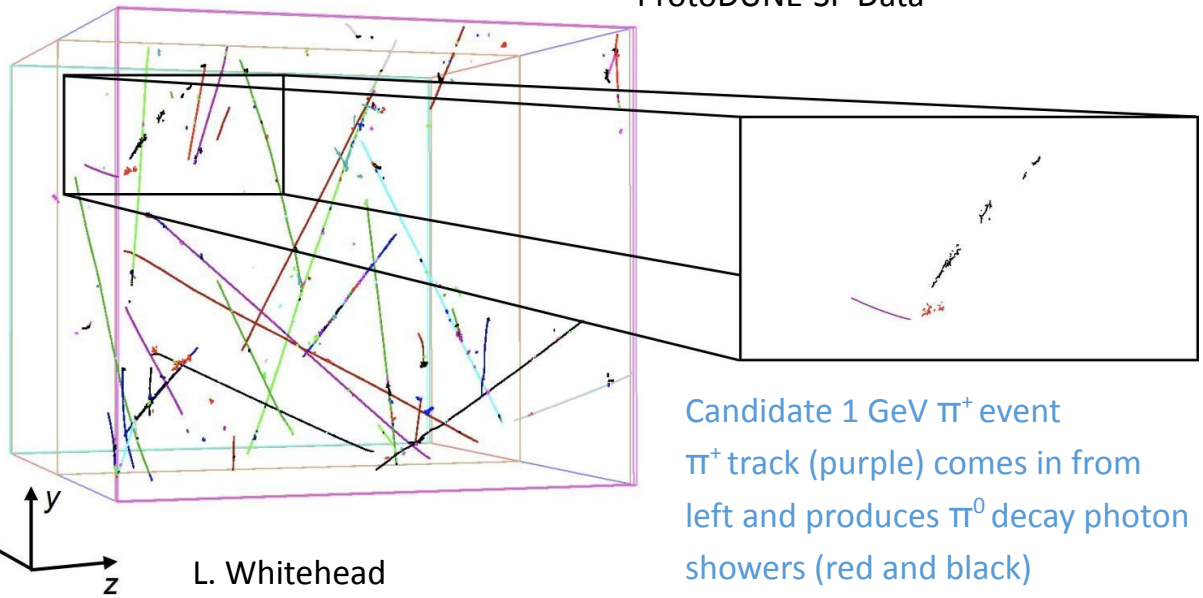
Fit clean waveform with N Gaussians, where N is number of peaks in a pulse. Each Gaussian represents a hit.



Event reconstruction – pattern recognition

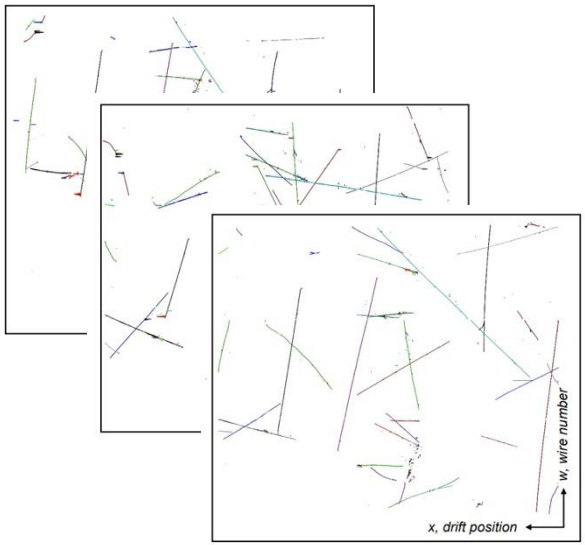
- Main aims of the pattern recognition step are to:
  - Produce 3D reconstructed particles, based on inputs of 3 x 2D images
  - Reconstruct the hierarchy of particles resulting from an interaction.

ProtoDUNE-SP Data



L. Whitehead

Candidate 1 GeV π^+ event
π^+ track (purple) comes in from left and produces π^0 decay photon showers (red and black)

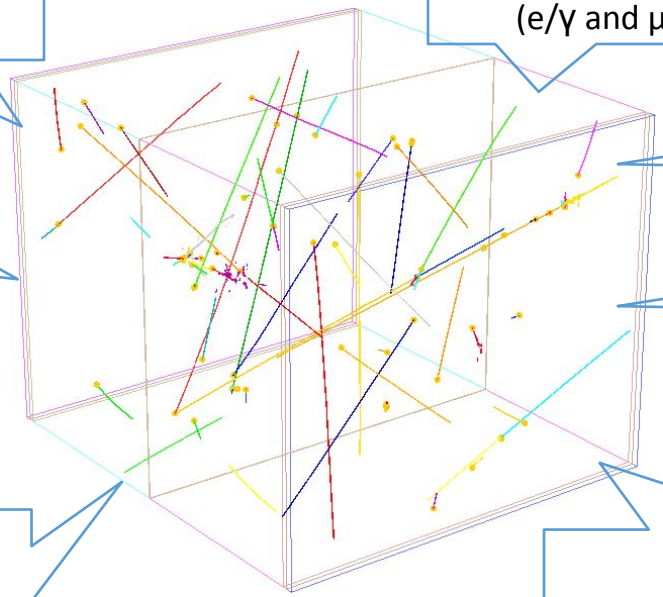


Output from hit finding

Event reconstruction – high-level characterisation

Calorimetry
dE/dx from ionisation

Particle ID
PID from dE/dx
Start/End for separation
(e/γ and μ/K/p)



π^0 Mass Reconstruction

Neutrino Interactions
Flavour ID
Background rejection
Energy estimation

Michel Electron Studies

ProtoDUNE-SP Simulation

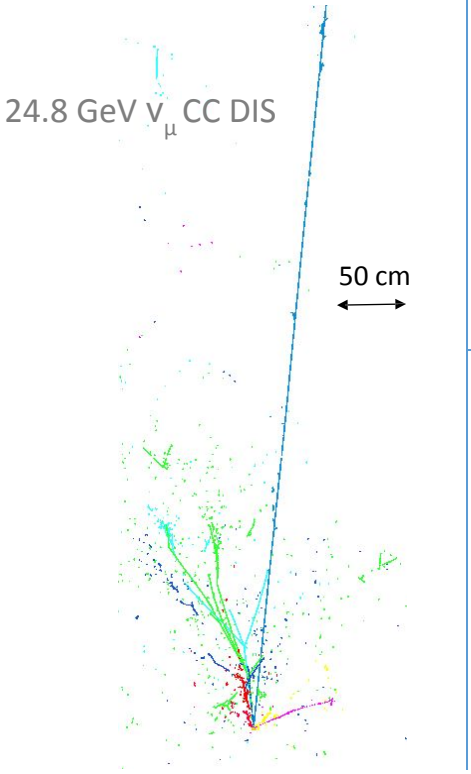
Energy Estimation
Muons
EM showers
Hadronic activity

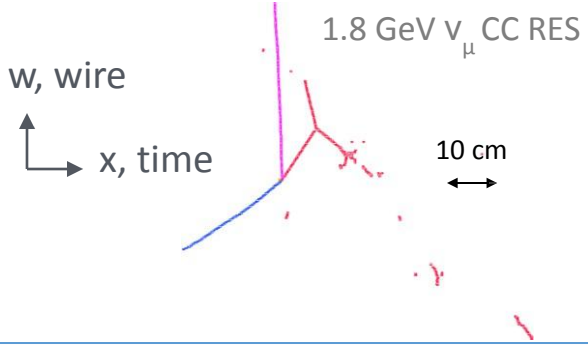
Slide: S. Green

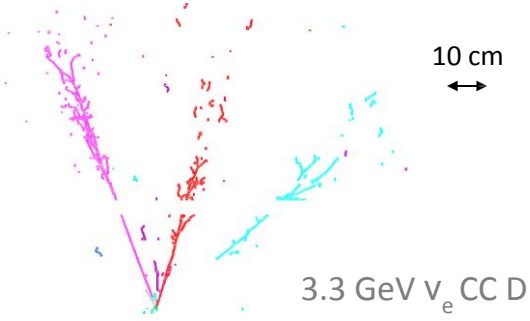
Track Fitting
Trajectory determination
Momentum from multiple Coulomb scattering for exiting tracks

Event reconstruction – focus on pattern recognition

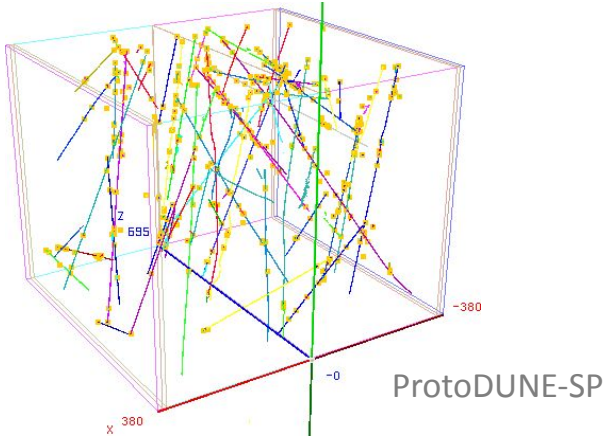
It is a significant challenge to develop automated, algorithmic LArTPC pattern recognition







- Complex, diverse topologies
- Long exposures due to long drift times (up to a few milliseconds)
- Significant cosmic-ray muon background in surface-based detectors



Pandora multi-algorithm approach

- Single clustering approach is unlikely to work for such complex topologies:
  - Mix of track-like and shower-like clusters
- Pandora uses a multi-algorithm approach:
  - Build up events gradually
  - Each step is incremental - aim not to make mistakes (undoing mistakes is hard…)
  - Deploy more sophisticated algorithms as picture of event develops
  - Build physics and detector knowledge into algorithms

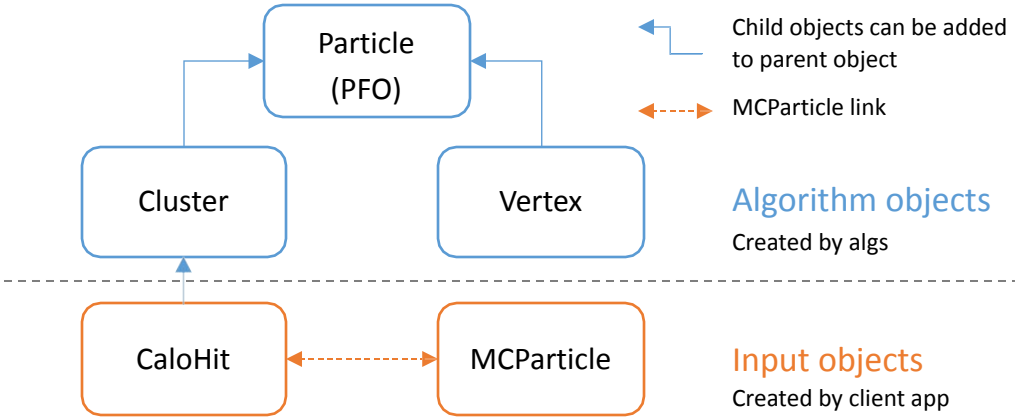
Pandora multi-algorithm approach

- Algorithms contain high-level logic and concentrate on the important bits
  - Physics and pattern recognition ideas
- Pandora software development kit (SDK) supports algorithms
  - Functions to access objects
  - Make new objects
  - Modify existing objects, etc.

Algorithm 1 Cluster creation pseudocode. The logic determining when to create new Clusters and when to extend existing Clusters will vary between algorithms.

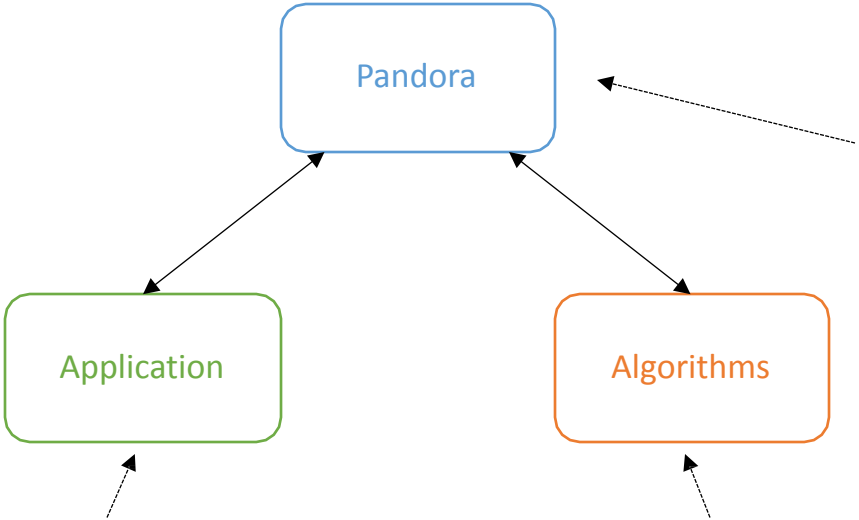
```
1:  procedure CLUSTER CREATION
2:      Create temporary Cluster list
3:      Get current CaloHit list
4:      for all CaloHits do
5:          if CaloHit available then
6:              for all newly-created Clusters do
7:                  Find best host Cluster
8:              if Suitable host Cluster found then
9:                  Add CaloHit to host Cluster
10:             else
11:                 Add CaloHit to a new Cluster
12:     Save new Clusters in a named list
```

Example algorithm structure



Event Data Model

Pandora in LArSoft



pandora (framework, visualization)

Re-usable libraries to support multi-alg approach

Hosted on PandoraPFA GitHub
pre-built as external package by LArSoft

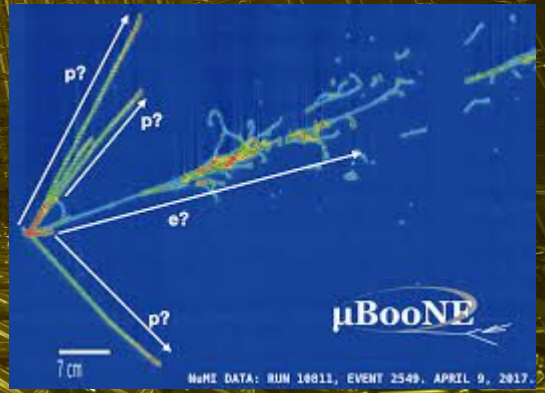
Producer module, provides translation LArSoft⟷Pandora

Hosted via LArSoft GitHub, built with mrb

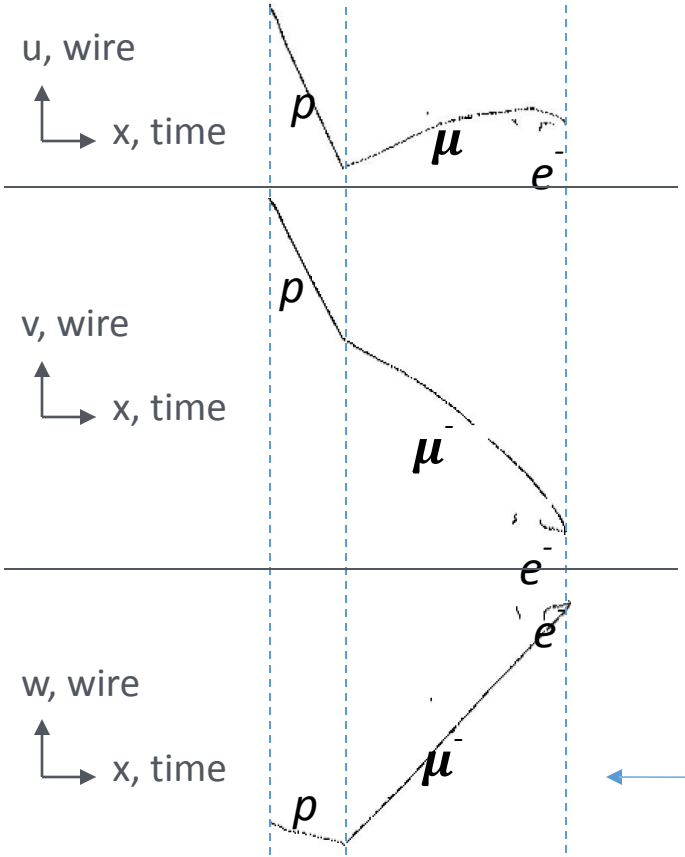
100+ algorithms and tools that control the patrec

Hosted via LArSoft and PandoraPFA GitHub, can build with mrb

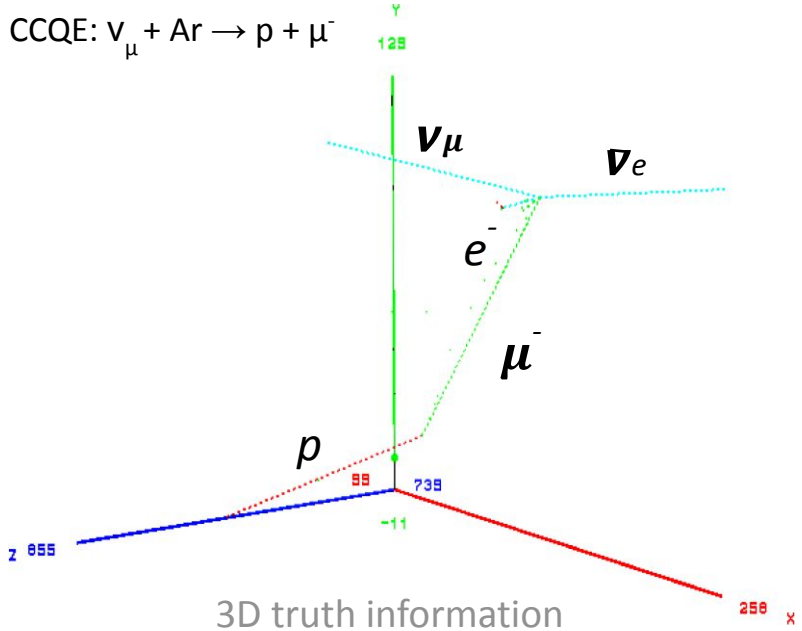
2. Running the Reconstruction



Inputs to Pandora



CCQE: $\nu_\mu + Ar \rightarrow p + \mu^-$



3D truth information

Three 2D representations with common x coordinate, derived from drift time

LArSoft workflows

- In this session we'll be working from detsim files
  - You'll see a little more detail on workflows in the next exercise
  - Your detsim files have gone through μ+p particle gun, G4 propagation and detector simulation, but before we run Pandora, we also need to run "reco1", which includes things like the hit-finding and disambiguation
- Where does Pandora run?
  - Pandora typically runs as part of the "reco2" workflow step
  - It is followed by various high-level reconstruction steps that use its outputs
  - Typically, these steps will produce reco2 output files on which analyzers will run

We will now try running the larpandora module in LArSoft

Goals

- Main goal 1 - Find and get to grips with the DUNE FD1-HD reconstruction FHiCL files
  - Find the standard_reco1_dune10kt_1x2x6.fcl and standard_reco2_dune10kt_1x2x6.fcl configuration files Look at the different reconstruction steps that we run
  - Understand what each of them do
- Main goal 2 - Run the reconstruction
  - Run the reconstruction on the files we simulated yesterday This includes running Pandora
  - Dump out the new output products to confirm we produced what we wanted

Before we get started…

- Later today we'll be running the event display
- Follow the steps from yesterday to get everything set up. To check, make sure MRB_TOP points to where you expect. Let us know if you have any issues here

```
$ echo $MRB_TOP # This should print the path to your development
area
```

Main Goal 1

Understanding the DUNE FD1-HD reconstruction FHiCL files

DUNE FD1-HD reconstruction FHiCL files

- The full reconstruction is split into two fcls:
  1) standard_reco1_dune10kt_1x2x6.fcl
  2) standard_reco2_dune10kt_1x2x6.fcl

  which are run in succession

- reco1 generally corresponds to the ‘lower-level’ reconstruction e.g. hit formation, disambiguation etc…
- reco2 generally corresponds to the ‘higher-level’ reconstruction e.g. particle creation

- Let’s take a closer look at the fcls to see what these stages in more detail

reco1

- Open standard_reco1_dune10kt_1x2x6.fcl in your favourite text editor

```
$ vim $DUNESW_DIR/fcl/standard_reco1_dune10kt_1x2x6.fcl

or

$ code $DUNESW_DIR/fcl/standard_reco1_dune10kt_1x2x6.
```

- Find the trigger_paths: [...] block
  - Can't find it?
  - This fcl ***#include "standard_reco1_dune10kt.fcl"*** so let's check in there.
  - Find standard_reco1_dune10kt.fcl (using the same directory as above) and take a look inside

```
#include "standard_reco1_dune10kt.fcl"

process_name: Reco1

services:
{
    @table::services
    @table::dunefd_1x2x6_reco_services
}

physics.producers:
{
    @table::physics.producers
    wclsmcnfsp: @local::dune10kt_1x2x6_mc_nfsp
}
```

standard_reco1_dune10kt_1x2x6.fcl

reco1

- Open standard_reco1_dune10kt.fcl in your favourite text editor

```
$ vim $DUNESW_DIR/fcl/standard_reco1_dune10kt.fcl
```

- Find the trigger_paths: [...] block
  - Still nothing …
  - This line is interesting. We should remember this for a little bit later
  - Let's check the next include

```
# Dom Brailsford
# March 2022
#
# Runs the low level reconstruction for DUNE FD

#include "standard_reco_dune10kt.fcl"
#
process_name: Reco1

services.TFileService.fileName: "reco1_hist.root"

source.inputCommands: ["keep *_*_*_*", "drop *_*_*_Reco1" ]

physics.reco: [ @sequence::dunefd_horizdrift_workflow_reco1 ]

outputs.out1.fileName: "%ifb_reco1.root"
outputs.out1.dataTier: "hit-reconstructed"
```

standard_reco1_dune10kt.fcl

reco1

- Open standard_reco_dune10kt.fcl in your favourite text editor

```
$ vim $DUNESW_DIR/fcl/standard_reco_dune10kt.fcl
```

- Find the trigger_paths: [...] block
  - Found it!
  - This loads a sequence called ***reco***
  - The previous fcl override this sequence
    - ***dunefd_horizdrift_workflow_reco1***
    - Now we need to check how this sequence is defined. Let's open another include: workflow_reco_dune10kt.fcl

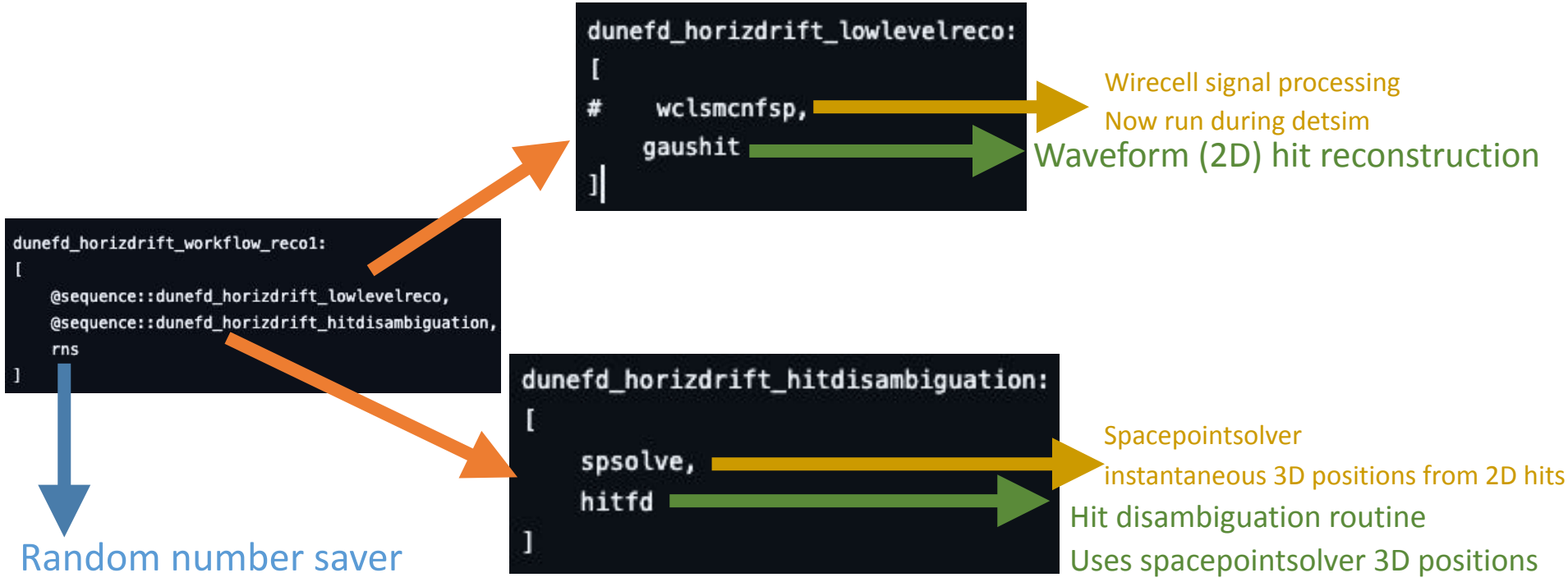
```
#define the output stream, there could be more than o
stream1:  [ out1 ]

#trigger_paths is a keyword and contains the paths th
#ie filters and producers
reco: [ @sequence::dunefd_horizdrift_workflow_reco ]
trigger_paths: [ reco ]

#end_paths is a keyword and contains the paths that d
#ie analyzers and output streams.  these all run simu
end_paths:     [stream1]
}
```

(snippet of)
standard_reco_dune10kt.fcl

reco1

- Open workflow_reco_dune10kt.fcl in your favourite text editor

```
$ vim $DUNESW_DIR/fcl/workflow_reco_dune10kt.fcl
```



reco2

- Now we'll follow the same procedure for reco2: open standard_reco2_dune10kt_1x2x6.fcl

```
$ less
$DUNESW_DIR/fcl/standard_reco2_dune10kt_1x2x6.fcl
$ less $DUNESW_DIR/fcl/standard_reco2_dune10kt.fcl
$ less $DUNESW_DIR/fcl/standard_reco_dune10kt.fcl
```

includes

- Find the physics.reco path in standard_reco2_dune10kt.fcl
  - Q: which producers are we going to run?
  - A: It's the ones in the dunefd_horizdrift_workflow_reco2 path
    - The sequence contents are found higher up in the fcl file

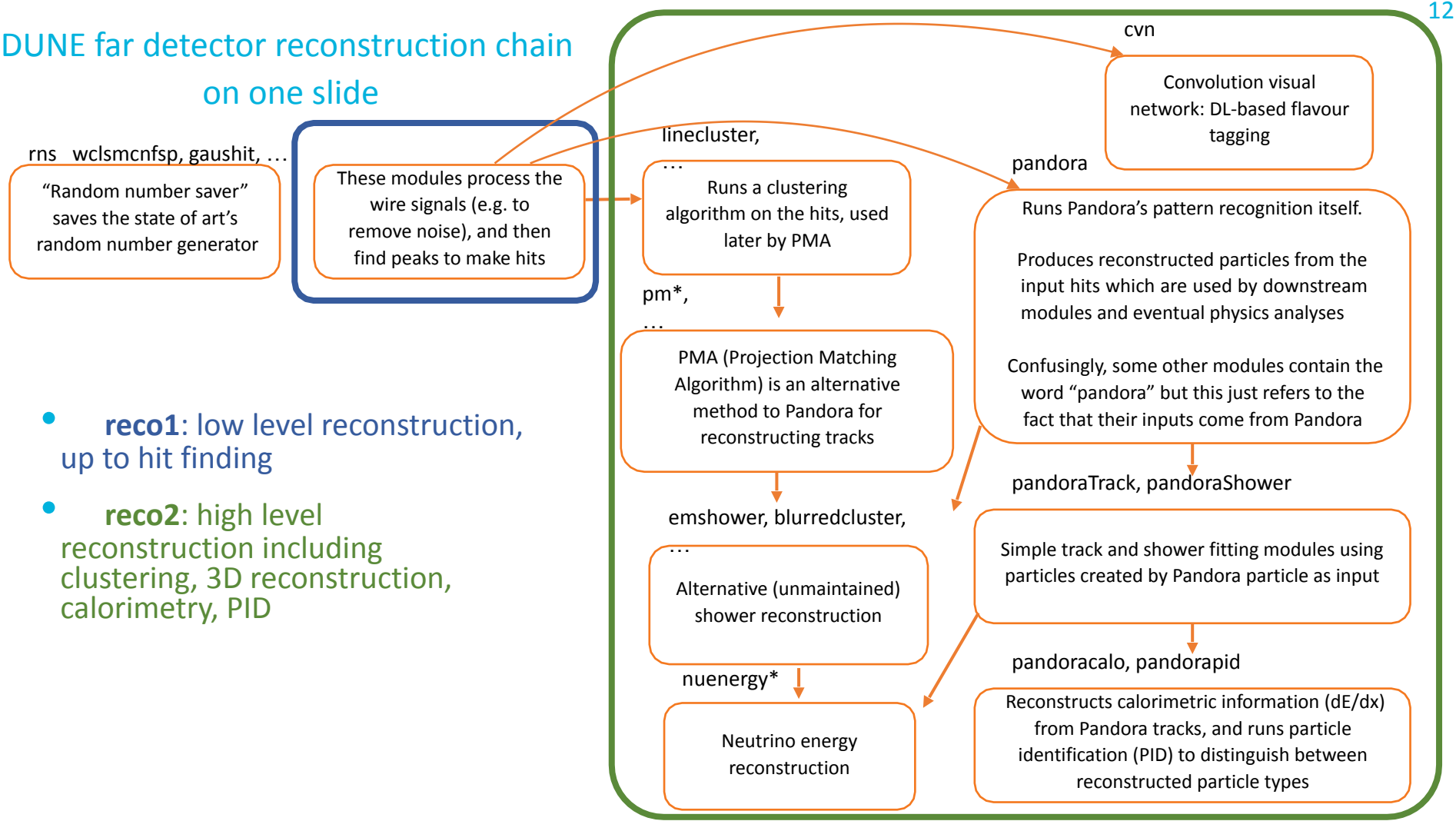
First time using less? Use the ↑ / ↓ arrow keys to navigate the file, and press q to quit

```
dunefd_horizdrift_workflow_reco2: [
    @sequence::dunefd_horizdrift_2dclustering,
    @sequence::dunefd_horizdrift_pandora,
    @sequence::dunefd_horizdrift_pmtrack,
    @sequence::dunefd_horizdrift_pmtrack_…,
    @sequence::dunefd_horizdrift_pmtrack_…,
    @sequence::dunefd_horizdrift_pmtrack_…,
    @sequence::dunefd_horizdrift_cvn,
    @sequence::dunefd_horizdrift_nuenergy,
    @sequence::dunefd_horizdrift_pd_reco1,
    @sequence::dunefd_horizdrift_pd_reco2,
    rns
]
```

```
dunefd_horizdrift_pandora:
   [ pandora,
   pandoraTrack,
   pandoraShowe,
   pandoracalo,
   pandorapid ]
```

- Next is a single-slide overview of all the reco steps - not nearly enough to do them justice - but today we will mainly be focusing on pandora

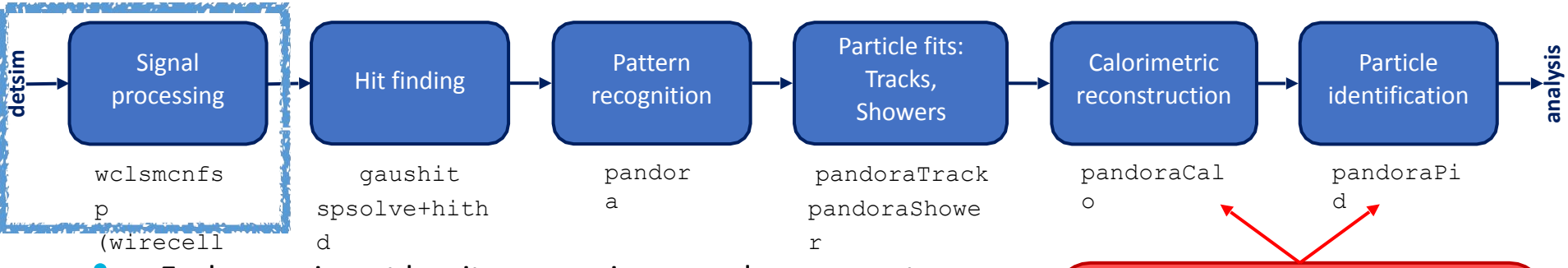
DUNE far detector reconstruction chain on one slide



- **reco1**: low level reconstruction, up to hit finding
- **reco2**: high level reconstruction including clustering, 3D reconstruction, calorimetry, PID

A cumulative reconstruction output

- There's a lot happening in the reconstruction stage
- To demonstrate how these modules accumulate a reconstruction output, let's focus on the Pandora 'workflow' i.e. ignoring the optical reconstruction



- Each experiment has its own unique needs, so expect to see some differences in the reconstruction chain if you work on MicroBooNE, ProtoDUNE, DUNE, etc.
- Similarly, Pandora is configured differently for different experiments, and different use cases (e.g. atmospheric vs beam neutrino interactions)
- Next, let's see how Pandora is configured for DUNE FD1-HD

Historically, modules using Pandora outputs frequently have Pandora in the name.

Most of these modules aren't part of Pandora and aren't maintained by the Pandora development team.

Pandora's configuration

- Practically all LArSoft modules have a configuration defined by parameters e.g. say that we had a producer module that added a number to our root file – it would likely have a parameter which defines the number that is added, and that parameter is set in our fcl files
- fcl files often *#include* many other fcl files, and it can be unclear where our parameters are set
- fhicl-dump answers this question, and follows all *#includes* to get the bottom-line configuration
- We can pipe (|) its output to less and search for a producer to learn more:

```
$ fhicl-dump standard_reco2_dune10kt_1x2x6.fcl | less -p "pandora:"
```

Less's –p option allows us to jump straight to the part of the file that we are interested in. In less, use the ↑/↓ keys to move up and down, and 'q' to exist.

```
{
   ConfigFile: "PandoraSettings_Master_DUNEFD.xml"
   EnableLineGaps: true
   EnableMCParticles: true
   EnableProduction: true
   GeantModuleLabel: "largeant"
   HitFinderModuleLabel: "hitfd"
   PrintOverallRecoStatus: false
   ShouldPerformSliceId: false
   ShouldRunAllHitsCosmicReco: false
   ShouldRunCosmicHitRemoval: false
   ShouldRunCosmicRecoOption: false
   ShouldRunNeutrinoRecoOption: true
   ShouldRunSlicing: false
   ShouldRunStitching: false
   SimChannelModuleLabel: "tpcrawdecoder:simpleSC"
   UseGlobalCoordinates: true
   UseHitWidths: true
}  module_type: "StandardPandora"
```

The settings file that contains the list of algorithms that Pandora will run

The producer module that created the hits that we are going to feed into Pandora

The steering parameters that tell Pandora which of its high-level reconstruction steps it should execute

The type of the LArSoft module to use

Pandora Config

(For reference)

- Firstly, what steering parameters do we have and what do they mean?

```
ShouldRunAllHitsCosmicRec
o
ShouldRunStitchin
g
ShouldRunCosmicHitRemova
l
ShouldRunSlicin
g
ShouldRunCosmicRecoOptio
n
ShouldRunNeutrinoRecoOptio
n
ShouldPerformSliceI
d
```

should we initially run a fast and simplified version of the reconstruction? and do we need to 'stitch' this reconstruction output between different detector chambers?

should we try to remove 'obvious' cosmic rays from this output e.g. tracks that both enter and leave the detector?

do we expect multiple cosmic rays? or multiple neutrinos? or both? And therefore, do we need to group (or slice) the input hits such that each individual interaction exists within it's own slice?

do we expect cosmic-rays? and so should we run the cosmic algorithm chain over these slices?

do we expect neutrinos? and so should we run the neutrino algorithm chain over these slices?

do we need to determine whether a slice contains a cosmic ray or neutrino interaction?

Main Goal 2

Running the reconstruction

Running the reconstruction

- We are now poised to run the reconstruction! Make a directory to work in, and run it:

```
$ mkdir -p $MRB_TOP/reco/work
$ cd $MRB_TOP/reco/work
$ lar -c standard_reco1_dune10kt_1x2x6.fcl -n -1 -s /path/to/my/detsim/file.root -o reco1_events.root
$ lar -c standard_reco2_dune10kt_1x2x6.fcl -n -1 -s reco1_events.root -o reco2_events.root

Can also run on pre-made gen+g4+detsim files in:
/home/leonardo/arqsEBN/prod_1mu1p_novar_10evts_g4_detsim.root
```

The -n -1 option means run over all events in the input file

```
TimeTracker printout (sec)                                  Min          Avg          Max          Median       RMS          nEvts
Full event                                                  14.4235      16.7021      22.9312      15.3978      3.17841      5
source:RootInput(read)                                      0.000452972  0.000635409  0.000780154  0.000701023  0.000128122  5
reco:linecluster:LineCluster                                0.0167877    0.0195162    0.0252377    0.0173858    0.00333468   5
reco:trajcluster:TrajCluster                                0.0550071    0.0701446    0.0917665    0.0652082    0.0124367    5
reco:pandora:StandardPandora                                0.748147     1.25079      2.75524      0.862925     0.760793     5
reco:pandoraTrack:LArPandoraTrackCreation                   0.00308148   0.00406956   0.00777318   0.00316728   0.00185255   5
reco:pandoraShower:LArPandoraModularShowerCreation          0.0162824    0.0203825    0.0253648    0.0192886    0.00303779   5
reco:pandoracalo:Calorimetry                                0.00431279   0.00504067   0.00715086   0.00455915   0.00106172   5
reco:pandorapid:Chi2ParticleID                              9.6464e-05   0.000435184  0.00172821   0.000110911  0.000646606  5
reco:trkshowersplit:TrackShowerHits                         0.00274607   0.00313384   0.0042873    0.00287603   0.000582228  5
reco:pmtrack:PMAlgTrackMaker                                0.54331      1.18013      2.20622      1.2577       0.603529     5
reco:pmtrackcalo:Calorimetry                                0.000723637  0.00180062   0.00485471   0.00117207   0.00155474   5
…
```

We can check to see that everything we expected has been executed, and see how long each took

So… what's new?

- Run eventdump.fcl on the .root file to see the new collections we've just made

```
$ lar -c eventdump.fcl -s reco2_events.root -n
1
```

```
PROCESS NAME | MODULE LABEL..... | PRODUCT INSTANCE NAME..... | DATA PRODUCT TYPE..... | .SIZE
G4.......... | largeant......... | ..... | sim::ParticleAncestryMap..... | .....-
G4.......... | largeant......... | LArG4DetectorServicevolTPCPlaneUOuter. | std::vector<sim::SimEnergyDeposit>..... | .....0
G4.......... | largeant......... | LArG4DetectorServicevolTPCActiveOuter. | std::vector<sim::SimEnergyDeposit>..... | .....0
detsim...... | opdigi........... | ..... | std::vector<sim::OpDetDivRec>..... | ...383
detsim...... | tpcrawdecoder..... | simpleSC..... | std::vector<sim::SimChannel>..... | 384000
detsim...... | tpcrawdecoder..... | wiener..... | std::vector<recob::Wire>..... | 384000
detsim...... | rns.............. | ..... | std::vector<art::RNGsnapshot>..... | .....1
Reco1....... | spsolve.......... | ..... | std::vector<recob::SpacePoint>..... | ...159
Reco1....... | hitfd............ | ..... | std::vector<recob::Hit>..... | ...298
Reco1....... | spsolve.......... | ..... | std::vector<recob::PointCharge>..... | ...159
Reco1....... | TriggerResults... | ..... | art::TriggerResults..... | .....1
...
Reco2....... | pandoraShower.... | ..... | art::Assns<recob::Shower,recob::PCAxis,void>..... | .....3
Reco2....... | pandoraShower.... | ..... | art::Assns<recob::Shower,recob::SpacePoint,void>..... | ...252
Reco2....... | pandora.......... | ..... | art::Assns<recob::Slice,recob::Hit,void>..... | ...298
Reco2....... | pandora.......... | ..... | std::vector<recob::PFParticle>..... | .....4
...
```

these are the existing data products from the previous steps

these are the new data products that we produced in reco1

… and in reco2

You'll see a much longer list! I've cut out a lot output so that it all fits on the slide

So… what's new?

- Ru… …ew collections we've just made


```
$ la
1

PROCESS NAME | …                                                                    | DAT…
G4.......... | …artic…
G4.......... | …ector…
G4.......... | …ector…
detsim...... | …ector…
detsim...... | …ector<sim::SimChannel>.......................... | 384000
detsim...... | …ector<recob::Wire>.............................. | 384000
detsim...... | …ector<art::RNGsnapshot>.......................... |      1
Reco1....... | …ector<recob::SpacePoint>........................ |    159
Reco1....... | …ector<recob::Hit>............................... |    298
Reco1....... | …ector<recob::PointCharge>....................... |    159
Reco1....... | …riggerResults................................... |      1
Reco2....... | …ssns<recob::Shower,recob::PCAxis,void>.......... |      3
Reco2....... | …ssns<recob::Shower,recob::SpacePoint,void>...... |    252
Reco2....... | …ssns<recob::Slice,recob::Hit,void>.............. |    298
Reco2....... | …ector<recob::PFParticle>........................ |      4
```

You're now able to coffee break !!!!!

…he
…ta
…om
…us
steps
these are the new data products that we produced in reco1

… and in reco2

…u'll see a much longer list!
I've cut out a lot output so that it all fits on the slide

Additional information

Configuring Pandora steps (For reference)

- Pandora's full reconstruction chain is designed to handle neutrino interactions in dense cosmic environments. As you will hear later, there are two main algorithm chains optimised for cosmic rays, and neutrinos respectively
- For DUNE-FD - we normally want to run the cosmic and neutrino reconstruction completely independently
- For cosmic events, we only need to run the cosmic algorithm chain. For neutrino events, we only need to run the neutrino algorithm chain. We can configure Pandora to run one, many or all of the steps in its full reconstruction chain by modifying the FHiCL steering parameters
- Make a new directory to work in for this session, and add a new FHiCL file with the following lines, then save and close the file:

```
$ mkdir -p $MRB_TOP/reco/config
$ cd $MRB_TOP/reco/config # Put your new .fcl file here
$ vim my_reco.fcl
```

Please use your favourite text editor, here we use vim. If you accidentally opened vim and want to close it type Esc, :qa, Return ↵

```
#include "standard_reco2_dune10kt_1x2x6.fcl"

physics.producers.pandora.ShouldRunAllHitsCosmicReco: false
physics.producers.pandora.ShouldRunStitching: false
physics.producers.pandora.ShouldRunCosmicHitRemoval: false
physics.producers.pandora.ShouldRunSlicing: false
physics.producers.pandora.ShouldRunCosmicRecoOption: false
physics.producers.pandora.ShouldRunNeutrinoRecoOption: true
physics.producers.pandora.ShouldPerformSliceId: false
```

Include the standard configuration

Example:
Only run the neutrino algorithm chain

Pointing to a new configuration (For reference)

- We want to make sure that LArSoft will know where to look for our new FHiCL file, to do this we add it to the FHICL_FILE_PATH environment variable. Start by printing it to the terminal:

```
$ echo
$FHICL_FILE_PATH
```

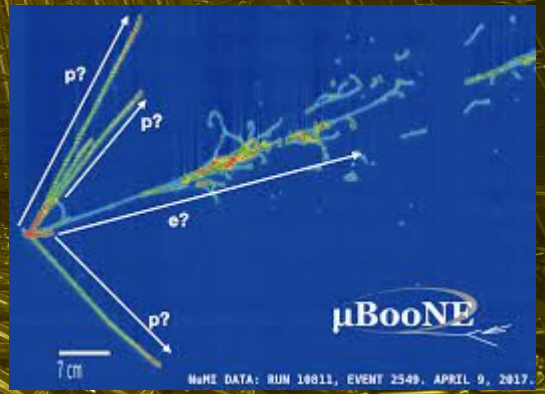
- You will see many, many directories, all separated by a ':'. To add our reco/config folder to this list run the following command:

```
$ export
FHICL_FILE_PATH=$MRB_TOP/reco/config:$FHICL_FILE_PATH
```

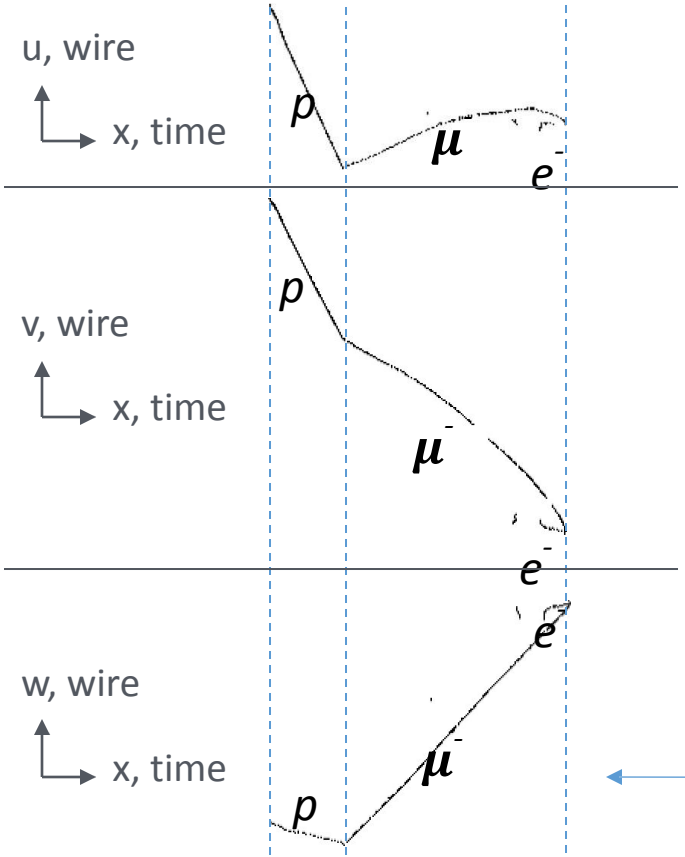
- Echo the FHICL_FILE_PATH again to check that everything worked (it should be the first in the list)
- Now run fhicl-dump again to make sure our new configuration file is set up as we want

```
$ fhicl-dump my_reco.fcl | less -p
"pandora:"
```

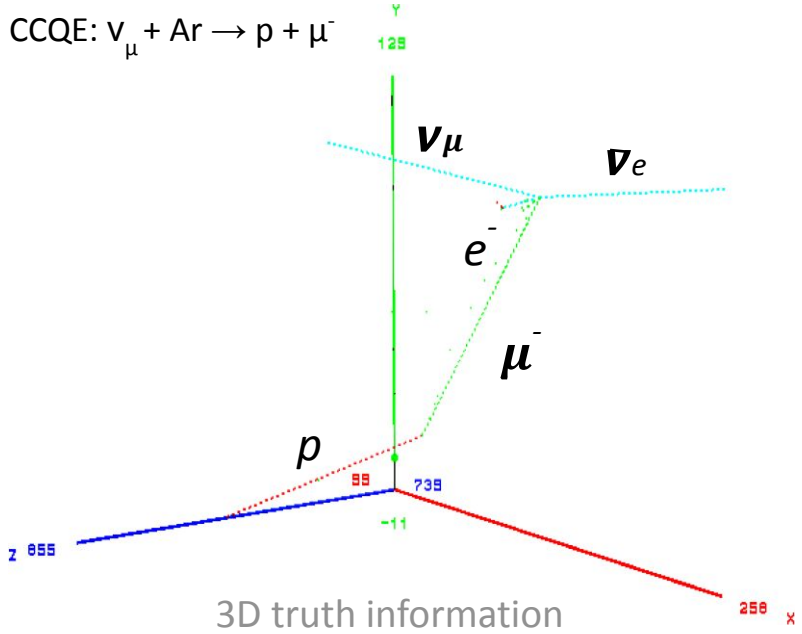
3. Pandora's Algorithms



Inputs to Pandora



CCQE: $\nu_\mu + Ar \rightarrow p + \mu^-$

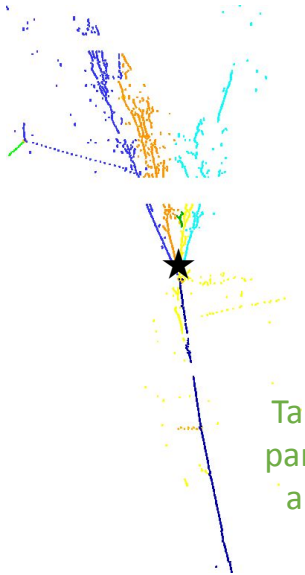


3D truth information

Three 2D representations with common x coordinate, derived from drift time

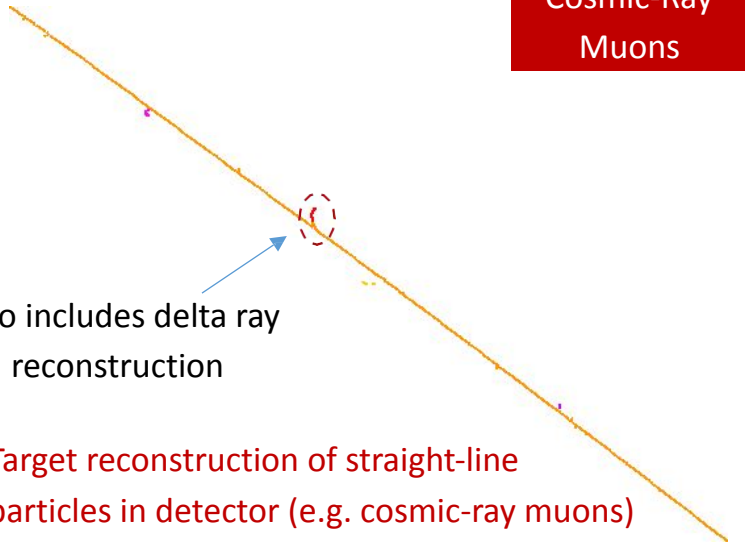
Consolidated reconstruction

- We use a multi-algorithm approach to create two algorithm chains:
- Consolidated reconstruction uses these chains to guide reconstruction for all use cases:
- Cosmic rays ✓, Multiple drift volumes ✓, Arbitrary wire angles ✓ , 2 or 3 wire planes ✓



Neutrino (or TestBeam)

Target reconstruction of particles emerging from an identified vertex ★

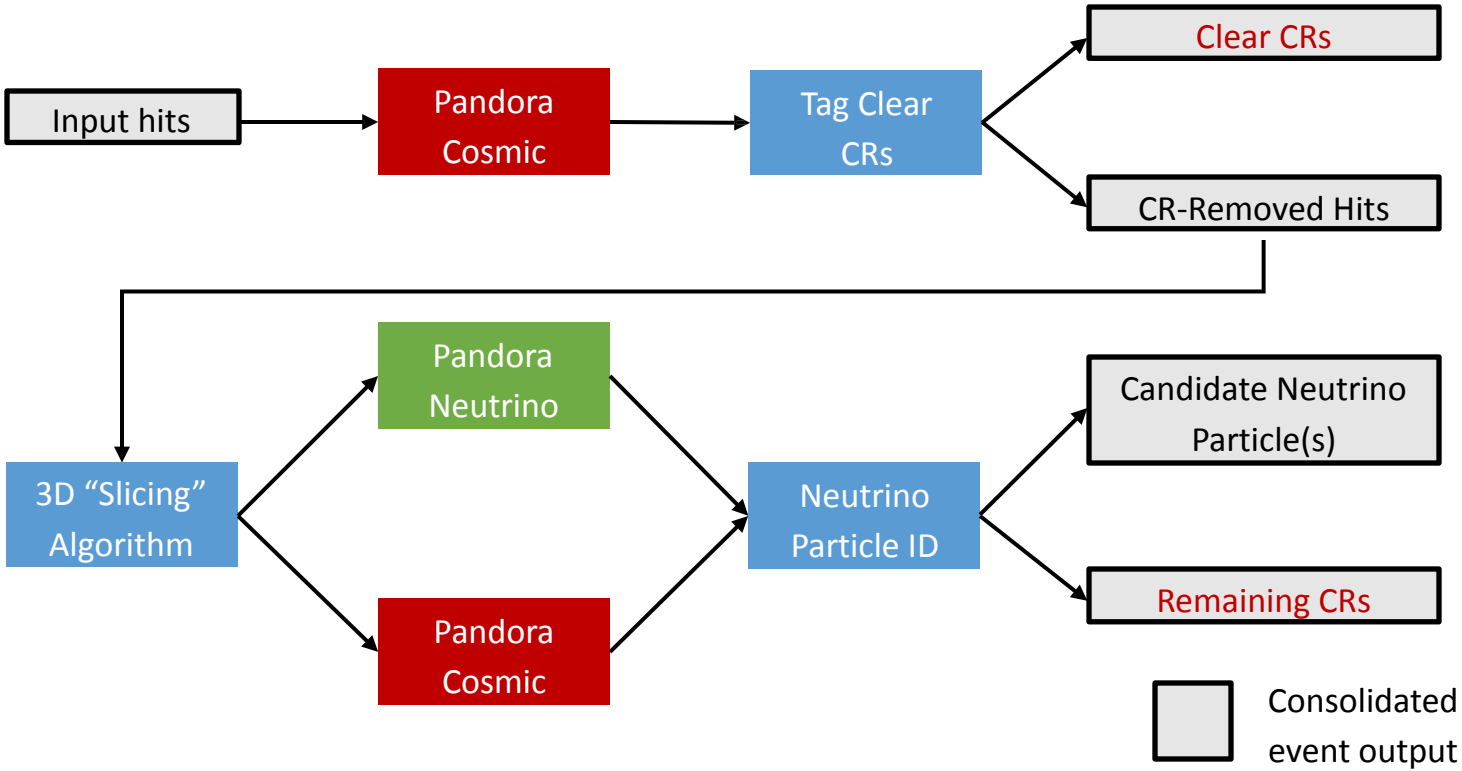


Cosmic-Ray Muons

Also includes delta ray reconstruction

Target reconstruction of straight-line particles in detector (e.g. cosmic-ray muons)

Consolidated reconstruction

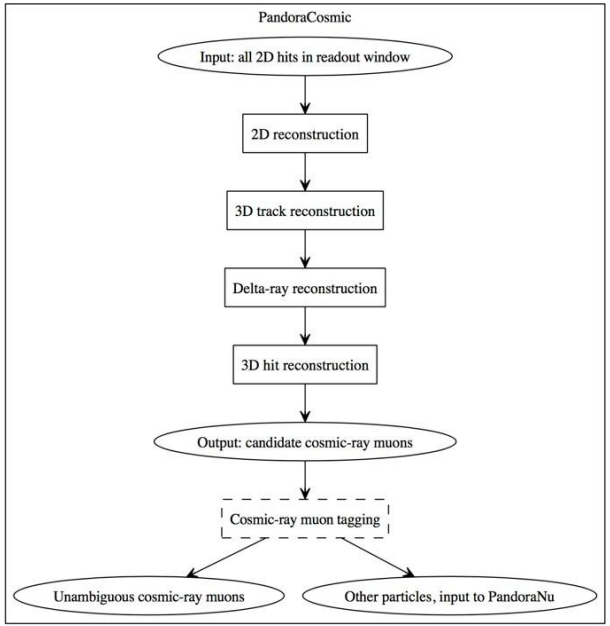


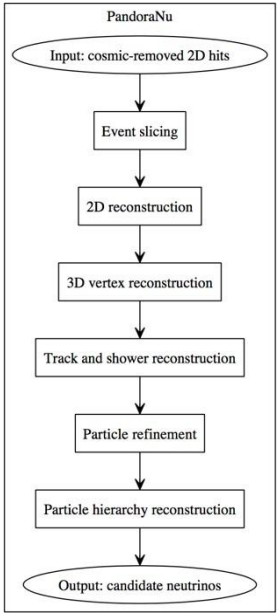
Consolidated reconstruction - Algorithm chains

- Two Pandora algorithm chains created for LArTPC use, with many algs in common:
  - PandoraCosmic: strongly track-oriented; showers assumed to be delta rays*, added as daughters of primary muons; muon vertices at track high-y coordinate.
  - PandoraNu: finds neutrino interaction vertex and protects all particles emerging from vertex position. Careful treatment to address track/shower tensions.

Initially use a two-pass approach: Input to PandoraNu excludes hits from unambiguous cosmic rays.

*delta rays: secondary electrons ejected from atoms due to interactions with other charged particles.

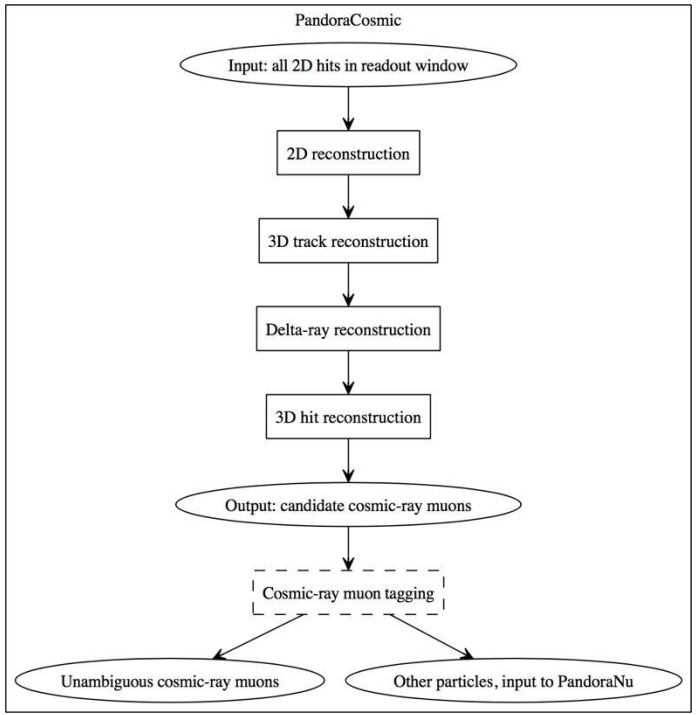


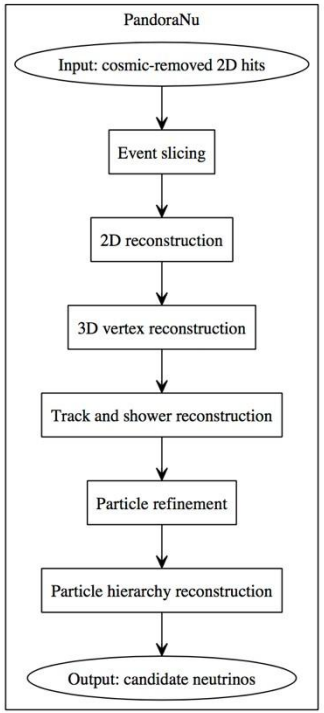


Other particles, input to PandoraNu

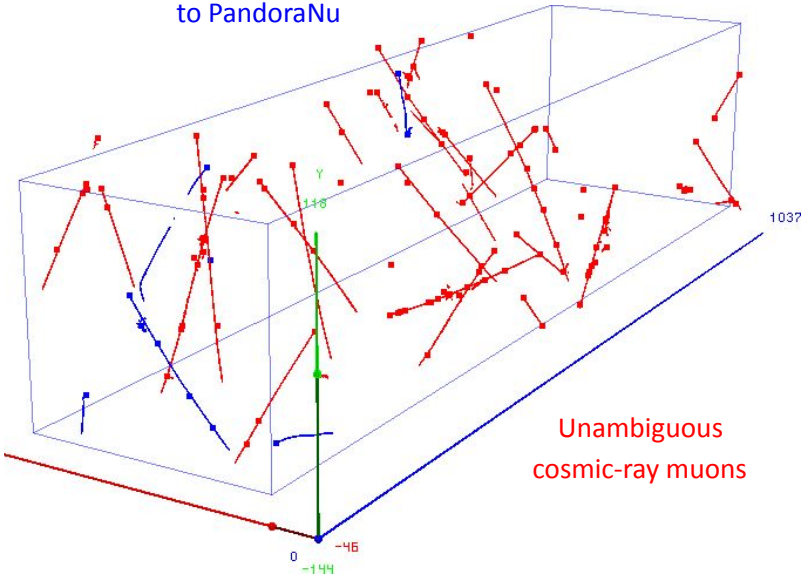
PandoraCosmic → PandoraNu

Unambiguous cosmic-ray muons





Other particles, input to PandoraNu

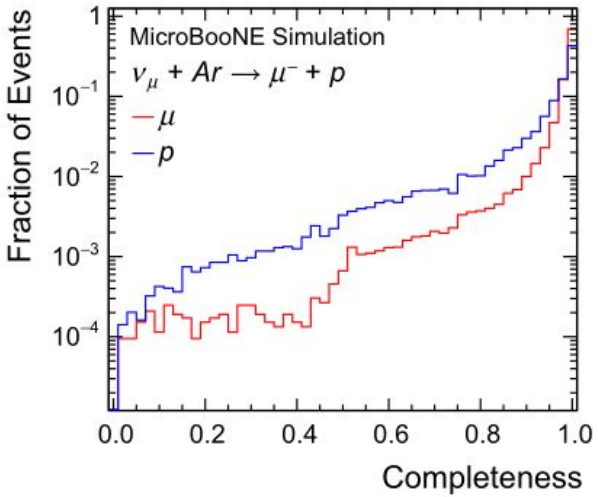


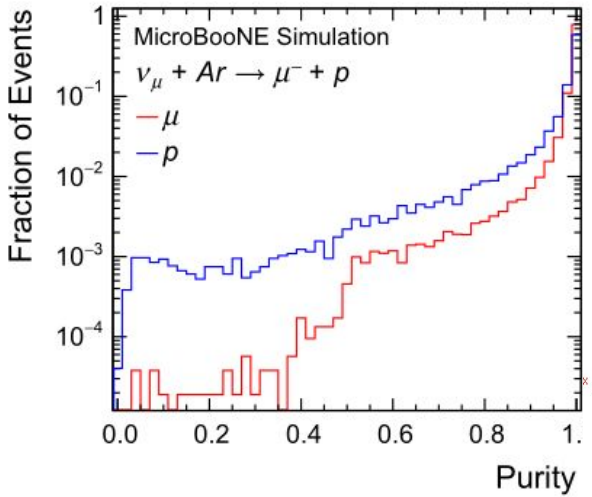
Unambiguous cosmic-ray muons

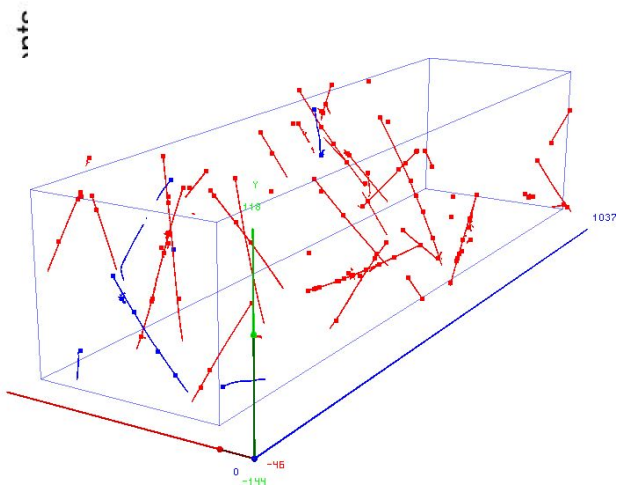
What do we want from reconstruction?

- Purity: representing energy deposits from exactly one true particle.
- Completeness: fraction of the total hits associated with the true particle.

And good vertex identification, but this we are going to see later!





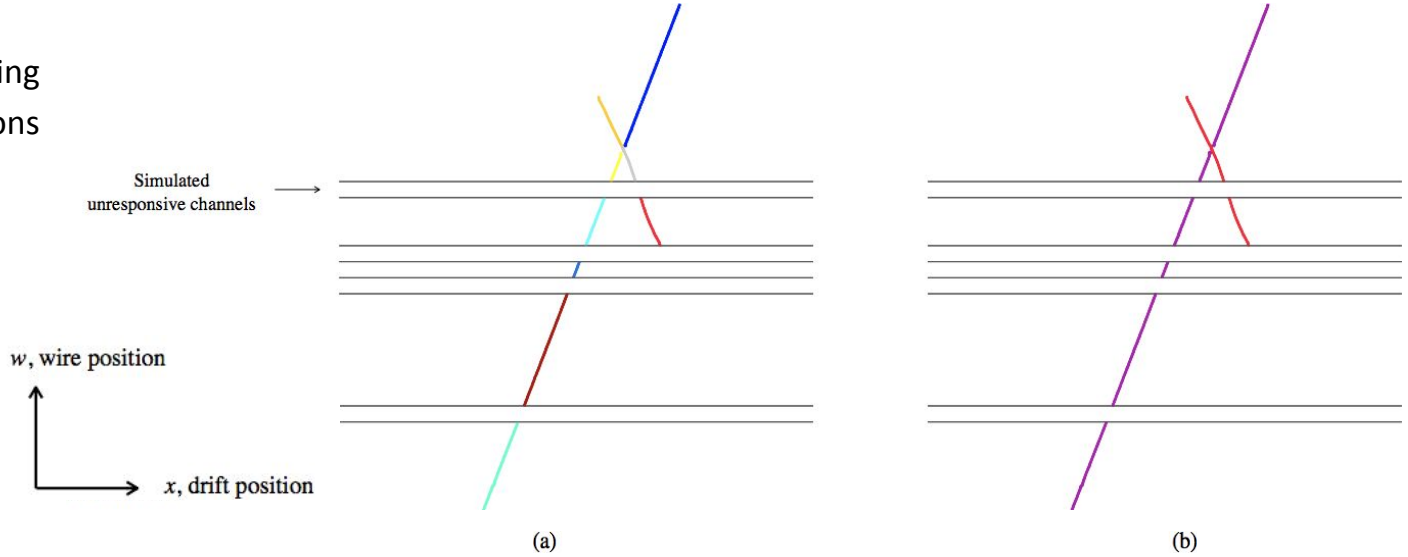


Eur. Phys. J. C (2018) 78:82

Cosmic-Ray Muon Reconstruction - 2D

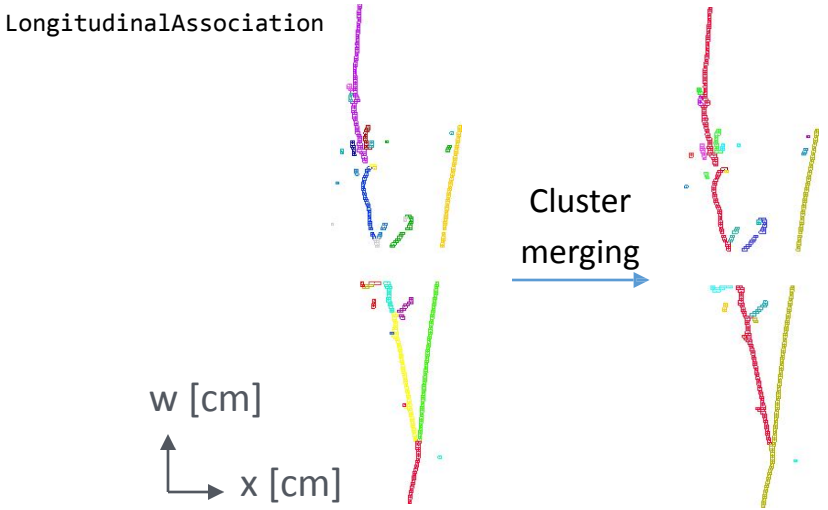
- For each plane, produce list of 2D clusters that represent continuous, unambiguous lines of hits:
  - PandoraCosmic: strongly track-oriented; showers assumed to be delta rays, added as daughters of primary muons; muon vertices at track high-y coordinate.
- Clusters refined by series of cluster-merging and cluster-splitting algs that use topological info.

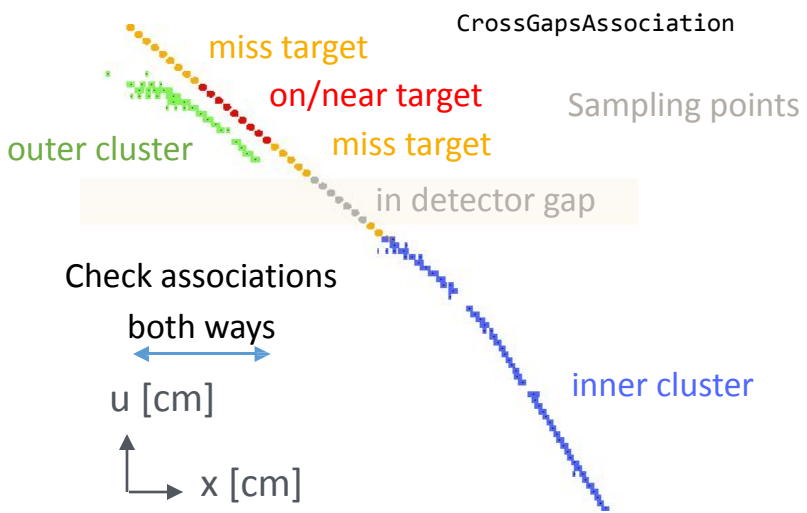
Example: Crossing cosmic-ray muons



Topological Association - 2D

- Cluster-merging algorithms identify associations between multiple 2D clusters and look to grow the clusters to improve completeness, without compromising purity.
  - The challenge for the algorithms is to make cluster-merging decisions in the context of the entire event, rather than just considering individual pairs of clusters in isolation.
  - Typically need to provide a definition of association (for a given pair of clusters), then navigate forwards and backwards to identify chains of associated clusters that can be safely merged.

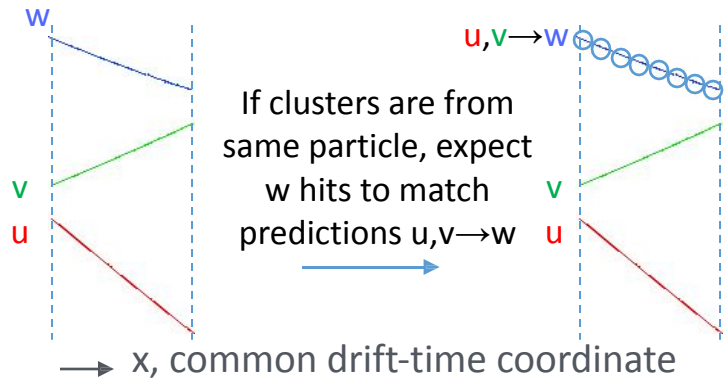




Track Pattern Recognition - 3D

- Our original input was 3x2D images of charged particles in the detector.
- Should now have reconstructed three separate 2D clusters for each particle:
  - Compare 2D clusters from u, v, w planes to find the clusters representing same particle.
  - Exploit common drift-time coordinate and our understanding of wire plane geometry.
  - At given x, compare predictions {u,v⟶w; v,w⟶u; w,u⟶v} with cluster positions, calculating χ^2

Candidate 2D Clusters



If clusters are from same particle, expect w hits to match predictions u,v⟶w

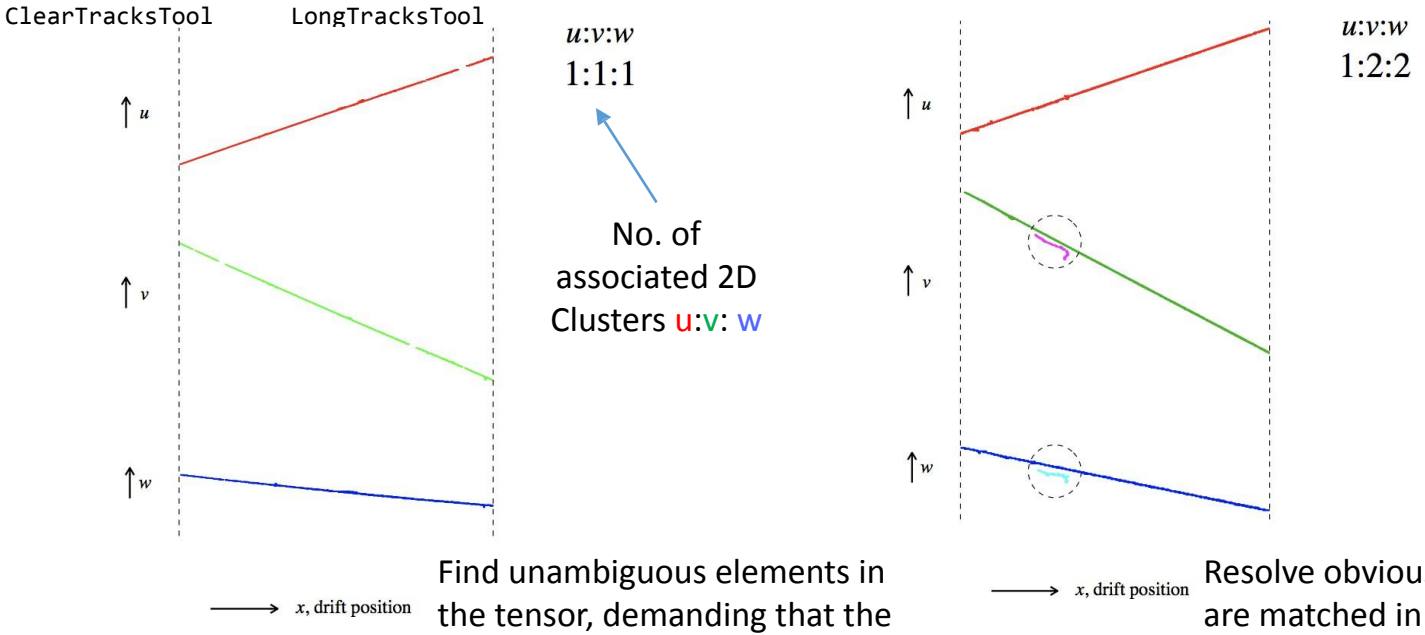
Close agreement: predictions sit right upon real hits here

Sample Cluster consistency across common x-overlap region

Store all results in a "tensor", recording x-overlap span, no. of sampling points, no. of "matched" sampling points and χ2. Documents all 2D cluster-matching ambiguities.

Track Pattern Recognition - 3D

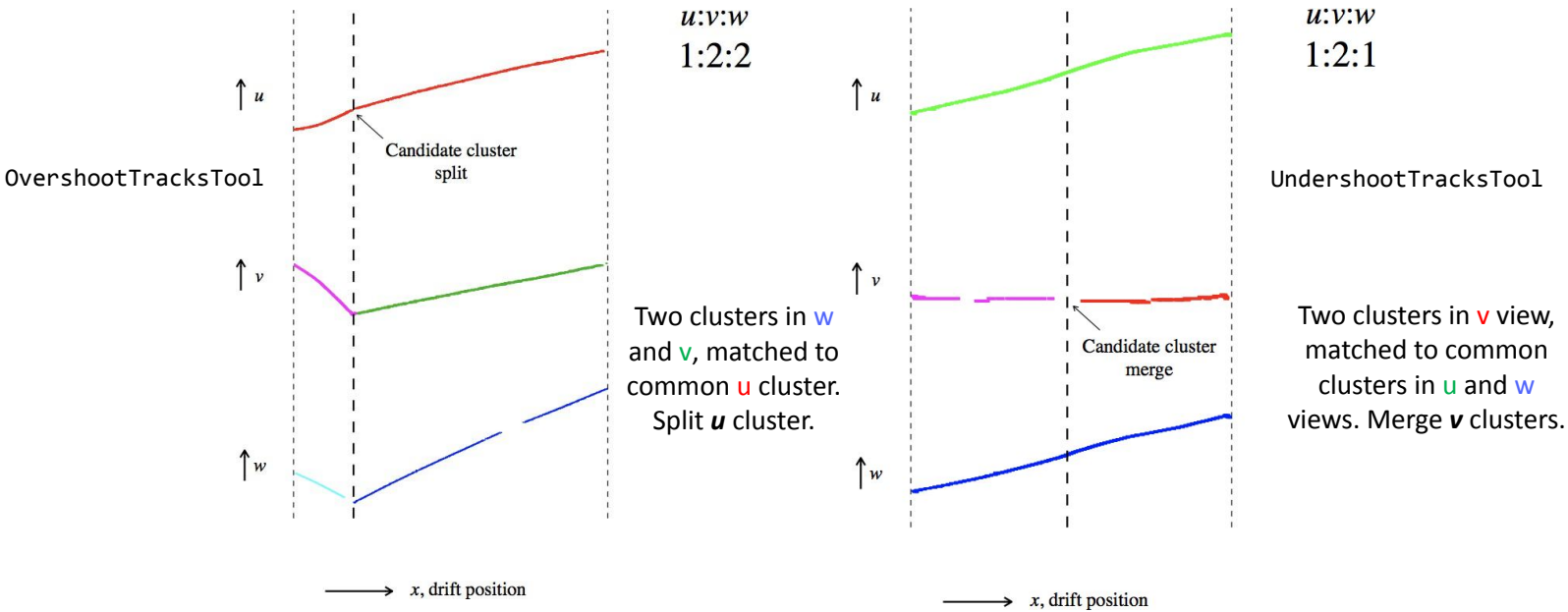
Tensor stores overlap details for trios of 2D clusters. Tools make 2D reco changes to resolve any ambiguities. If a tool makes a change (e.g. splits a cluster), all tools run again.



Find unambiguous elements in the tensor, demanding that the common x-overlap is 90% of the x-span for all three clusters.

Resolve obvious ambiguities: clusters are matched in multiple configurations, but one tensor element is much “better” than others.

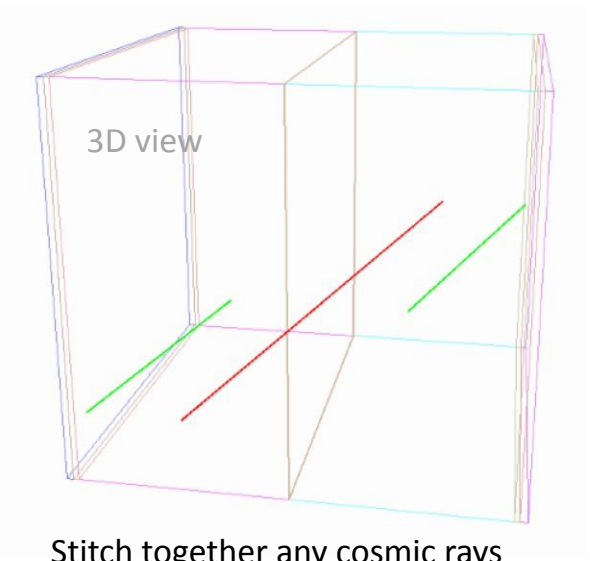
Track Pattern Recognition - 3D



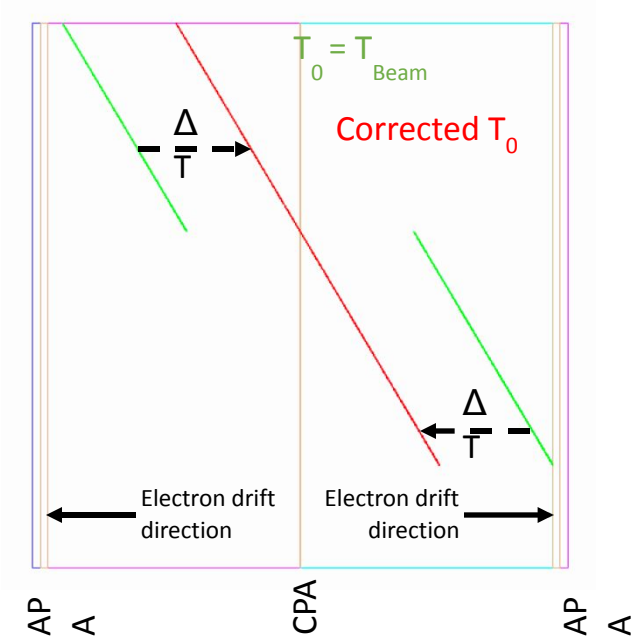
- Use all connected clusters to assess whether this is a true 3D kink topology.
- Modify 2D clusters as appropriate (i.e. merge or split) and update cluster-matching tensor.
- Initial ClearTracks tool then able to identify unambiguous groupings of clusters and form particles.

Stitching and T_0 Identification

- In a LArTPC image, one coordinate derived from drift times of ionisation electrons:
  - But, only know electron arrival times, not actual drift times: need to know start time, T_0
  - For beam particles, can use time of beam spill to set T_0, but unknown for cosmic rays
  - Place all hits assuming $T_0 = T_{Beam}$, but can identify T_0 for any cosmic rays crossing volumes

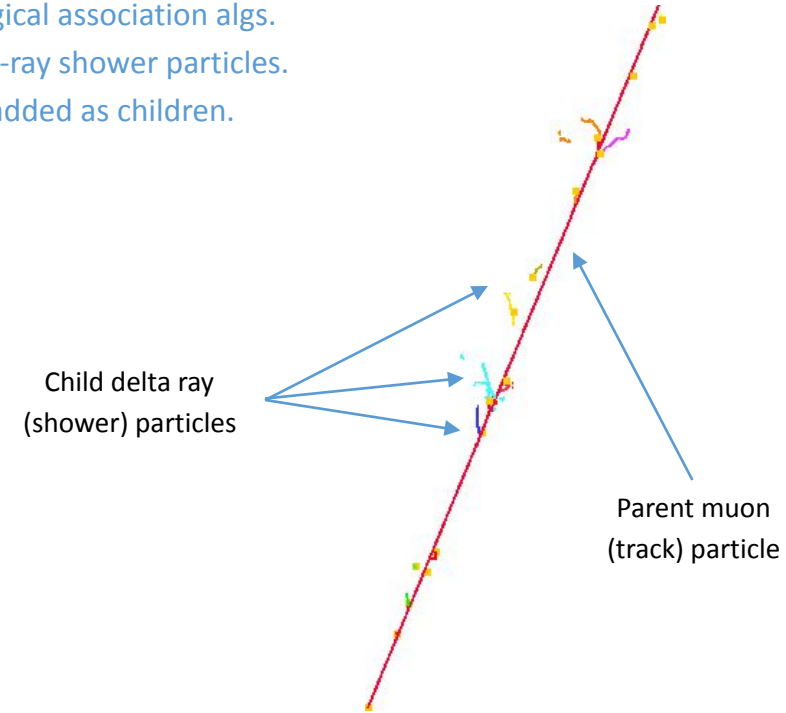


Stitch together any cosmic rays crossing between volumes, identifying T0



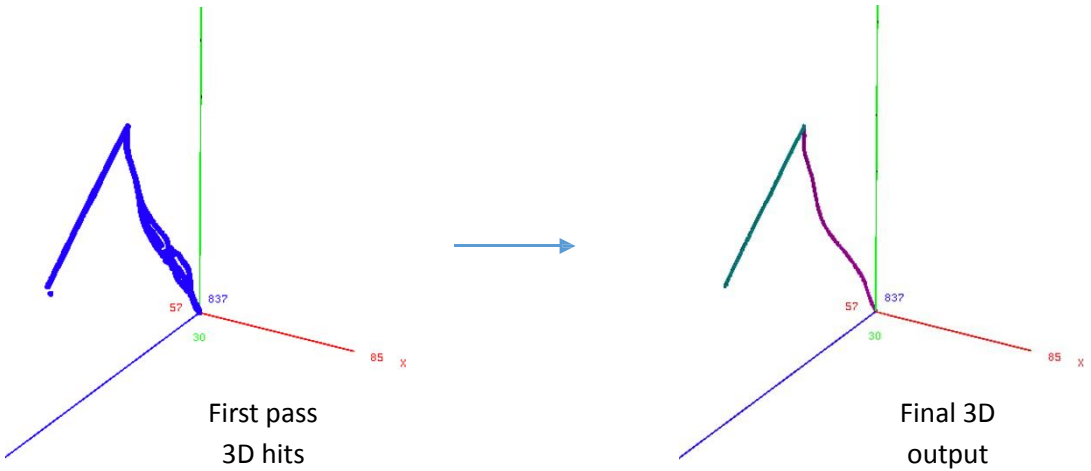
Delta-Ray Reconstruction - 2D, 3D

- Assume any 2D clusters not in a track particle are from delta-ray showers:
  - Simple proximity-based re-clustering of hits, then topological association algs.
  - Delta-ray clusters matched between views, creating delta-ray shower particles.
  - Parent muon particles identified, and delta-ray particles added as children.



3D Hit/Cluster Reconstruction

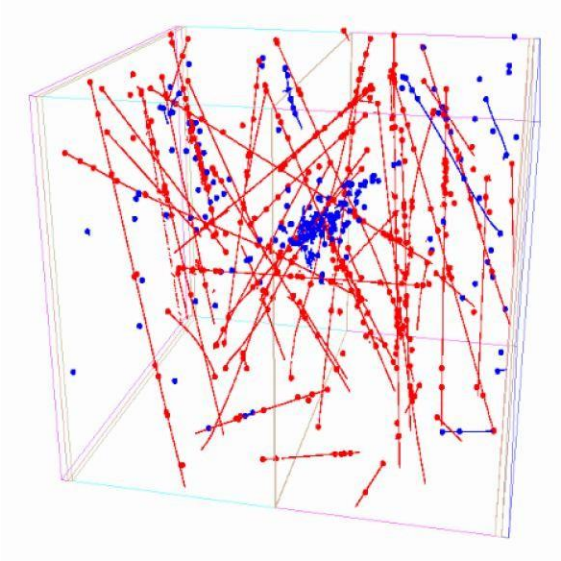
- For each 2D Hit, sample clusters in other views at same x, to provide u_{in}, v_{in} and w_{in} values
- Provided u_{in}, v_{in} and w_{in} values don't necessarily correspond to a specific point in 3D space
- Analytic expression to find 3D space point that is most consistent with given u_{in}, v_{in} and w_{in}
  - $\chi^2 = (u_{out} - u_{in})^2 / \sigma^2 + (v_{out} - v_{in})^2 / \sigma^2 + (w_{out} - w_{in})^2 / \sigma^2$
  - Write in terms of unknown y and z, differentiate wrt y, z and solve
  - Can iterate, using fit to current 3D hits (extra terms in χ^2) to produce smooth trajectory



First pass 3D hits

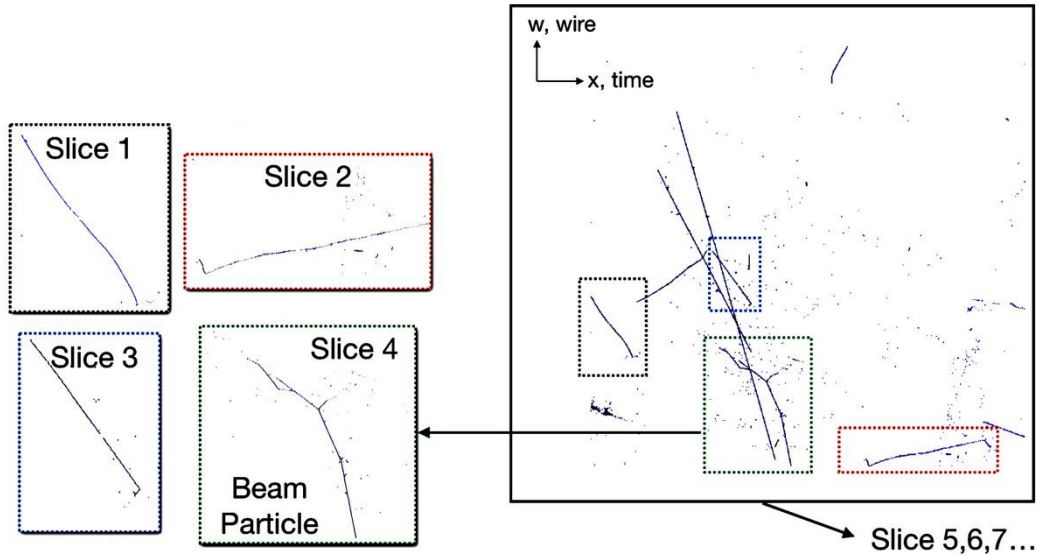
Final 3D output

Cosmic Ray Tagging and Slicing



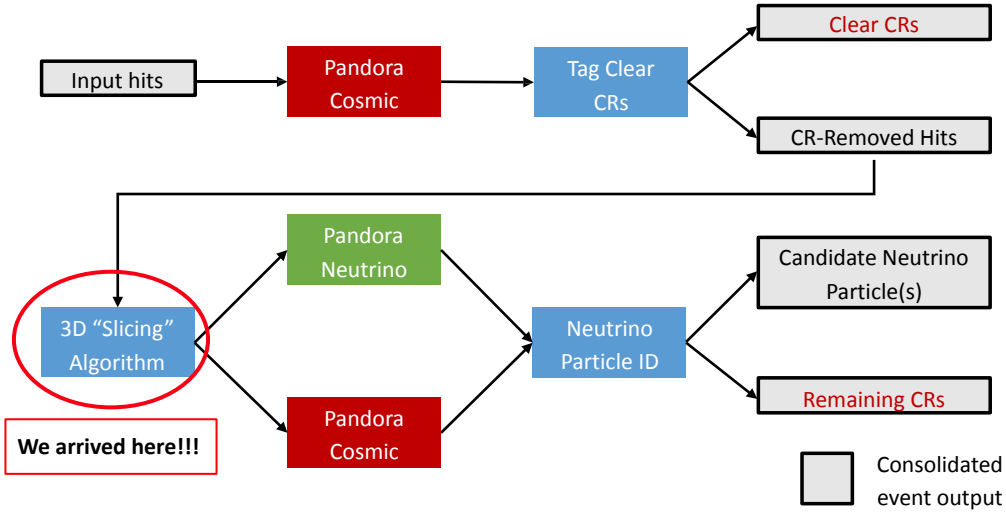
Identify clear cosmic rays (red) and hits to reexamine under test beam hypothesis (blue)

- Clear cosmic rays:
  - Particles appear to be “outside” of detector if $T_0 = T_{Beam}$
  - Particles stitched between volumes using a $T_0 \neq T_{Beam}$
  - Particles pass through the detector: “through going”



- Slice/divide blue hits from separate interactions
- Reconstruct each slice as test beam particle or neutrino candidate
- Then choose between cosmic ray or test beam outcome for each slice

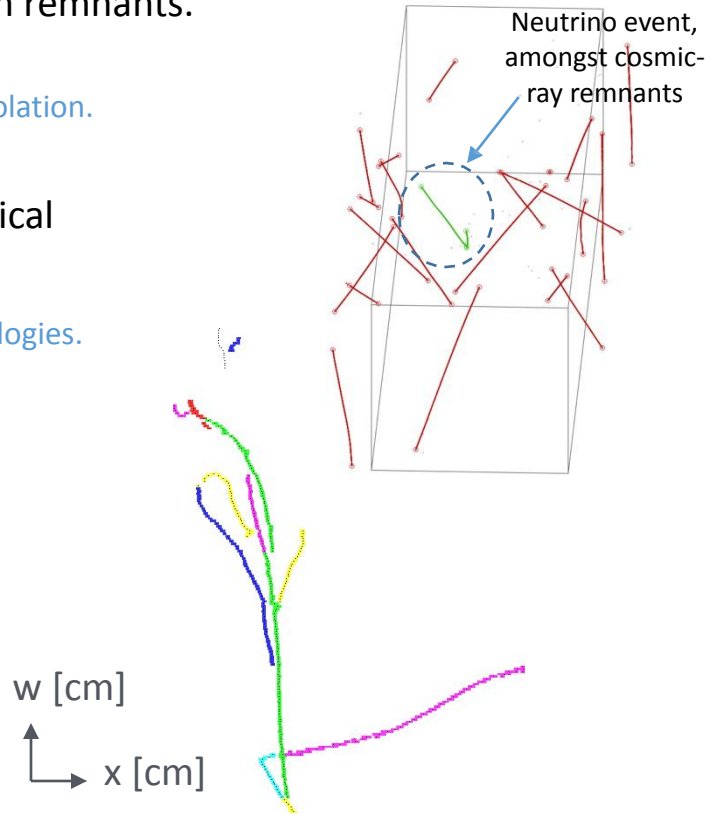
Consolidated reconstruction



Neutrino Candidate!

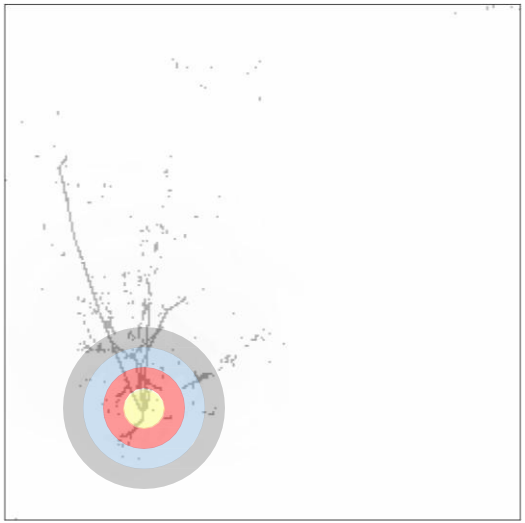
Neutrino Reconstruction

- Must be able to deal with presence of any cosmic-ray muon remnants.
  - Run fast version of reconstruction, up to 3D hit creation
  - “Slice” 3D hits into separate interactions, processing each slice in isolation.
  - Each slice ⇒ candidate neutrino particle.
- Neutrino pass reuses track-oriented clustering and topological association.
  - Topological association algs must handle rather more complex topologies.
  - Specific effort to reconstruct neutrino interaction vertex.
  - More sophisticated efforts to reconstruct showers.

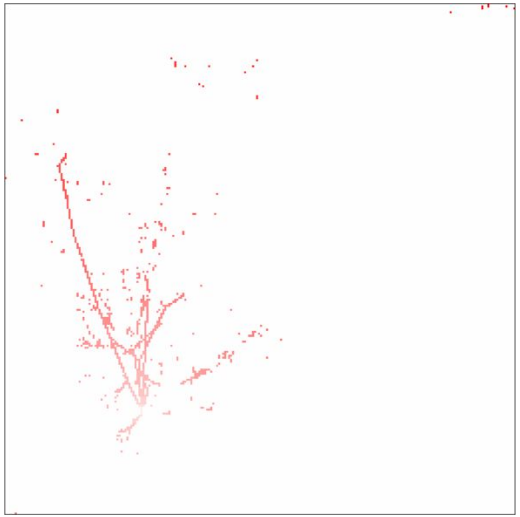


Vertexing reconstruction – U-Net version

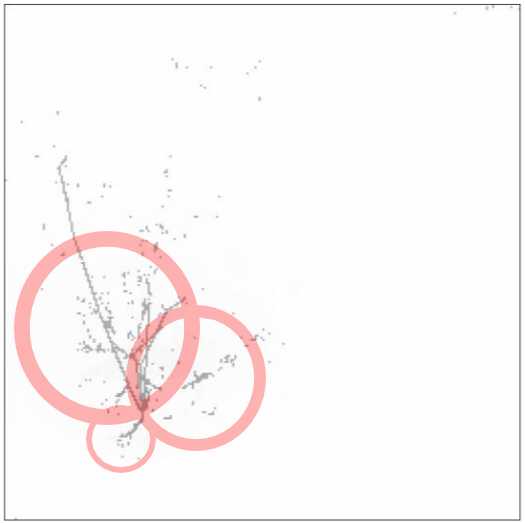
In training hits are assigned a class according to distance from true vertex



Network trained to learn those distances from input images

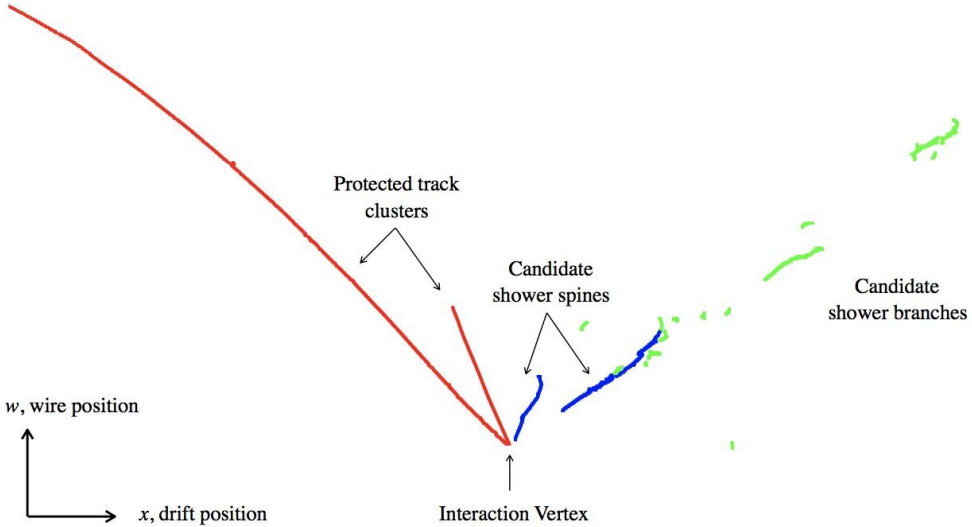


Network infers hit distances and resultant heat map isolates candidate vertex



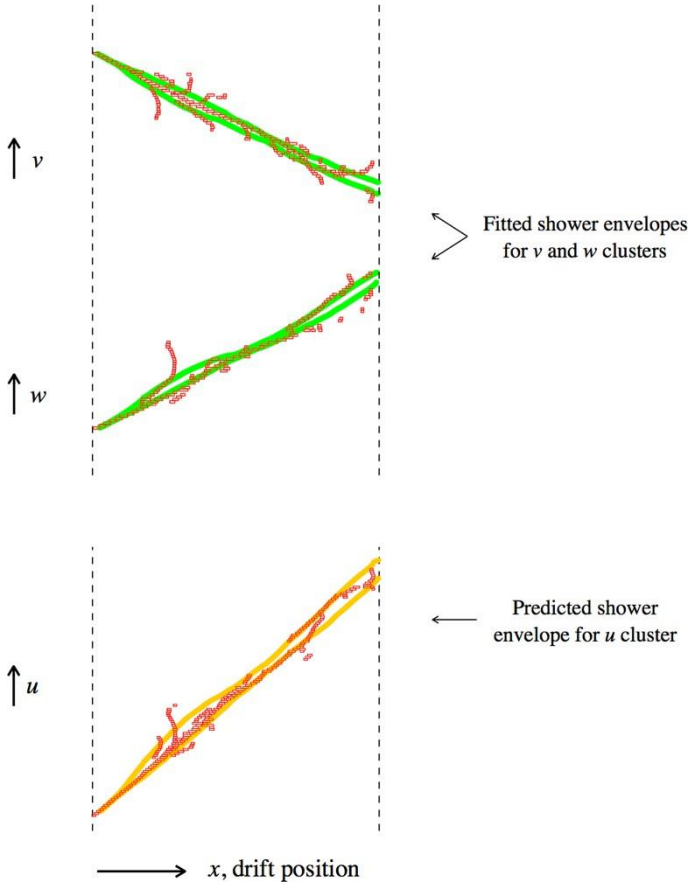
Shower Reconstruction - 2D

- Track reconstruction exactly as in PandoraCosmic, but now also attempt to reconstruct primary electromagnetic showers, from electrons and photons:
  - Characterise 2D clusters as track-like or shower-like and use topological properties to identify clusters that might represent shower spines.
  - Add shower-like branch clusters to shower-like spine clusters. Recursively identify branches on the top - level spine candidate, then branches on branches, etc.



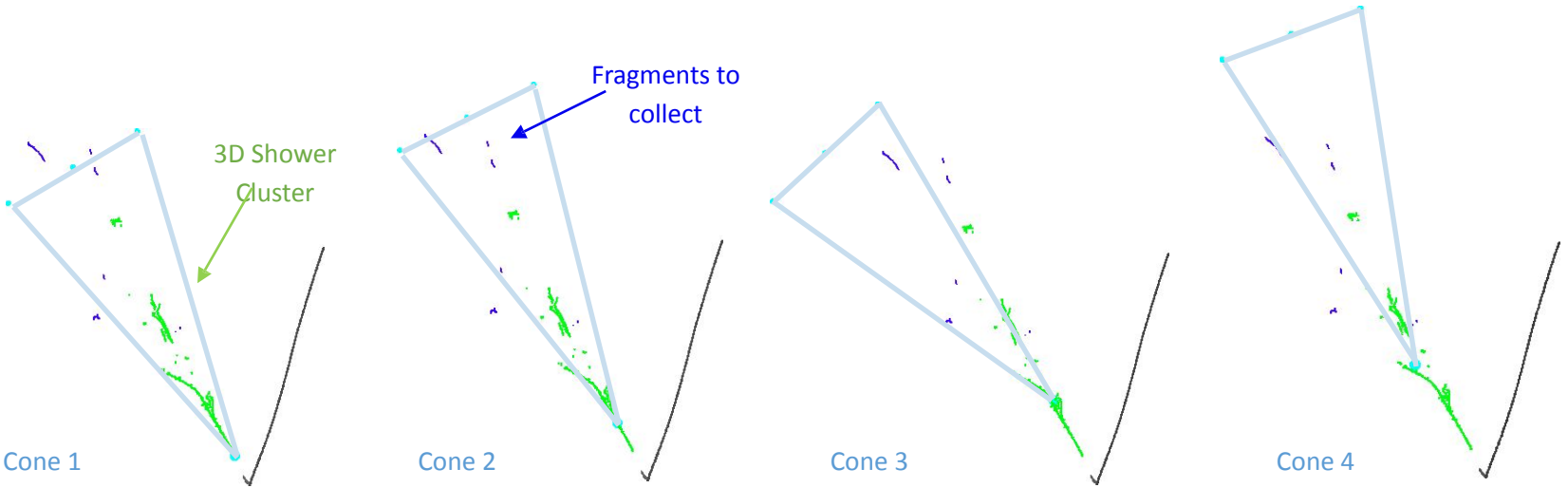
Shower Reconstruction - 3D

- Reuse ideas from track reco to match 2D shower clusters between views:
  - Build a tensor to store cluster overlap and relationship information.
  - Overlap information collected by fitting shower envelope to each 2D cluster.
  - Shower edges from two clusters used to predict envelope for third cluster.



Particle Refinement - 2D, 3D

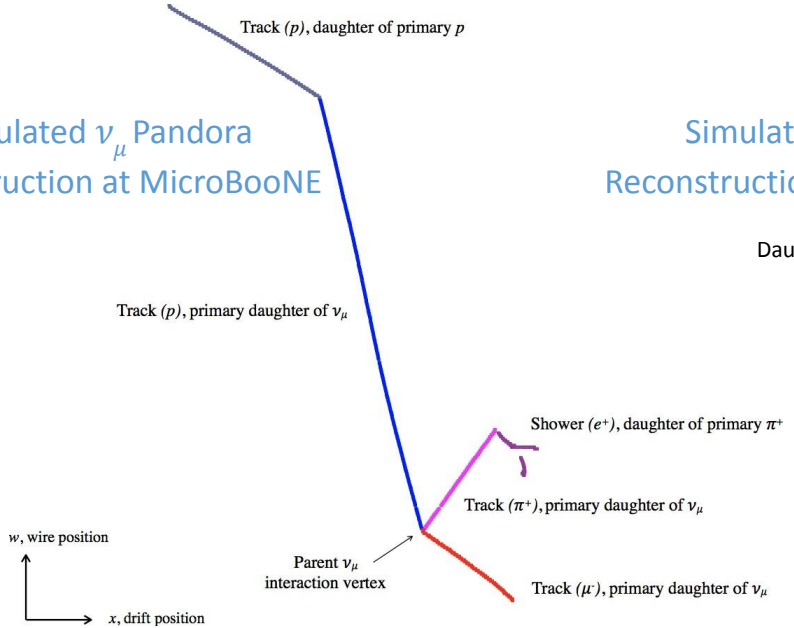
- Series of algs deal with remnants to improve particle completeness (esp. sparse showers):
  - Pick up small, unassociated clusters bounded by the 2D envelopes of shower-like particles.
  - Use sliding linear fits to 3D shower clusters to define cones for merging small downstream shower particles or picking up additional unassociated clusters.
  - If anything left at end, dissolve clusters and assign hits to nearest shower particles in range.



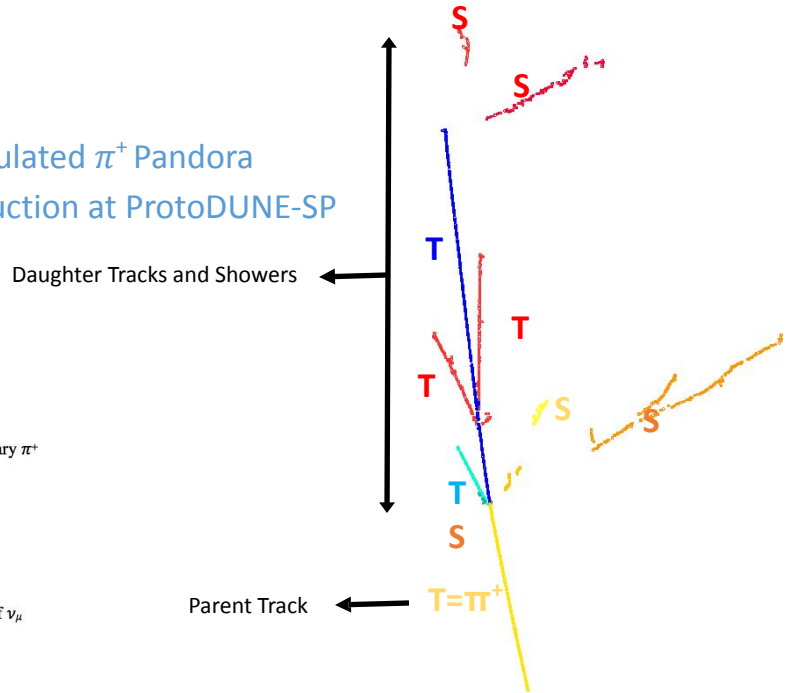
Particle Hierarchy Reconstruction - 3D

- Use 3D clusters to organize particles into a hierarchy, working outwards from interaction vtx
  - Use the hierarchy to access particles in analyzers

Simulated ν_μ Pandora Reconstruction at MicroBooNE

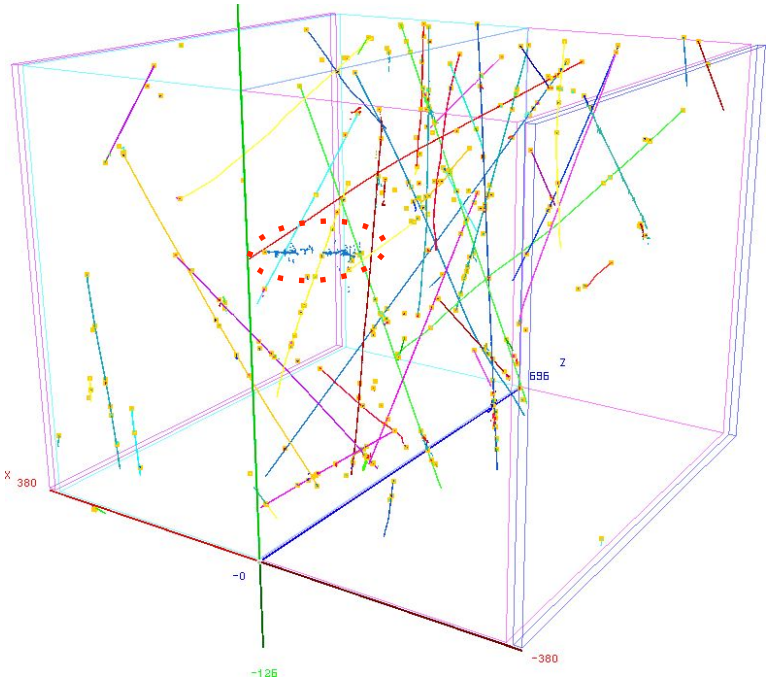


Simulated π^+ Pandora Reconstruction at ProtoDUNE-SP

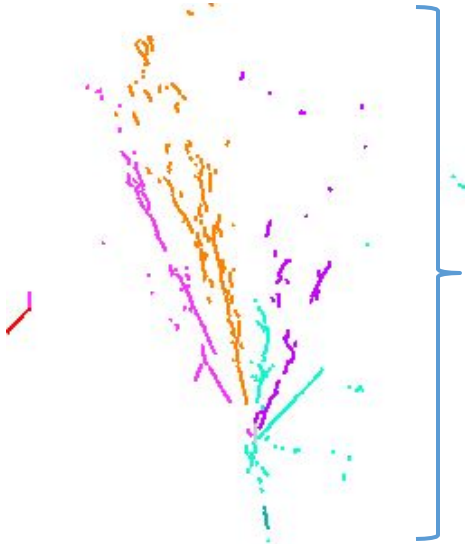


EPJC (2018) 78:82

Consolidated output



E.g. Reconstruction output: test beam particle (electron)
and: N reconstructed cosmic-ray muon hierarchies



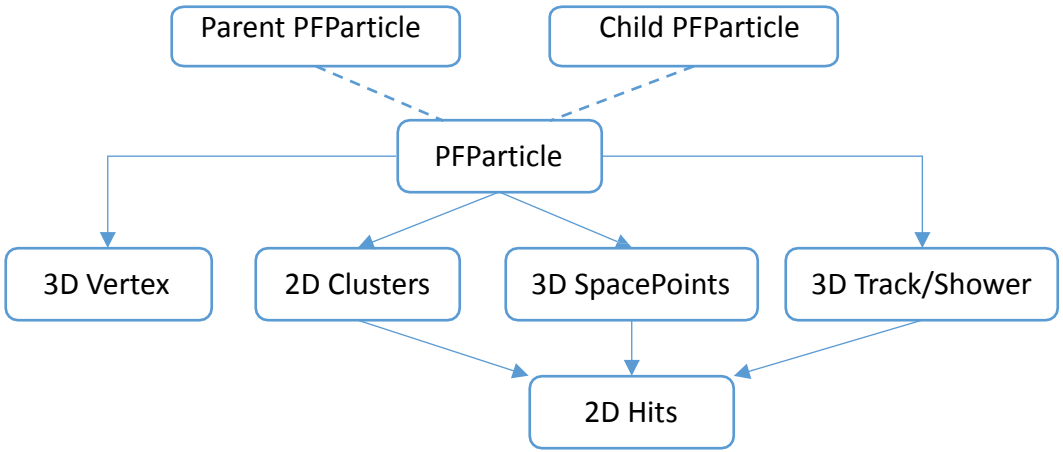
Child tracks and showers

Parent track

E.g. Test beam particle: charged pion

Reconstruction Output

- Must translate output from Pandora Event Data Model to LArSoft Event Data Model. The key output is the PFParticle (PF ⇒ Particle Flow):
  - Each PFParticle corresponds to a distinct track or shower and is associated to 2D clusters.
  - 2D clusters group hits from each readout plane, and are associated to the input 2D hits.
  - PFParticles also associated to 3D spacepoints and a 3D vertex.
  - PFParticles placed in a hierarchy, with identified parent-daughter relationships.
  - PFParticles flagged as track-like or shower-like (both outcomes are persisted).

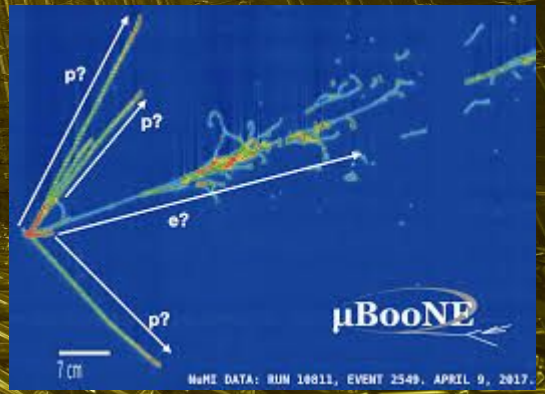


Just the most important outputs shown here

Overall summary

- The use of Liquid Argon technology is one of the cornerstones of the current and future neutrino programmes.
- High-performance reconstruction techniques are required in order to fully exploit the imaging capabilities offered by LArTPCs:
  - Pandora multi-algorithm approach uses large numbers of decoupled algorithms to gradually build up a picture of events.
  - Output is a carefully-arranged hierarchy of reconstructed particles, each corresponding to a distinct track or shower.

4. Studying/Debugging the Reconstruction



Reconstruction is hard

- The events you've looked at so
- Two track-like trajectories eme
- Nonetheless, you may have se e.g.:

Split tracks

tracks

Merg

- Most neutrino interactions wil
  - Track-like and shower-like topolo
  - Re-interactions
  - High particle multiplicity

You will come across mis-rec

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

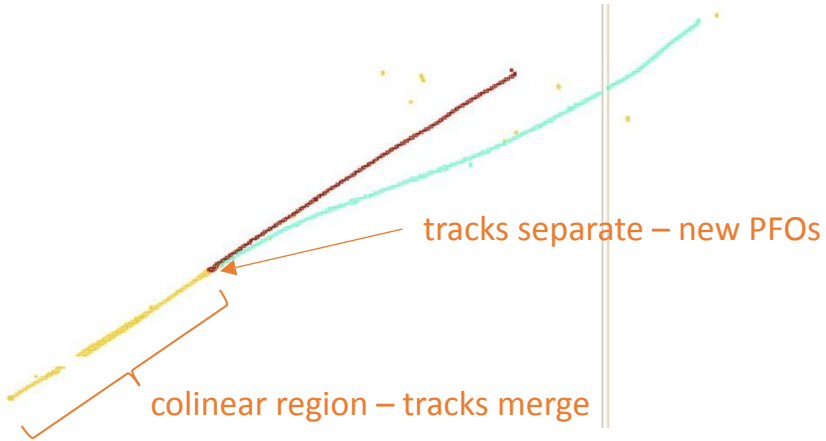
- However, a number of circumstances exist that more reliably cause problems
  - We'll give some **examples** in the next few slides!

- 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 (1)

- 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

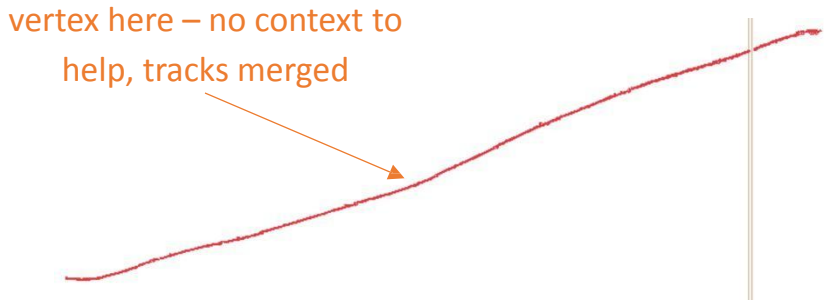
Example:
Tracks merged due to collinearity



Overlapping and back-to-back trajectories (2)

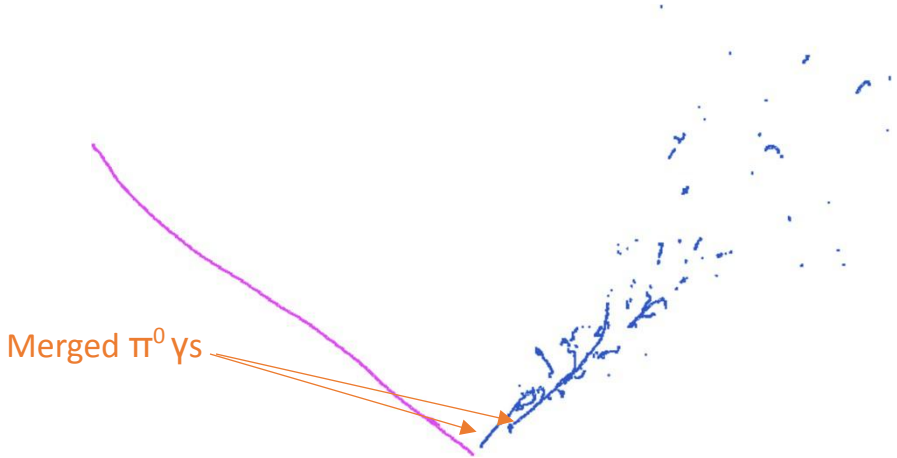
- Neutrino interactions happen in 3D, but we have (typically) three, 2D projections of the interaction
- Trajectories that are clearly distinct in 3D can appear indistinguishable in a 2D projection
- If the trajectories can be distinguished in two of the projections, it is still possible to effectively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



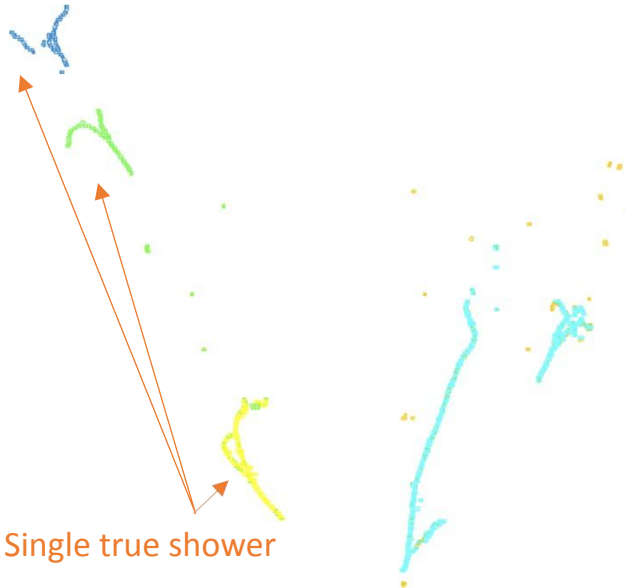
Small opening angles and sparse showers (1)

- 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



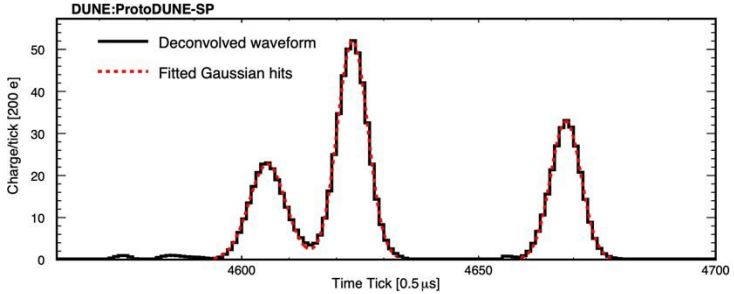
Small opening angles and sparse showers (2)

- 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

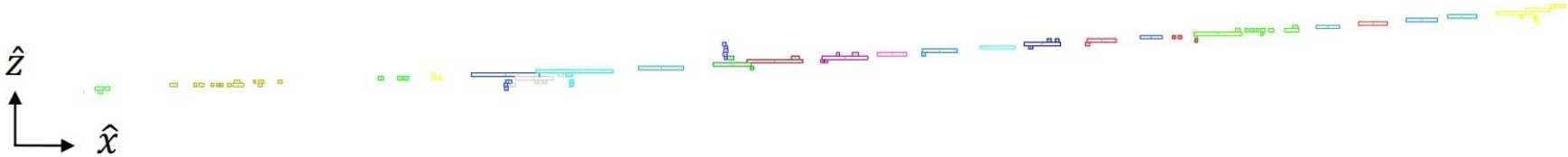


Awkward trajectories (1)

- 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

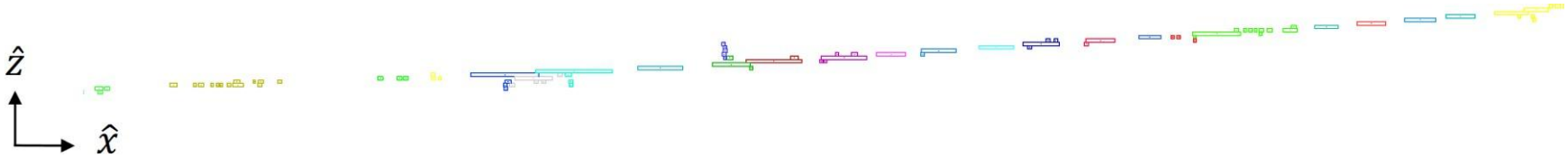


- If a particle trajectory is perpendicular to the wire planes, its drift electrons interact with a single wire/strip, from which it is challenging to extract hits
  - The result is **a small number of wide hits** (i.e. a high uncertainty in the position along the drift direction)
  - Such hits can be 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 (1)

- 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

- It's not realistic to expect you to develop new algorithms today
- We've created a very simplistic reconstruction workflow that introduces failures that we want you to try to fix
  - You can introduce existing algorithms
  - You can tune algorithms, but keep in mind, if you tune things too much for one event, you'll break others

Goals

- Main goal – Identify the source of reconstruction failures
  - Add visualizations at key points of the reconstruction chain to understand how the reconstruction proceeds
  - Investigate the behaviour of relevant algorithms to understand how they work
- Secondary goal – Fix events
  - Identify algorithms that might address the issues and add those algorithms into the chain
  - Attempt to tune key parameters in XML, as needed
  - Continue to use intermediate visualization to understand what your changes have done
- Please don’t worry if you don’t fix the issues
  - Even if you fix an error in one place, your efforts might be undone later in the reconstruction chain
    - The reconstruction algorithms interact with each other
  - You may be able to fix parts of an event, but not others
  - Reconstruction is hard

Investigate the baseline reconstruction output

A new event

- We'll be using custom neutrino events for this task, rather than the 1 muon, 1 proton events you've been looking at so far:

```
cp /home/leonardo/arqsEBN/ reco1_neutrino.root $MRB_TOP/reco/work
```

- Make sure to be in the directory you've been working in so far for the reconstruction tutorials and then copy the configuration files we'll be using for this session:

```
$ cd $MRB_TOP/reco/config
$ cp /home/leonardo/arqsEBN/ PandoraSettings_Master_Simple.xml .
$ cp /home/leonardo/arqsEBN/ PandoraSettings_Neutrino_Simple.xml .
```

- These files contain a greatly simplified reconstruction workflow, where many Pandora algorithms have been removed, to introduce reconstruction failures to these events

An aside on enabling the Pandora event display

- In this session we'll be using a custom configuration, but the typical top-level Pandora configuration file will be called something like PandoraSettings_Master_DUNEFD.xml
- If all you want to do is enable the Pandora event display in an otherwise standard configuration you can copy this file locally and enable Pandora Monitoring with the following modification:

```
<pandora>
    <!-- GLOBAL SETTINGS -->
    <IsMonitoringEnabled>true</IsMonitoringEnabled>
    ...
```

Ensuring your custom files are found

- At the moment, Pandora won't know about our custom configuration files, so we need to add our config directory to the FW_SEARCH_PATH so Pandora knows where to look for it (you might already have this in a setup script) and do the same for the FHICL_FILE_PATH:

```
$ export FW_SEARCH_PATH=$MRB_TOP/reco/config:$FW_SEARCH_PATH
$ export FHICL_FILE_PATH=$MRB_TOP/reco/config:$FHICL_FILE_PATH
```

- Now we need to set up a new FHiCL file to run our custom configuration…

Writing a FHiCL file to run custom reconstruction

- Let's make a new FHiCL file that just runs the reconstruction using our custom XML configuration and produces an output file without the default timestamp tag

```
$ vim reco_driver.fcl
```

- Add the lines below to reco_driver.fcl, save and close:

```
#include "standard_reco2_dune10kt_1x2x6.fcl"

physics.producers.pandora.EnableMCParticles:  true
physics.producers.pandora.ConfigFile: "PandoraSettings_Master_Simple.xml"
physics.reco: [ @sequence::dunefd_horizdrift_2dclustering,
                pandora ]
physics.end_paths: []
```

Start from the standard reco chain

Point to our new XML settings file
This is a very convenient way to customize FHiCL – inherit and minimally modify

Only run up to Pandora pat rec

- Run Pandora over the event and look at the reconstruction output

```
$ lar -c reco_driver.fcl -s reco1_neutrino.root -n 1
```

When copy-pasting commands there can be strange character conversions, particularly for hyphens, these often show up as an inability to find a config file. Try deleting the hyphens and just typing them

A brief introduction to the Pandora event display

Understanding the Pandora event display

Every time the visual monitoring algorithm runs, we get a new event display (enumerated from zero)

Try checking and unchecking the boxes to turn on and off the hits from each of the views

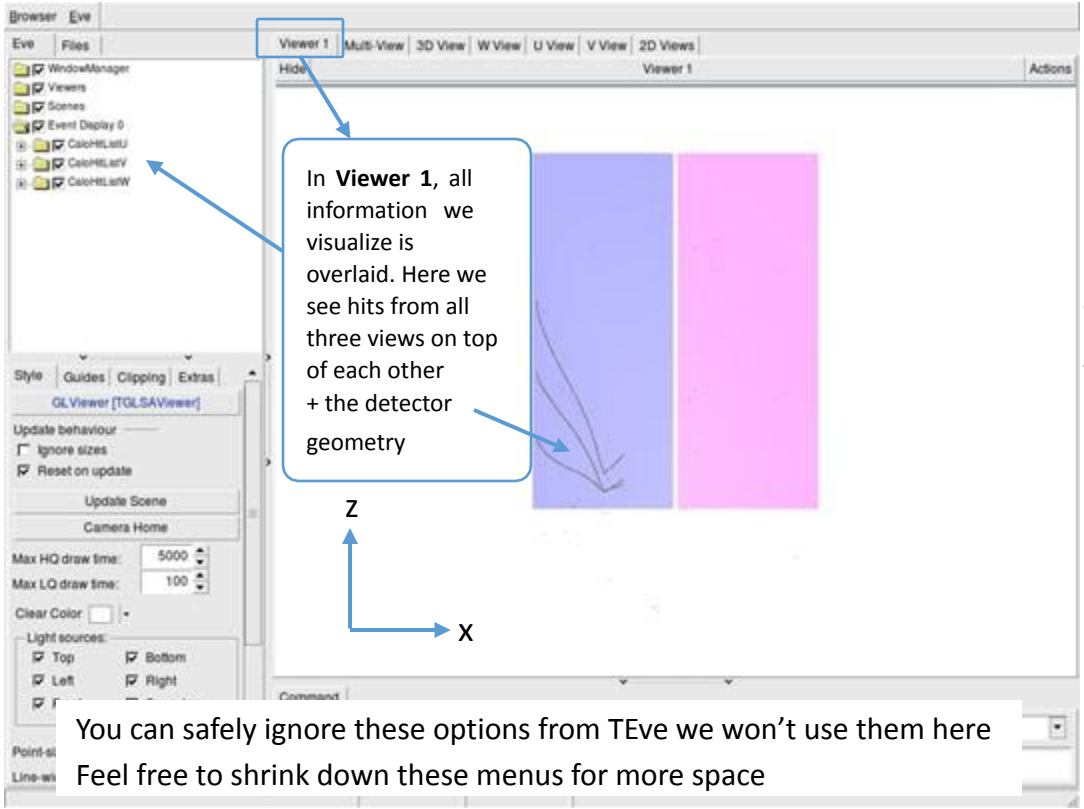
- ❒ CaloHitListU
- ❒ CaloHitListV
- ❒ CaloHitListW

The 2D hit coordinates are stored in Pandora as 3D coordinates (X, Y, Z)

X = drift time coordinate

Y = 0

Z = wire number coordinate



Wheel up - zoom out
Wheel down - zoom in Wheel
press + drag - pan viewport

(Or whatever the mapping of these operations is for your laptop. Arrow keys can move the view around as well)

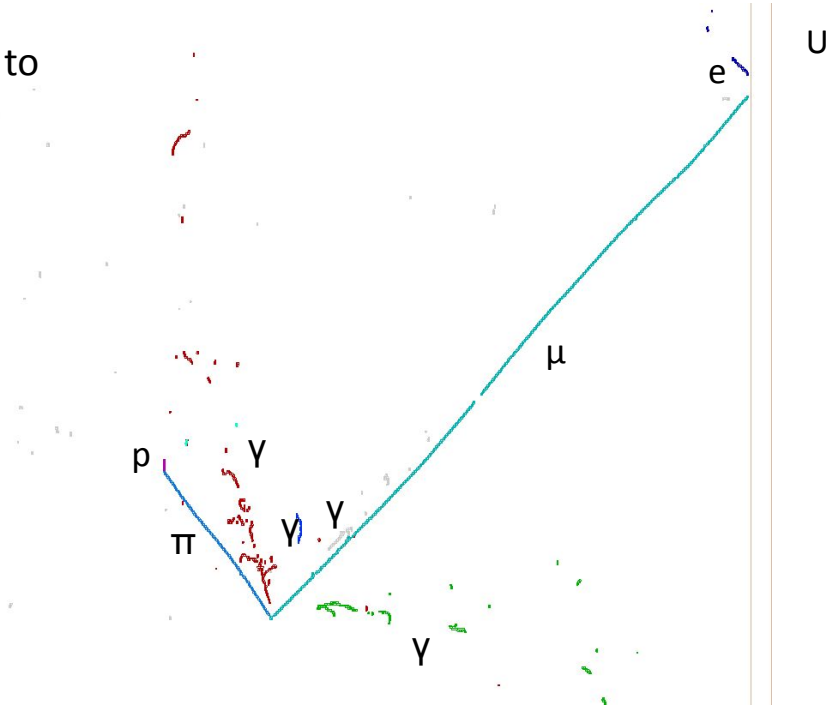
W - wireframe mode
R - return from wireframe mode

If you click in the terminal window and press Return ↵ Pandora will continue running

An example event

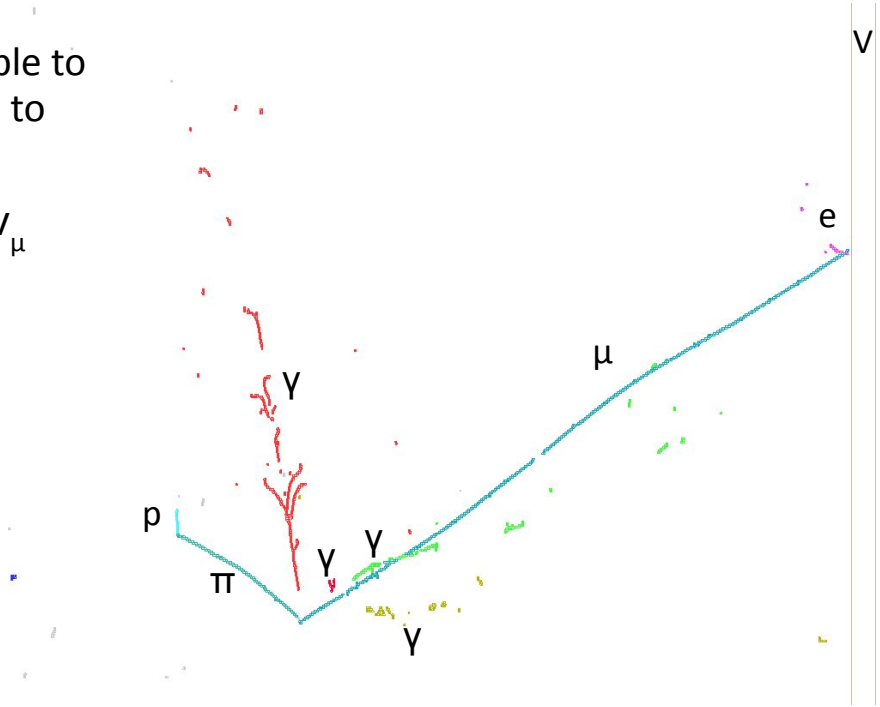
The underlying event (1)

- We'll highlight a couple of events that are available to investigate in this session, but we'll focus on one to outline the process
- Depicted to the right is the MC truth for this CC ν_μ interaction in the U view (V and W on the next two slides)
- The primary particles in this event are
  - Muon
  - Charged pion – from $\Sigma^+ \rightarrow n\ \pi^+$
  - Four photons – from two $\pi^0 \rightarrow \gamma\ \gamma$



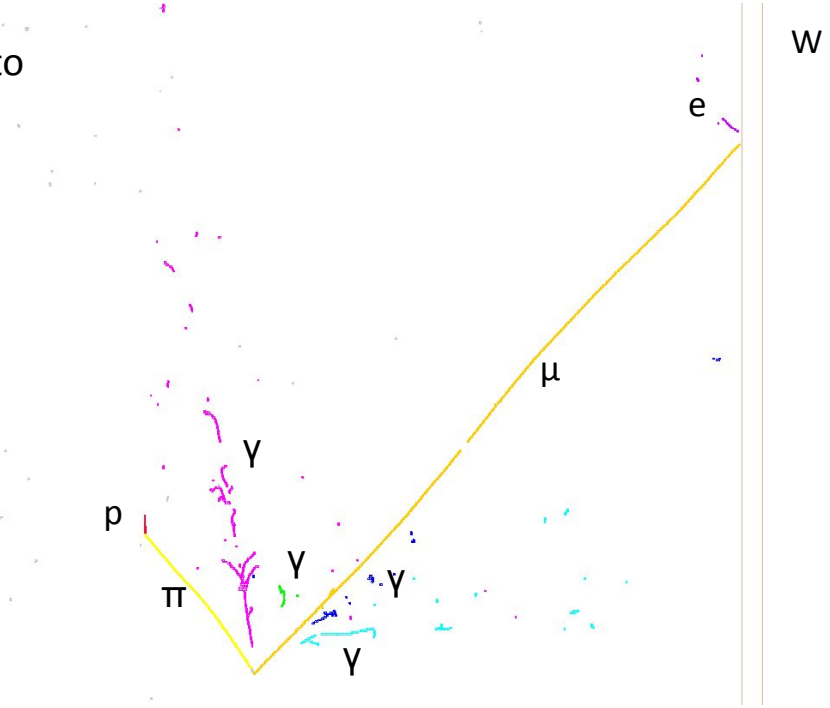
The underlying event (2)

- We'll highlight a couple of events that are available to investigate in this session, but we'll focus on one to outline the process
- Depicted to the right is the MC truth for this CC ν_μ interaction in the V view
- The primary particles in this event are
  - Muon
  - Charged pion – from $\Sigma^+ \rightarrow n\ \pi^+$
  - Four photons – from two $\pi^0 \rightarrow \gamma\ \gamma$



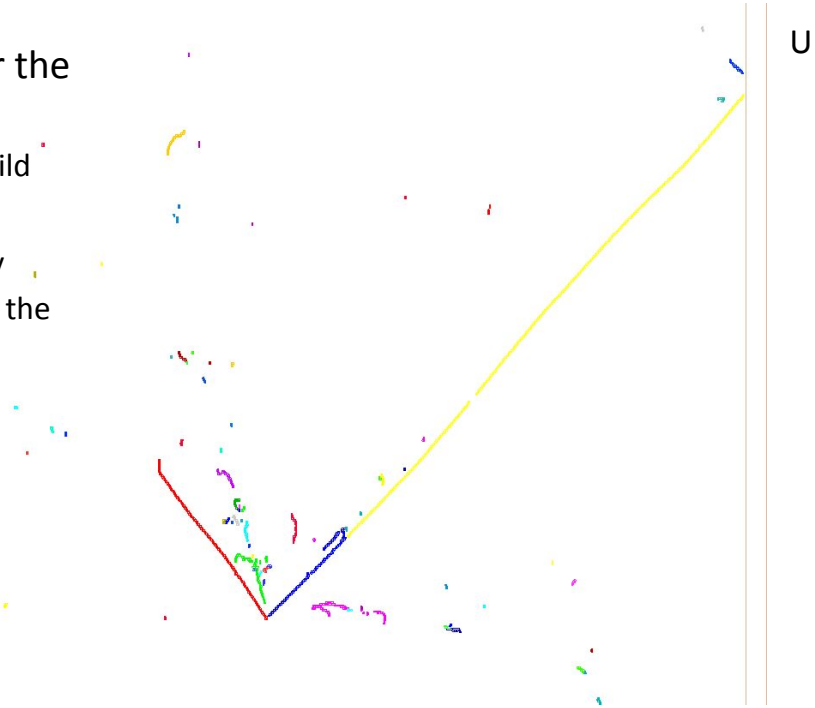
The underlying event (3)

- We'll highlight a couple of events that are available to investigate in this session, but we'll focus on one to outline the process
- Depicted to the right is the MC truth for this CC ν_μ interaction in the W view
- The primary particles in this event are
  - Muon
  - Charged pion – from $\Sigma^+ \longrightarrow n\ \pi^+$
  - Four photons – from two $\pi^0 \longrightarrow \gamma\ \gamma$



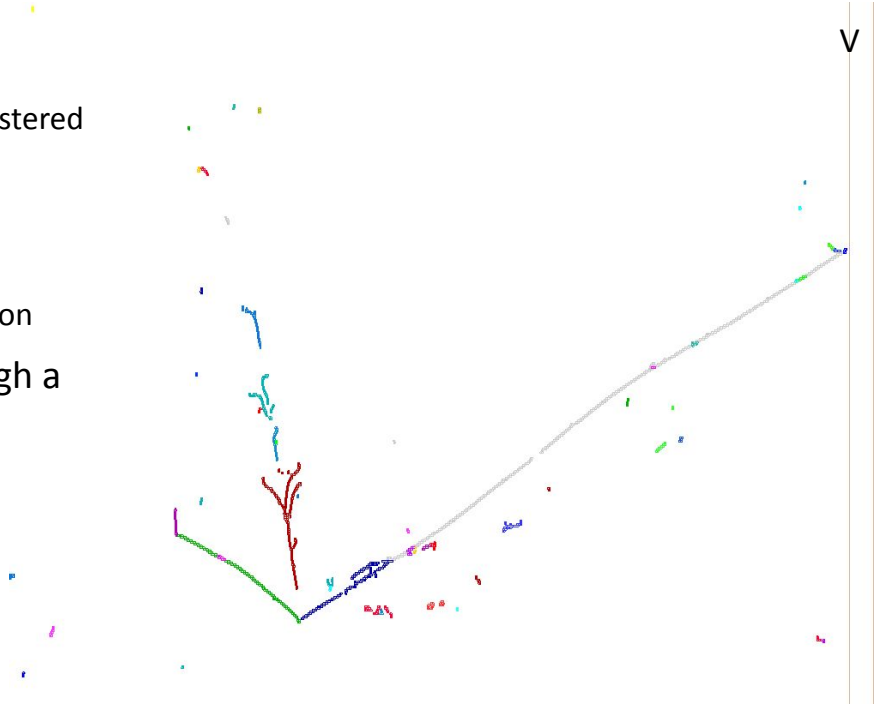
Reconstructing the event (1)

- If we run our simplified reconstruction and consider the clusters in the U view, a few things are evident:
  - The pion is well-reconstructed, but also merged with its child proton
  - The muon is split into two clusters as a result of the nearby photon activity (the end of the first cluster actually follows the connected hits from the photon)
  - The shower reconstruction is much less successful
    - There is notable shower fragmentation



Reconstructing the event (2)

- In the V view, there are some additional issues:
  - Both the muon and pion have additional separately clustered segments due to the more transverse topology
  - However, the child proton of the pion is separately reconstructed
  - The shower reconstruction exhibits similar fragmentation
- The W view, not shown, has similar features, though a better a track reconstruction



A few pointers (1)

- There are more than 100 algorithms available to Pandora's pattern recognition process, so to help narrow things down, we'll highlight a few potentially interesting algorithms that might help resolve some or all of these issues
- We won't go through the details now, but in the backup section of these slides are brief descriptions of what these algorithms do, with a focus on the 2D algorithms
- 2D clustering algorithms – these can be used after algorithms like LArTransverseExtension
  - LArCrossGapsAssociation, LArCrossGapsExtension, LArOvershootSplitting, LArBranchSplitting, LArKinkSplitting, LArTrackConsolidation, LArHitWidthClusterMerging
  - Note these algorithms run for each of U, V and W, and so anything you add must be replicated in each of the relevant sections
- 2D mop up algorithms – can be used after LArThreeDShowers
  - LArBoundedClusterMopUp, LArConeClusterMopUp, LArNearbyClusterMopUp

A few pointers (2)

- There are more than 100 algorithms available to Pandora's pattern recognition process, so to help narrow things down, we'll highlight a few potentially interesting algorithms that might help resolve some or all of these issues
- We won't go through the details now, but in the backup section of these slides are brief descriptions of what these algorithms do, with a focus on the 3D algorithms
- 3D track algorithms – can be used after LArThreeDLongitudinalTracks
  - LArThreeDTrackFragments
- 3D recovery algorithms - can be used after LArThreeDShowers
  - LArVertexBasedPfoRecovery, LArParticleRecovery
- 3D mop up algorithms - can be used after LArThreeDHitCreation
  - LArSlidingConePfoMopUp, LArSlidingConeClusterMopUp, LArIsolatedClusterMopUp

Understanding the reconstruction (1)

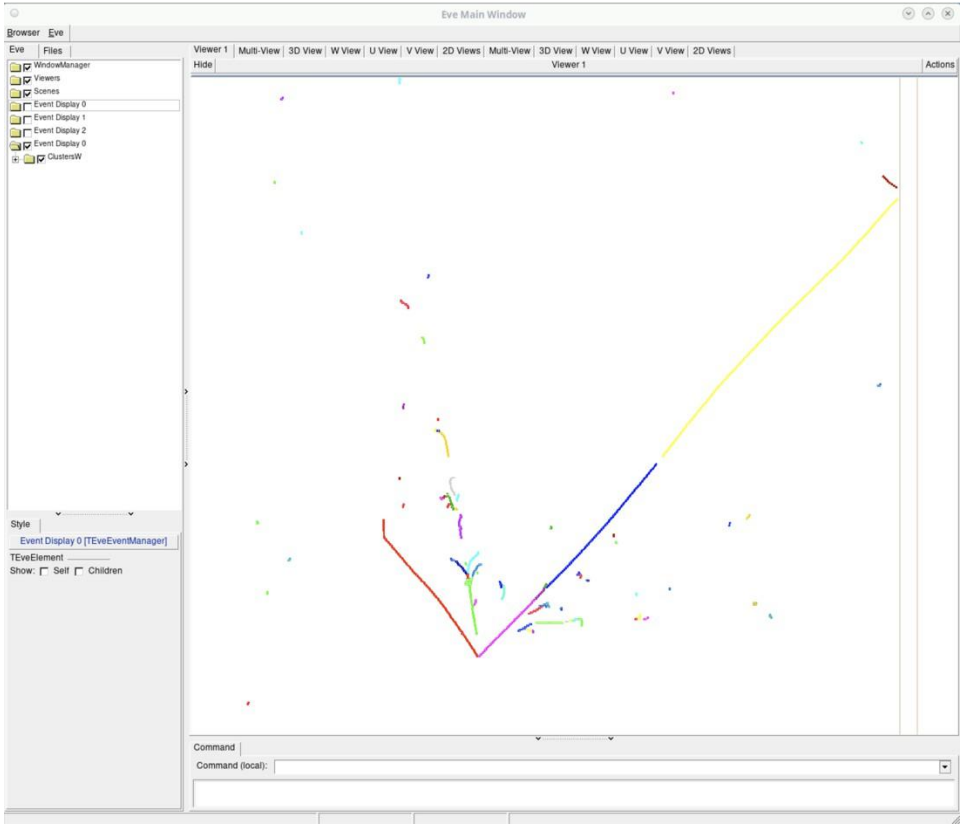
- We're going to use Pandora's event display to explore how the clustering proceeds in an effort to understand where we might want to intervene
- We'll start by focusing on the 2D clustering algorithms: After the W view <LArClusteringParent> algorithm (check it's in the W view block), add the following instance of the LArVisualMonitoring algorithm

```
<!-- TwoDReconstruction -->
...
<algorithm type = "LArClusteringParent">
    <algorithm type = "LArTrackClusterCreation" description =
    "ClusterFormation"/>
    <InputCaloHitListName>CaloHitListW</InputCaloHitListName>
    ...
</algorithm>

<algorithm type = "LArVisualMonitoring">
    <ClusterListNames>ClustersW</ClusterListNames>
    <ShowDetector>true</ShowDetector>
</algorithm>
```

- You may also want to set ShouldDisplayAlgorithmInfo to true at the beginning of both the master and neutrino XML, in order to keep track of where you are as you step through the event displays

Understanding the reconstruction (2)

- Run Pandora and start to observe how the clustering proceeds
- Focus on the W view to begin with by deselecting the other views in the tree
- Note how small many of the clusters are
- Provisional clustering is track-centric and quite strict
  - Ambiguity, gaps or sharp deviations in trajectory limit the growth of clusters
  - In this view there aren't many ambiguities, so there's little doubt about how the tracks should grow, and they're well reconstructed
  - The showers are fragmented



Understanding the reconstruction (3)

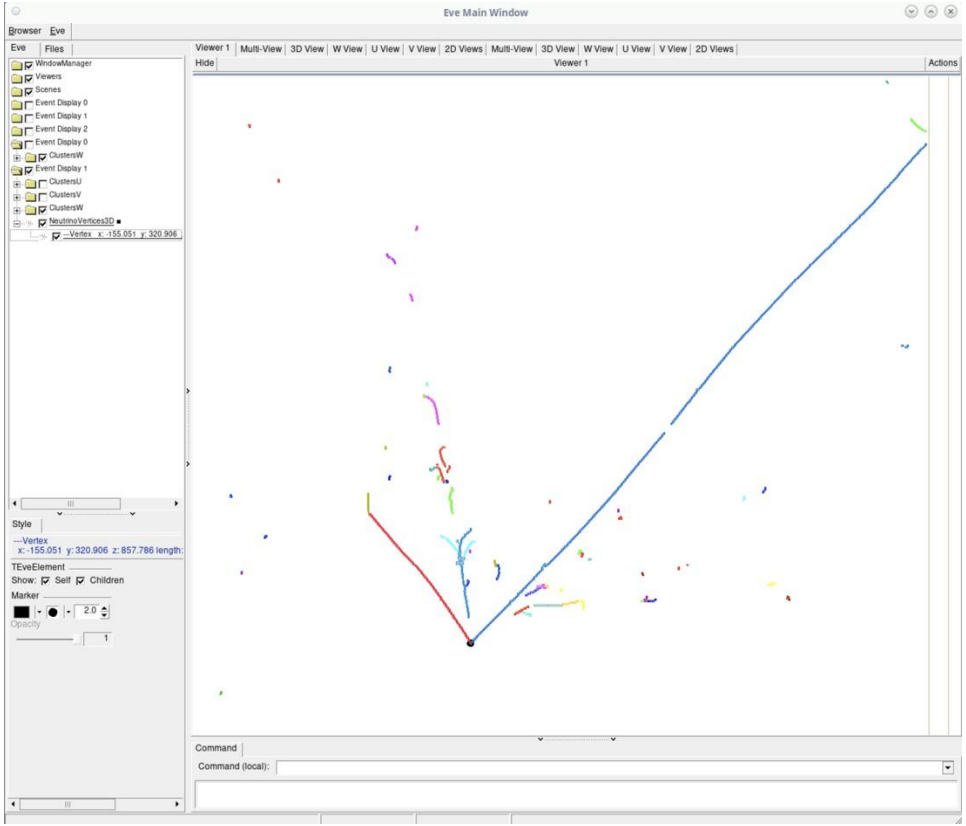
- Now add another instance of the LArVisualMonitoring algorithm, but this time just after the <LArCutClusterCharacterisation> algorithm, shown below
- Here we'll look add clusters in all views, but focus on the W view in the next slide, along with the selected Neutrino interaction vertex (which, being in 3D, only projects onto the W view)
  - The vertex is an important reference point for the reconstruction, so errors here can have knock on effects
  - In DUNE, Pandora employs a deep neural network to determine the vertex location

```
<algorithm type = "LArCutClusterCharacterisation">
    <InputClusterListNames>ClustersU ClustersV
    ClustersW</InputClusterListNames>
    <ZeroMode>true</ZeroMode>
</algorithm>

<algorithm type = "LArVisualMonitoring">
    <ClusterListNames>ClustersU ClustersW ClustersW</ClusterListNames>
    <VertexListNames>NeutrinoVertices3D</VertexListNames>
    <ShowDetector>true</ShowDetector>
</algorithm>
```

- Re-run Pandora to see this new event display (if you haven't removed it, you'll need to skip past the previous event display first)

Understanding the reconstruction (4)

- You'll see more complete clusters as some ambiguities are resolved, but now the reconstructed neutrino interaction vertex as well
  - Pandora renders these vertices in yellow, which can be tricky to spot
  - If you expand the NeutrinoVertices3D node in the tree and select Vertex, you'll see options to change the colour and size of the marker, as we've done here
  - Here the vertex appears well placed
- The track clusters are improved
  - The pion and child proton are separated
  - Note the entire muon is actually reconstructed at this early stage, so it gets broken later



Understanding the reconstruction (5)

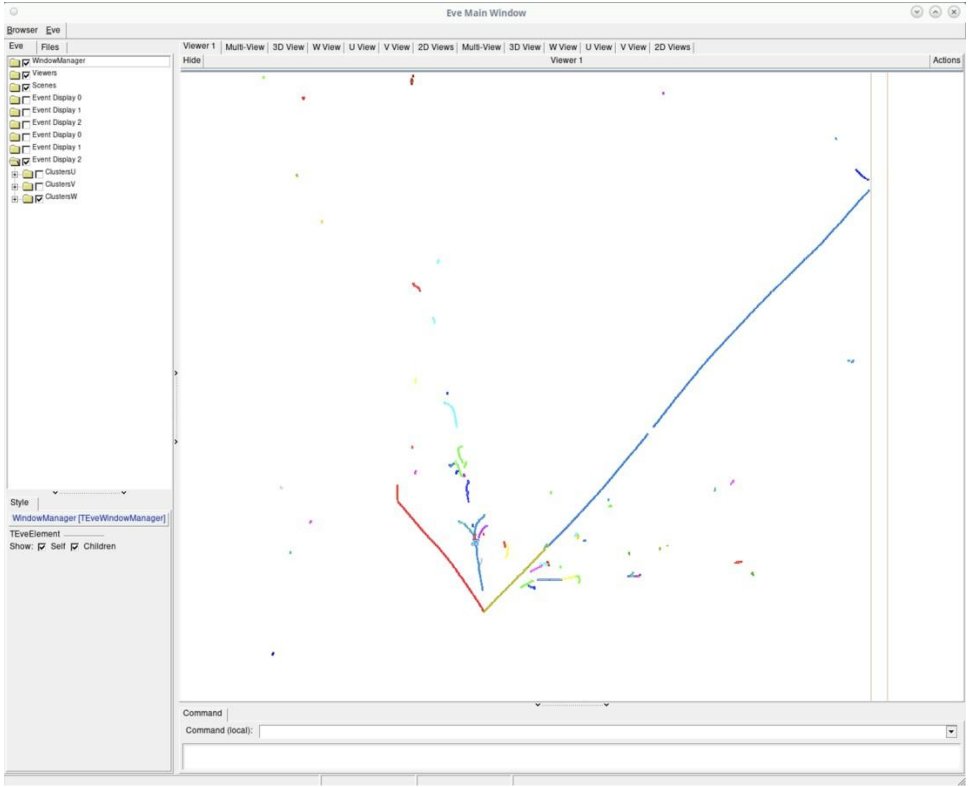
- Next we'll add another instance of the LArVisualMonitoring algorithm, now just after the <!-- ThreeDTrackAlgorithms --> block, shown below
- We'll focus on the W view again here, but all three views are available in the full visualization, because that's typically most useful

```
<!-- ThreeDTrackAlgorithms -->
...
<algorithm type = "LArThreeDLongitudinalTracks">
    <InputClusterListNameU>ClustersU</InputClusterListNameU>
    <InputClusterListNameV>ClustersV</InputClusterListNameV>
    <InputClusterListNameW>ClustersW</InputClusterListNameW>
    <OutputPfoListName>TrackParticles3D</OutputPfoListName>
    <TrackTools>
        <tool type = "LArClearLongitudinalTracks"/>
        <tool type = "LArMatchedEndPoints"/>
    </TrackTools>
</algorithm>

<algorithm type = "LArVisualMonitoring">
    <ClusterListNames>ClustersU ClustersV ClustersW</ClusterListNames>
    <ShowDetector>true</ShowDetector>
</algorithm>
```

- Re-run Pandora to see this new event display

Understanding the reconstruction (6)

- Given the track reconstruction already looked quite complete before this point, we wouldn't expect to see much here
- However, one notable change is that our muon track has been erroneously split by the 3D track algorithms
  - Can you tell why? Hint: Pandora is using information from all three views at this point
- Note: Superficially, it looks like the proton downstream of the pion has been merged, but this is just a coincidence of colour assignment, the particles remain distinct



Understanding the reconstruction (7)

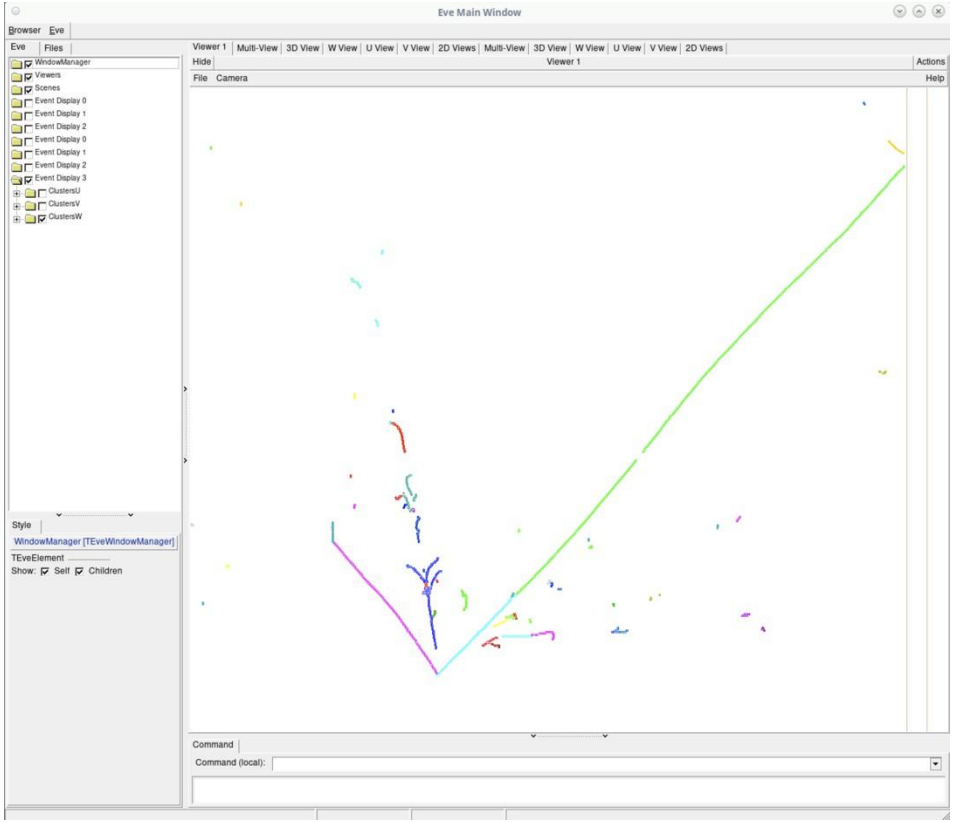
- Next, visualize just after the <!– ThreeDShowerAlgorithms --> block, shown below
- We'll focus on the W view again here, but all three views are available in the full visualization, because that's typically most useful

```
<!-- ThreeDShowerAlgorithms -->
...
<algorithm type = "LArThreeDShowers">
    <InputClusterListNameU>ClustersU</InputClusterListNameU>
    <InputClusterListNameV>ClustersV</InputClusterListNameV>
    <InputClusterListNameW>ClustersW</InputClusterListNameW>
    <OutputPfoListName>ShowerParticles3D</OutputPfoListName>
    <ShowerTools>
        <tool type = "LArClearShowers"/>
        <tool type = "LArSplitShowers"/>
        <tool type = "LArSimpleShowers"/>
    </ShowerTools>
</algorithm>

<algorithm type = "LArVisualMonitoring">
    <ClusterListNames>ClustersU ClustersV ClustersW</ClusterListNames>
    <ShowDetector>true</ShowDetector>
</algorithm>
```

- Re-run Pandora to see this new event display

Understanding the reconstruction (8)

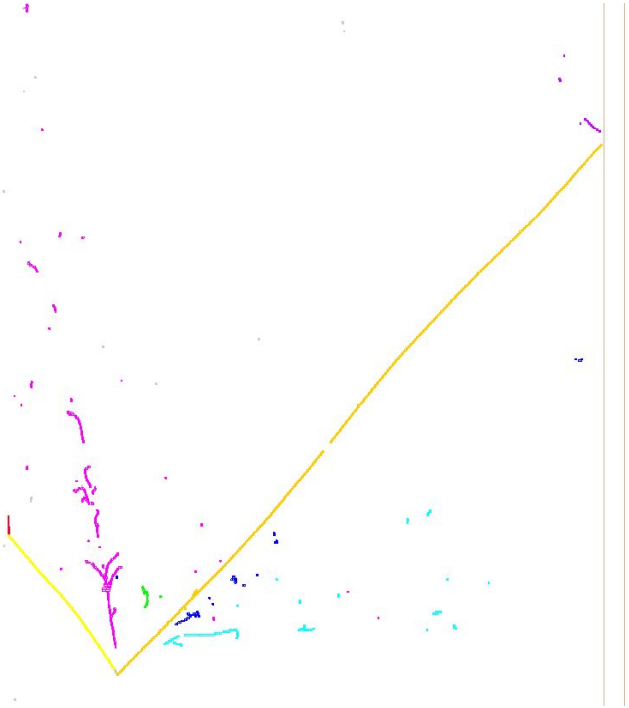
- Here we can see a small amount of merging of the clusters in the largest shower has been undertaken, but fragmentation remains high



Fixing the reconstruction output

The nature of the problem

- You've seen what the final event looks like, and you've seen how the reconstruction moves forward in a few key places
  - Broadly, the tracks are reasonable except where the muon overlaps with one of the shower, but the showers are highly fragmented
- Now you need to see if you can modify the reconstruction algorithm chain to improve the outcome to something closer to the image on the right
- Choose from among the suggestions on the "pointers" slides in the previous section and the guide to those algorithms in the next section to try to improve the reconstruction
  - You may need to undertake some tuning of algorithm parameters for the best results – but don't over-tune!
- If you make progress on this event (don't expect to fix it completely – if you do, please share!), feel free to explore other events in this file and fix those too!



A very brief guide to adding algorithms to the XML (1)

- Let's say you want to add the HitWidthClusterMerging algorithm to the end of the 2D clustering (we'll look at just one of the views here, but remember the initial clustering runs for each view)
- The simplest addition is to add a single line which includes an algorithm with type LArHitWidthClusterMerging, that uses default parameters

```
<!-- TwoDReconstruction -->
...
<algorithm type = "LArClusteringParent">
    <algorithm type = "LArTrackClusterCreation" description =
    "ClusterFormation"/>
    <InputCaloHitListName>CaloHitListW</InputCaloHitListName>
    ...
</algorithm>
...
<algorithm type = "LArTransverseExtension"/>
<algorithm type = "LArHitWidthClusterMerging"/>
```

  - Note that while the class is called HitWidthClusterMergingAlgorithm, the algorithm name is LArHitWidthClusterMerging
  - This is a common mapping of class to algorithm name, but you can find the full set of mappings here, where you can search for the class name to find the corresponding XML algorithm type.
- If you want to modify parameters of the algorithm the update is slightly different…

A very brief guide to adding algorithms to the XML (2)

- Let's say you want to add the HitWidthClusterMerging algorithm, but this time with a custom value for MinClusterSparseness
- Here the update now looks like this

```
<!-- TwoDReconstruction -->
...
<algorithm type = "LArClusteringParent">
    <algorithm type = "LArTrackClusterCreation" description =
    "ClusterFormation"/>
    <InputCaloHitListName>CaloHitListW</InputCaloHitListName>
    ...
</algorithm>
...
<algorithm type = "LArTransverseExtension"/>
<algorithm type = "LArHitWidthClusterMerging">
    <MinClusterSparseness>0.2</MinClusterSparseness>
</algorithm>
```

Note, the closing forward slash is dropped when we expand the block

- Any parameters you want to modify are contained within the algorithm block
- As will be noted in the next section, the collection of modifiable parameters can be found in each algorithm's ReadSettings function

A Brief Guide to the Algorithms of Interest

2D clustering algorithms

LArCrossGapsAssociation

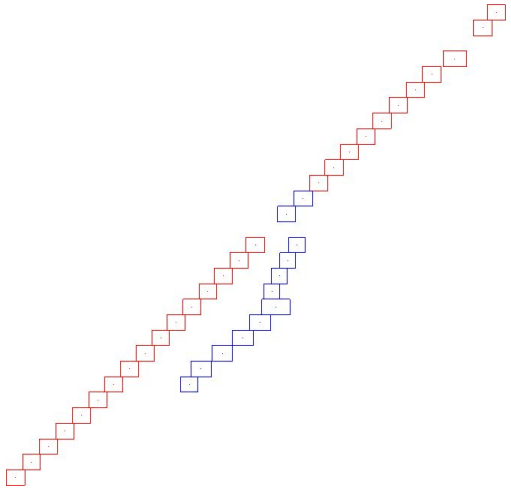
- This algorithm attempts to connect trajectories that may span gaps in the detector readout (e.g. dead channels, or gaps between APAs)
- As with many of the 2D algorithms, the clusters are first filtered according to selection criteria to determine if they should participate in the algorithm
  - A cluster must have a minimum number of hits and layers (you can think of a layer and a wire as synonymous)
  - The selection criteria and other parameters can be tuned in XML. The ReadSettings function is a useful resource
  - The resultant selected clusters are sorted by the starting layer
- Selected clusters are considered pairwise
  - Clusters are checked to see if they point in an approximately common direction and have start and end positions that could be consistent with belong to the same trajectory
  - Trajectories are then projected beyond a cluster in the direction of the target cluster and the target cluster is sampled to see if it is consistent with the projected trajectory
  - If enough points match (within a tolerance) the clusters can be associated
  - Clusters association must be bidirectional, it's not enough for one cluster to point to the other

LArCrossGapsExtension

- This algorithm attempts to connect trajectories that may span gaps in the detector readout (e.g. dead channels, or gaps between APAs), taking a different approach to the previous algorithm
- Clusters are first filtered according to selection criteria
  - A cluster must have a minimum length
  - See the ReadSettings function for more tunable parameters
  - The resultant selected clusters are sorted by the number of hits
  - Clusters are then checked for proximity to a gap
- Selected clusters are considered pairwise
  - In this algorithm, a pointing cluster is created from each cluster, which considers the local trajectory more strongly than the overall cluster trajectory
  - If each pointing cluster is close enough to the position of the target pointing cluster across the gap and the angle across the gap is not too large, the clusters are associated
  - As before, associations must be bidirectional

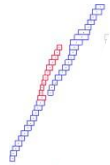
LArOvershootSplitting

- This algorithm attempts to identify cases where hit clustering may have been overzealous, and continued to add hits where an alternative clustering was available
- Clusters are first filtered according to selection criteria
  - A cluster must have a minimum length
  - See the ReadSettings function for more tunable parameters
  - The resultant selected clusters are sorted by the number of hits
- Selected clusters are considered pairwise
  - The algorithm looks for locations where projections of one cluster intersect the other
  - If these distance between the clusters is not too large, the intersection of the projections is determined
  - If this intersection is sufficiently far along a cluster (i.e. we're not too close to a possible, legitimate, vertex), the cluster is considered for splitting



LArBranchSplitting

- This algorithm attempts to identify instances where the continuation and branching of two clusters may have been wrong
- Clusters are first filtered according to selection criteria
  - A cluster must have a minimum length
  - See the ReadSettings function for more tunable parameters
- Selected clusters are considered pairwise
  - The algorithm looks for locations where one cluster appears to emerge from another
  - The angle shouldn't be too big (i.e. the clusters propagate in similar directions
  - If a redistribution of the hits leads to more consistent trajectories, the clusters are split



LArKinkSplitting

- As the name suggests, this algorithm attempts to identify sharp direction changes within a cluster that may indicate a need to split the cluster
- Clusters are first filtered according to selection criteria
  - A cluster must have a minimum length
  - See the ReadSettings function for more tunable parameters
- Selected clusters are considered individually
  - The algorithm performs sliding fits along the cluster trajectory
  - The angle between fits that cover adjacent sliding windows is determined
  - The maximum angle across all fit comparisons is identified
  - If the angle is too large, the cluster is split at the midpoint of relevant fits

LArTrackConsolidation

- This algorithm attempts to identify instances where shower-like clusters have stolen hits from track-like clusters
- Clusters are first filtered according to selection criteria
  - A track cluster must have a minimum length
  - A shower cluster must have a maximum length
  - See the ReadSettings function for more tunable parameters
- Selected clusters are considered pairwise
  - The algorithm checks to ensure the shower-like cluster is at most half the size of the track-like cluster
  - The algorithm then looks for shower hits that appear to fill gaps in the track-like trajectory
  - If such hits are found, they are removed from the shower-like cluster and added to the track-like cluster

LArHitWidthClusterMerging

- This algorithm attempts to merge clusters involving wide hits
  - Wide hits are liable to occur when many drift electrons arrive at a small number of wires in quick succession
  - This produces a very wide signal from which a small number of hits are created
  - Such hits can be difficult to merge because the infinitesimal position of hits can be widely separated
- Clusters are first filtered according to selection criteria
  - A cluster must have a sufficiently large average hit width
  - See the ReadSettings function for more tunable parameters
- Selected clusters are considered pairwise
  - The algorithm looks for a potential merge point for the clusters (not necessarily end-to-end as wide hits can lead to overlap)
  - The individual clusters are checked for a sufficiently similar direction
  - The direction of a potential merged cluster is also compared to those original directions to ensure it does not change too much
  - If everything is consistent, the clusters are merged

LArBoundedClusterMopUp

- This algorithm attempts to grow (by default) shower clusters
- Clusters are first filtered according to selection criteria
  - A cluster must have a minimum length
  - See the ReadSettings* function for more tunable parameters
- Selected clusters are considered pairwise
  - The algorithm finds particles produced by inter-plane matching and extracts their clusters (particle clusters)
  - The algorithm identifies clusters that did not produce particles by inter-plane matching (remnant clusters)
  - The edges of the shower are identified for the particle clusters by constructing bins along a track-like fit of the shower that span the perpendicular extent of the shower at that point
  - If a sufficiently large fraction of the hits in remnant clusters are bounded by the particle cluster edges, the clusters are merged

* Most algorithms in Pandora derive directly from the Algorithm class, but some, like this one, inherit from other classes. In this case, it can be useful to check if this base class has its own parameters (which you can then override in this algorithm's XML paramet ers), which are set in its own ReadSettings function. To see what extra parameters are available, you can look at its corresponding header file and see which class it inherits from. You can find the location of the corresponding base class by searching for the class in the LArContent class includes list.

LArConeClusterMopUp

- This algorithm attempts to grow (by default) shower clusters
  - See the ReadSettings function for more tunable parameters
- Selected clusters are considered pairwise
  - The algorithm finds particles produced by inter-plane matching and extracts their clusters (particle clusters)
  - The algorithm identifies clusters that did not produce particles by inter-plane matching (remnant clusters)
  - A fit is performed through the hits in the particle cluster
  - Walking along that fit, the positive maximally transverse coordinate are collected and a fit performed
  - Walking along that fit, the negative maximally transverse coordinate are collected and a fit performed
  - These two “edges” about the overall fit define a search cone
  - If a sufficiently large fraction of the hits in remnant clusters are bounded by the particle cluster edges, the clusters are merged

LArNearbyClusterMopUp

- This algorithm attempts to grow (by default) shower clusters
  - See the ReadSettings function for more tunable parameters
- Selected clusters are considered pairwise
  - The algorithm finds particles produced by inter-plane matching and extracts their clusters (particle clusters)
  - The algorithm identifies clusters that did not produce particles by inter-plane matching (remnant clusters)
  - The remnant clusters must have a minimum number of hits
  - If a remnant cluster is sufficiently close to a particle cluster, it can be merged

3D track algorithms

LArThreeDTrackFragments

- This algorithm attempts to create particles in instances in which we have two well-reconstructed views, and a third fragmented view
  - See the ReadSettings function for more tunable parameters
- The algorithm ONLY considers clusters that are not owned by an existing particle
- It identifies pairs of clusters (in different views) with significant overlap in the drift coordinate (and thus likely to belong to the same particle)
- Starting from the ‘best pair’:
  - The clusters are used to create projected positions in the ‘missing’ view
  - For each projected position, the closest ‘available’ hit is sought
  - The clusters to which these hits belonged are together postulated to for the third cluster of the particle
  - If the, now, three-cluster match presents a consistent 3D image a particle is created

3D recovery algorithms

LArVertexBasedPfoRecovery

- This algorithm attempts to recover any ‘tiny’ particles coming out of the neutrino vertex e.g. tiny proton stubs
  - See the ReadSettings function for more tunable parameters
- It first identifies all ‘available’ clusters ‘close-to’ the neutrino vertex
- It then attempts to ‘match’ found clusters across the views – to do this we refer to the overlap of the clusters in the drift coordinate and a ‘pseudo chi-squared’ metric that measures the extent to which a consistent 3D image can be formed
- First, three cluster matches are sought and particles are created in order of the best ‘chi-squared’
- Next, two cluster matches are sought, and again particle creation is prioritised by the ‘chi-squared’ metric

LArParticleRecovery

- This algorithm attempts to reconstructed hitherto ‘missed’ particles
  - See the ReadSettings function for more tunable parameters
- First, it identifies all ‘available’ clusters above a user defined ‘size’
- If the algorithm is running in ‘vertex mode’, tiny stub-like clusters originating from clusters coming out of the vertex are also identified
- Pairs of matches across the three views are sought (a cluster can exist in multiple pairs) – to do this we refer to the overlap of the clusters in the drift coordinate
- These pairs are then examined to identify groups of clusters that exist within the same drift region
- If the group corresponds to a triplet or doublet of clusters (each belonging to a different view) a particle is created

3D mop up algorithms

LArSlidingConePfoMopUp

- This algorithm attempts to optimise shower completeness by merging shower fragments into their parent shower
  - See the ReadSettings function for more tunable parameters
- First, all 3D showers and ‘small’ tracks are identified (these are the only particles considered in the algorithm)
- For each ‘large’ 3D shower:
  - we walk along it’s central axis, drawing a 3D cone at each point with an opening angle and length that increases with distance along the shower
  - any particles that are ‘significantly’ contained within a cone are recorded, alongside their potential parent
- After this process, a particle may have multiple potential parents, and ambiguities are resolved by picking the parent that most contained the particle
- Merges are then performed, and recursive merging is allowed (if running in vertex mode, which is the default, particles in ‘close’ proximity to the neutrino vertex cannot be merged)
- This whole process runs recursively, until no more merges can be made

LArSlidingConeClusterMopUp

- This algorithm attempts to optimise shower completeness by merging ‘lost’ 2D clusters into their parent shower
  - See the ReadSettings function for more tunable parameters
- ‘Significant’ 3D showers and ‘small’ 3D tracks are collected and are the seed particles the algorithm will look to grow
- ‘Small’ 2D clusters, that do not yet belong to any particles, are identified and are the clusters the algorithm will use to grow the showers with
- For each seed particle:
  - we walk along it’s central axis, drawing a 3D cone at each point
  - the cone is projected in each 2D view, and any clusters that are ‘significantly’ contained within a cone are recorded, alongside their potential parent
- After this process, a particle may have multiple potential parents, and ambiguities are resolved by picking the parent that most contained the particle
- Merges are then performed, and recursive merging is allowed

LArIsolatedClusterMopUp

- This algorithm is one of the final algorithms to run in Pandora
- Its job is to make sure that as many of the input hits as is possible are included in the reconstruction output

  ⇒ limiting the missing energy in downstream energy estimations

  - See the ReadSettings function for more tunable parameters
- It first finds the ‘seed’ 3D particles to grow – this is configurable and is either all shower particles (default) or all showers and tracks
- It then identifies all remaining ‘small’ 2D clusters and ‘dissolves’ them into their 2D hits
- Each, now free, 2D hit is added to the closest cluster – **only if** the cluster is within some proximity
- The hits are added as ‘isolated hits’
- Isolated hits **do not** contribute to trajectory fits but **do** contribute to energy calculations