Pandora dev for DUNE FD

Andy Chappell

17/09/2020

DUNE FD Sim/Reco Workshop

DEEP UNDERGROUND NEUTRINO EXPERIMENT

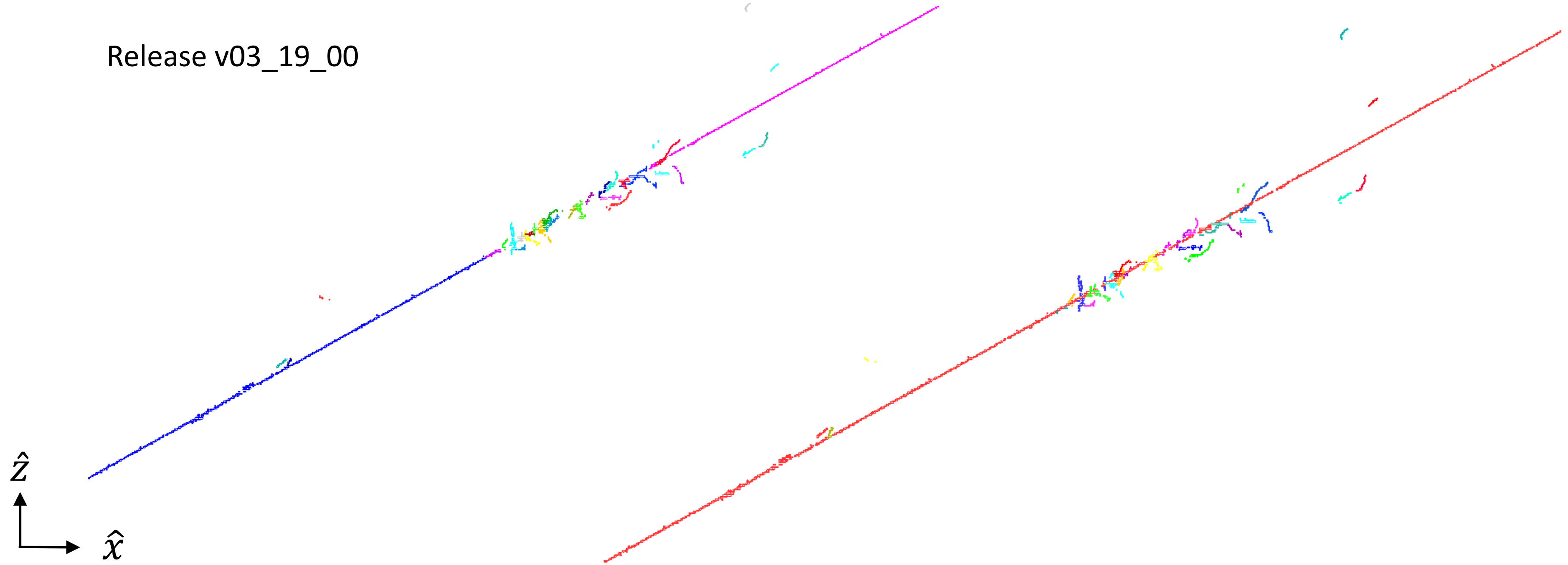
WARWICK

DUNEFD development work overview

WARWICK

- Identifying tracks split by EM showers (release v03_19_00)
  - Further refinements to deal with close proximity to a TPC boundary (in progress)
- Performance enhancements
  - Cluster variable caching (release v03_19_03)
  - Concave hull identification (in progress)
- PFO level track/shower characterisation BDT (under review)
- Machine learning
  - Hit-level track/shower classification network (v1 under review)
  - Vertex identification network (in progress)
- External clustering and vertexing algorithm support (under review)

Tracks through EM showers



- The TrackInEMShower algorithm is a clustering algorithm that identifies tracks which have been split by large amounts of EM behavior, it refines the endpoints of these tracks and merges them together alongside hits that are collected in the shower region

Tracks through EM showers

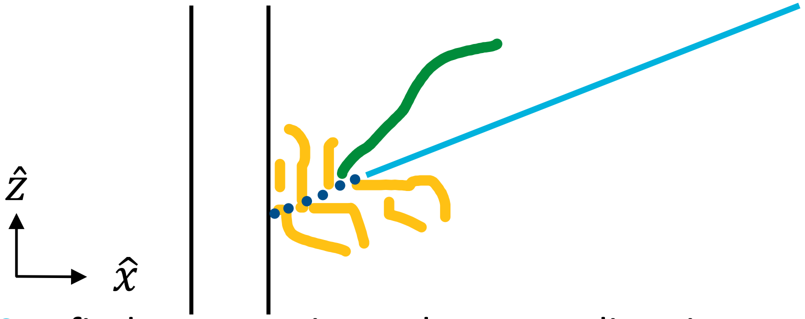
- Previously identified two remaining issues:
  - Clustering follows a delta ray/shower branch near the TPC boundary
  - Track showers at the TPC boundary
- Results in stitching failures in both cases
- In each case, want to:
  1. Remove any clustering errors
  2. Extend the track if appropriate
- Builds on TrackInEMShower algorithm with new algorithms to refine endpoints

I. Mawby

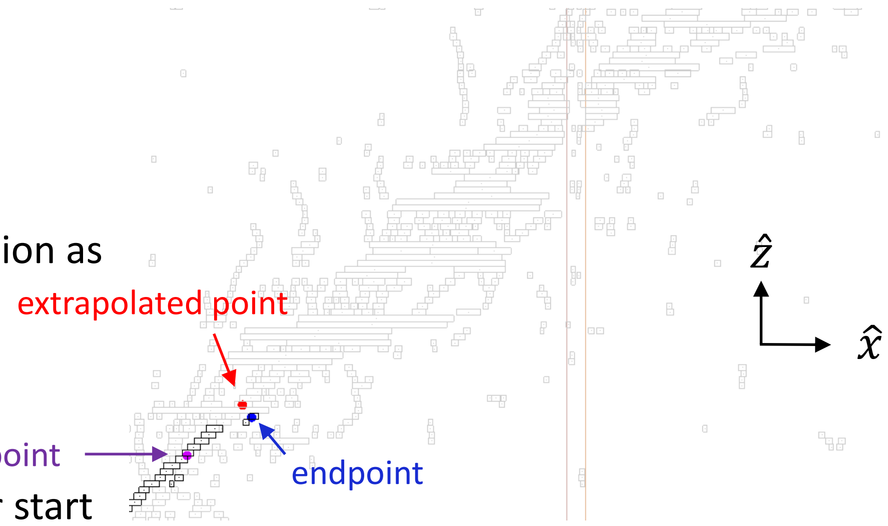
Tracks through EM showers

Refining cluster endpoints and extending to TPC boundaries given a list of track-like clusters:

1. Finds the best cluster endpoint to fix
    1. start with endpoint of cluster whose other endpoint is farthest from boundary



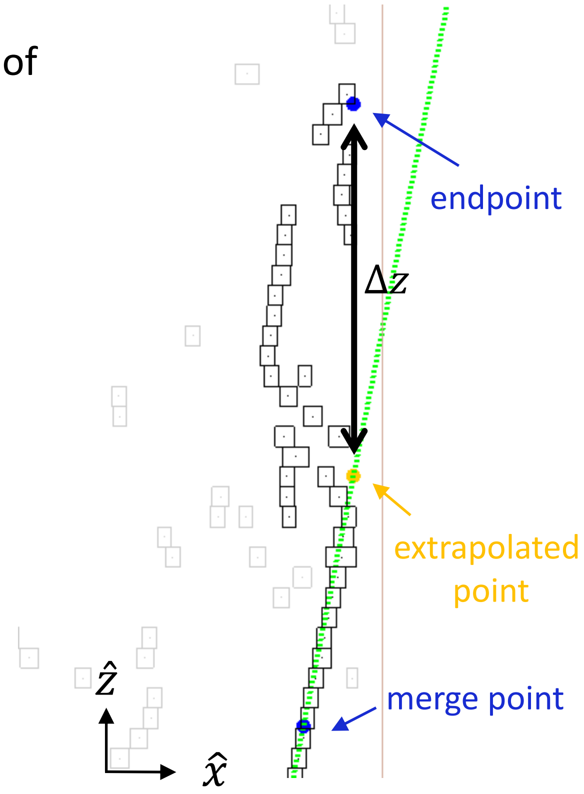
    2. find merge point and average direction
    3. reject if no x component in average direction as this suggests no boundary crossing.
    4. find distance between endpoint and merge point, extrapolate direction to get extrapolated position
    5. use impact parameters to identify shower start



Tracks past delta rays

Refining cluster endpoints and extending to TPC boundaries given a list of track-like clusters:

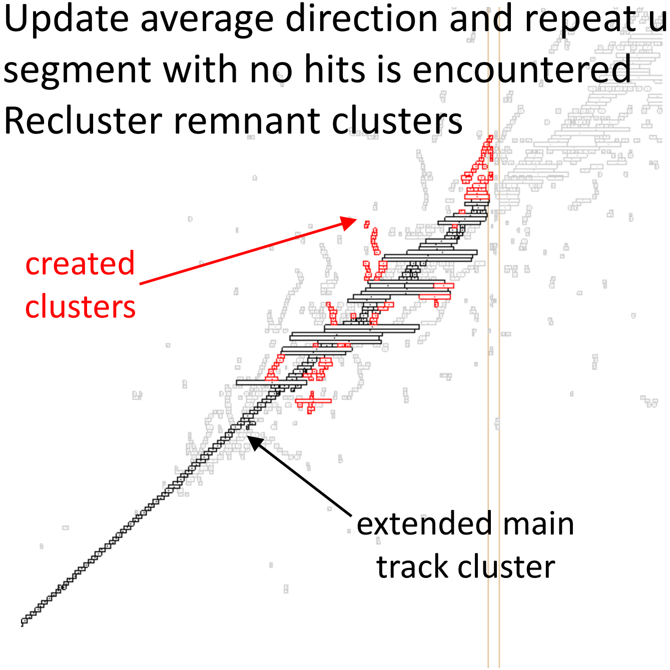
1. Finds the best cluster endpoint to fix
    1. sufficient separation between merge point and endpoint
    2. merge and extrapolated points close to boundary
    3. curvature beyond merge point indicative of delta ray
2. Cluster farthest from TPC boundary and meeting these requirements is best cluster

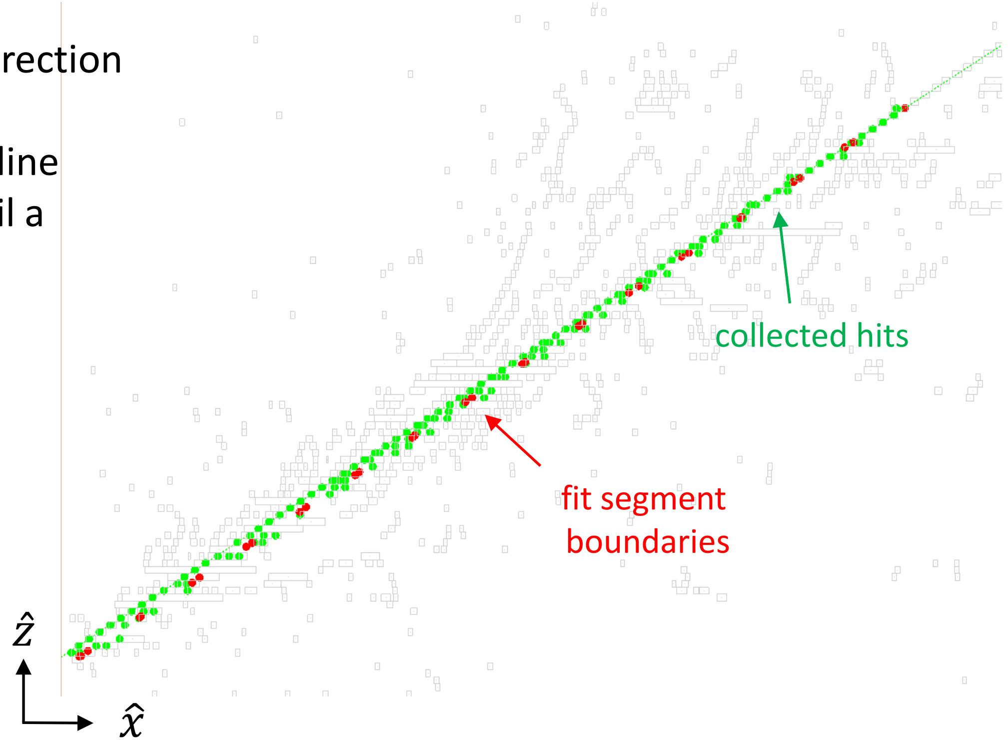


Identify extrapolated hits

From the merge point, extrapolate the average direction forward 5cm:

1. Collect any hits in close proximity to this line
2. Update average direction and repeat until a segment with no hits is encountered
3. Recluster remnant clusters





Performance

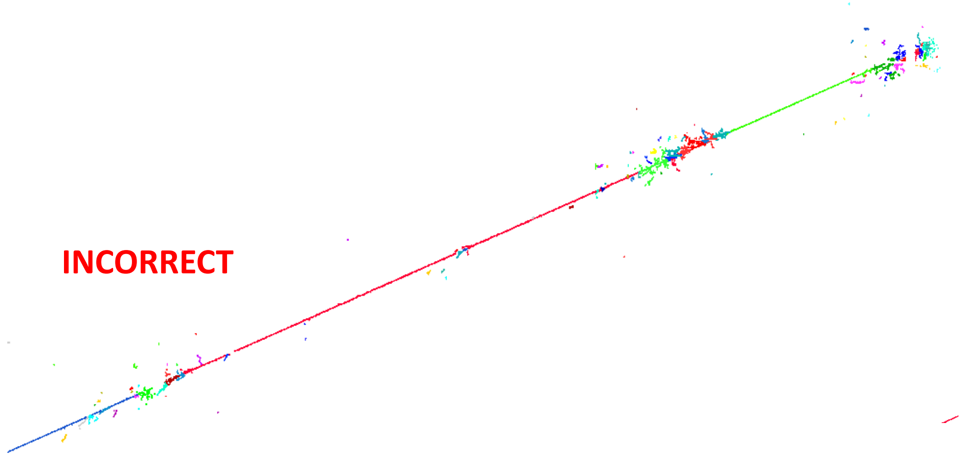
WARWICK

Correct event fraction = $\frac{\text{number of correctly reconstructed cosmic ray muons}}{\text{number of reconstructable MC cosmic ray muons}}$

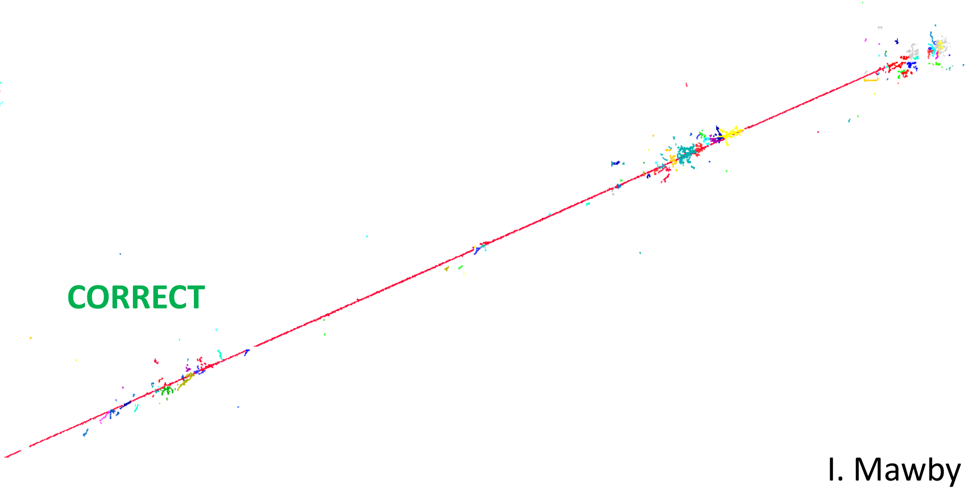
Average correct event fraction:

- Standard + Stitching + HW + TrackInEMShower: 93.8%
- Standard + Stitching + HW + TrackInEMShower + ExtensionPastDeltaRay + ExtensionThroughShower: ~95%

Potential for further improvement by considering hit widths in these updates



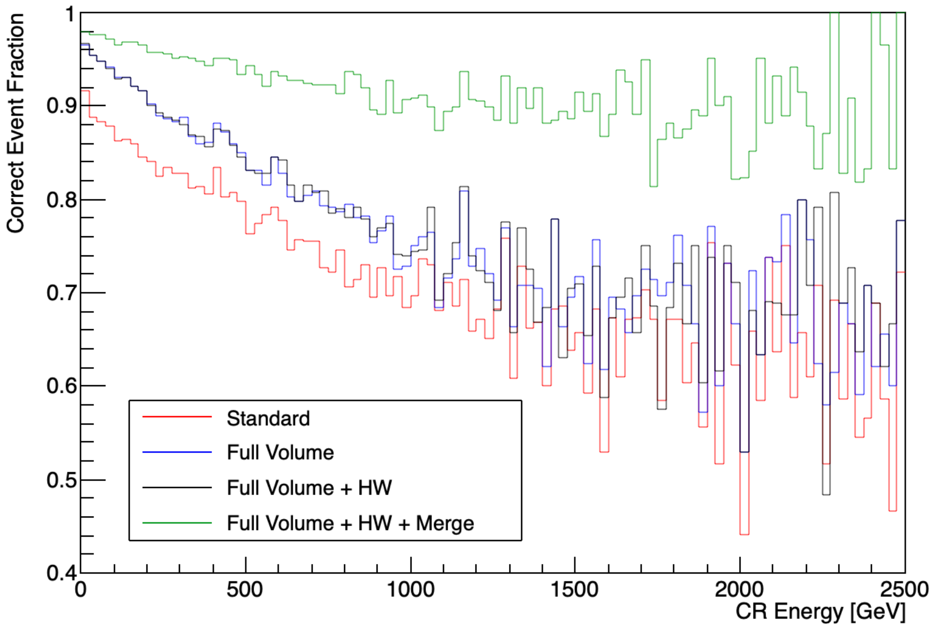
INCORRECT

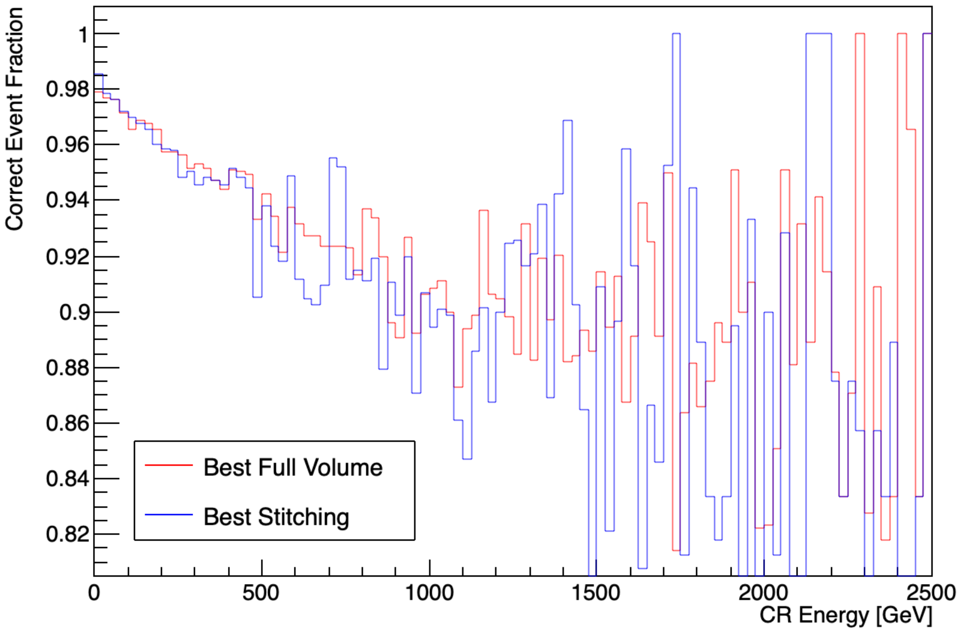


CORRECT

Alternative approach

- DUNE FD cosmics have no time offset, so can try turning off the stitching so that all hits are reconstructed together (Full Volume)
  - This works quite well (Best Stitching refers to the approach described in the preceding slides)





pandoraCosmic reco

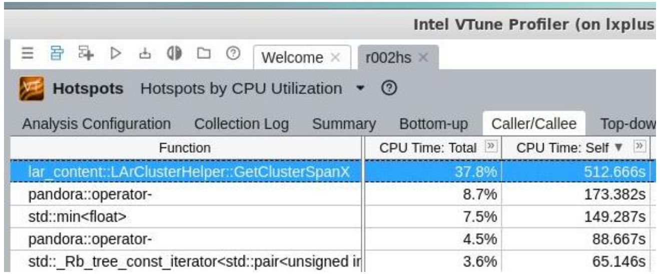
WARWICK

- To ensure that anyone looking at cosmics in DUNE FD benefits from the recent updates, a named fcl alias for the producer module, with the official pandoraCosmic for DUNE FD configuration, will be produced
- To be added to pandoramodules_dune.fcl by the end of the month, in time for the calibration group's production

Performance improvements

WARWICK

- Used Intel's VTune tools to identify bottlenecks in the Pandora reconstruction
  - Examples here are in the ProtoDUNE-SP context, but the DUNEFD context is similar
- GetClusterSpanX consumes a large fraction of the total reconstruction runtime:



Function	CPU Time: Total	CPU Time: Self ▼
lar_content::LArClusterHelper::GetClusterSpanX	37.8%	512.666s
pandora::operator-	8.7%	173.382s
std::min<float>	7.5%	149.287s
pandora::operator-	4.5%	88.667s
std::_Rb_tree_const_iterator<std::pair<unsigned ir	3.6%	65.146s

- As an example, the ThreeViewMatchingControl has a deeply nested loop calling ThreeViewRemnants::CalculateOverlapResult

```
for (const Cluster *const pClusterU : clusterVectorU)
{
    for (const Cluster *const pClusterV : clusterVectorV)
    {
        for (const Cluster *const pClusterW : clusterVectorW)
            m_pAlgorithm->CalculateOverlapResult(pClusterU, pClusterV, pClusterW);
    }
}
```

```
void ThreeViewRemnantsAlgorithm::CalculateOverlapResult(const Cluster *const pClust
{
    // Requirements on X matching
    float xMinU(0.f), xMinV(0.f), xMinW(0.f), xMaxU(0.f), xMaxV(0.f), xMaxW(0.f);
    LArClusterHelper::GetClusterSpanX(pClusterU, xMinU, xMaxU);
    LArClusterHelper::GetClusterSpanX(pClusterV, xMinV, xMaxV);
    LArClusterHelper::GetClusterSpanX(pClusterW, xMinW, xMaxW);
```

M. Brunetti & A. Chappell

Performance improvements

WARWICK

- A single cluster has its X span calculated many times, when once should be enough
- Updated function to cache the result, so future calls can just look it up
  - Some care is needed, certain functions in Cluster invalidate the span, so need to ensure value is recalculated when this happens
- End result is a 25% speed up for ProtoDUNE-SP and 15% for DUNEFD

Hotspots Hotspots by CPU Utilization

Analysis Configuration Collection Log Summary Bottom-up Caller/Callee T

Function	CPU Time: Total	CPU Time: Self ▼
pandora::operator-	12.4%	134.265s
pandora::operator-	6.9%	74.505s
std::unordered_set<pandora::Cluster const	5.2%	56.109s
operator new	4.9%	53.631s
lar_content::LArClusterHelper::GetClosestF	23.2%	50.081s
lar_content::LArClusterHelper::GetClosestF	17.6%	46.109s
std::_Rb_tree_const_iterator<std::pair<unsi	4.4%	45.482s
lar_content::TwoDSlidingFitResult::GetTran	5.1%	42.924s
std::unordered_map<pandora::Cluster cons	3.6%	39.229s

Analysis Configuration Collection Log Summary Bottom-up Caller/Callee T

Function	CPU Time: Total	CPU Time: Self ▼
lar_content::ThreeViewTrackFragmentsAlg	1.3%	6.869s
std::_Rb_tree_const_iterator<std::pair<unsi	0.6%	6.321s
lar_content::OverlapTensor<float>::GetCon	1.2%	6.290s
pandora::CartesianVector::GetUnitVector	1.3%	5.684s
pandora::operator*	0.5%	5.671s
lar_content::LArGeometryHelper::MergeThr	1.0%	5.510s
pandora::Cluster::GetClusterSpanX	0.5%	4.890s
lar_content::NViewTrackMatchingAlgorithm	0.5%	4.888s
std::sqrt	0.4%	4.719s

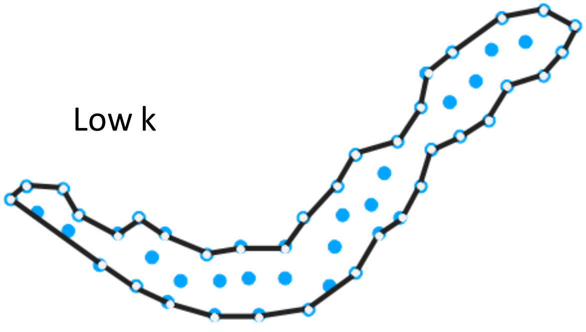
- Available by default as of larpandoracontent v03_19_03

Performance improvements

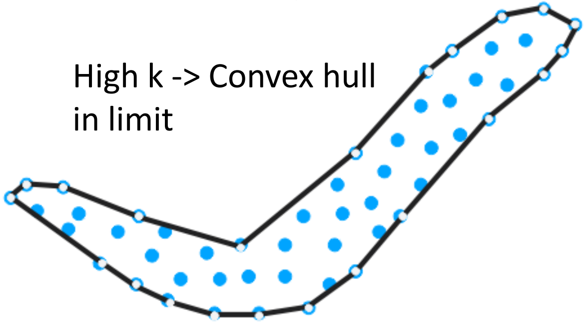
- Having eliminated the cluster span overhead the next bottleneck concerns cluster proximity detection
- A number of algorithms look to determine the closest approach between clusters when considering merges
    - This involves checking the proximity between many hits
    - Can the number of hits considered be reduced while retaining structural information?
    - Determine the concave hull of a cluster

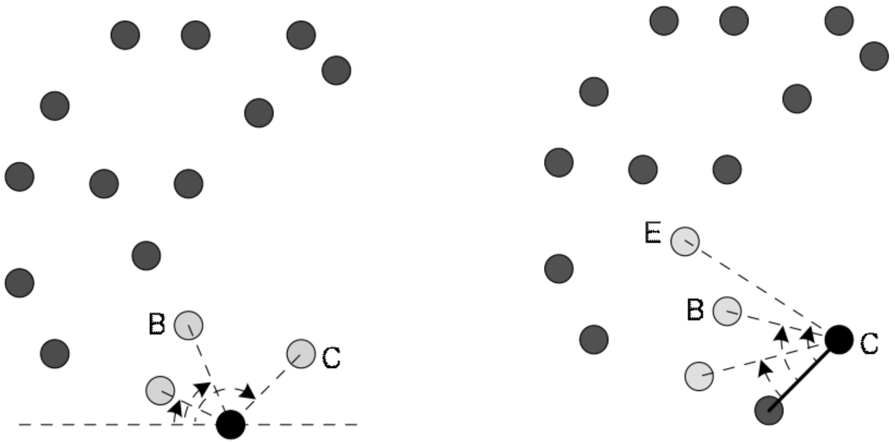
Concave hull idea

Low k



High k -> Convex hull in limit

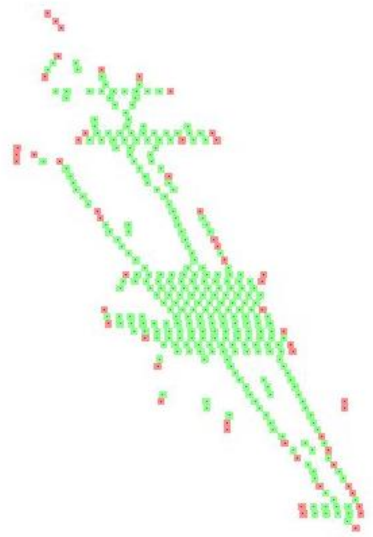




Implement K nearest neighbours algorithm developed by Moreira & Santos

Concave hull implementation

- Very preliminary – currently just checking that the hull looks sensible
- Example ProtoDUNE-SP cluster shown
  - Green points are the hits from a single cluster
  - Red points represent the vertices of the concave hull wrapping the cluster
- Little point in computing the hull for small clusters (<10 hits)
- For large clusters hit count reduced considerably while retaining representative structure
- Next step, determine if the overhead in computing the hull (once per cluster) offsets per hit pair (many per cluster) distance computations



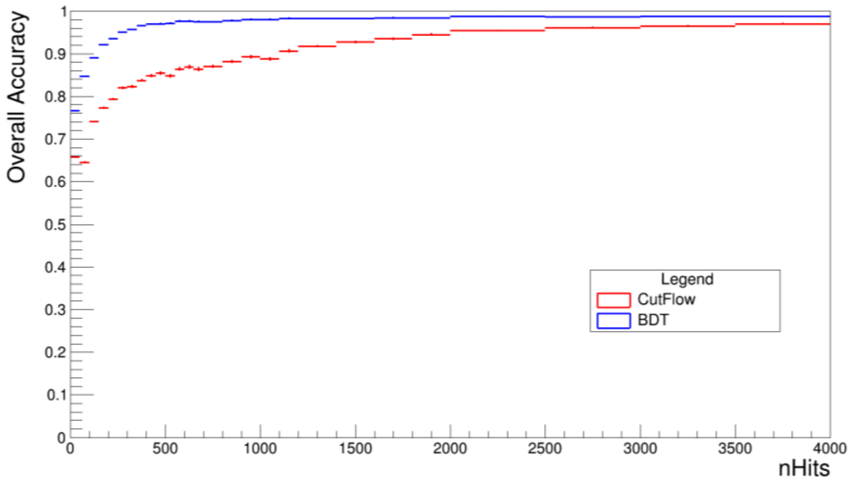
PFO-level track/shower classification

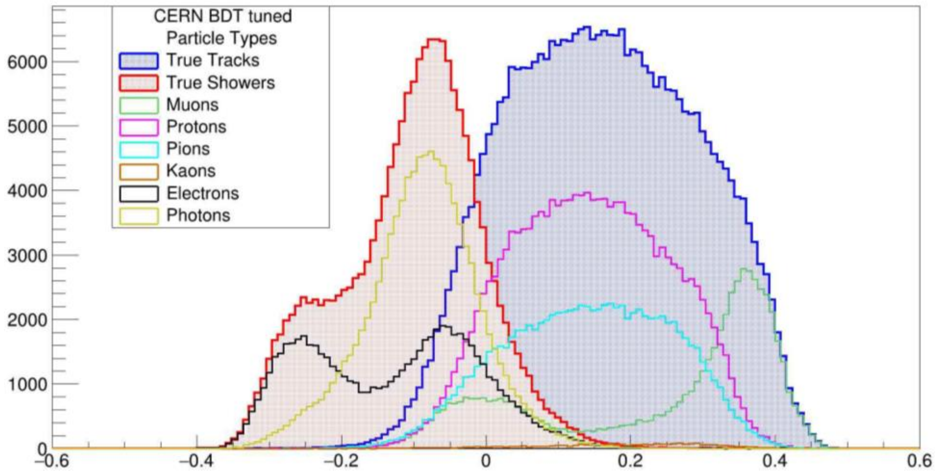
- BDT with 13 topological, calorimetric, and hierarchical variables
- Studied hyper-parameters to determine best values
- Definitions:

		True Conditions	
	Total PFOs	True Tracks (TT)	True Showers (TS)
Predicted Conditions	Predicted Tracks (PT)	True Positive (TP)	False Positive (FP)
	Predicted Showers (PS)	False Negative (FN)	True Negative (TN)

  - Sensitivity/True Positive Rate (TPR) = TP/TT
  - Specificity/True Negative Rate (TNR) = TN/TS
  - Accuracy = $(TP+TN)/(Total\ PFOs)$
- Maximize accuracy

PFO-level track/shower classification



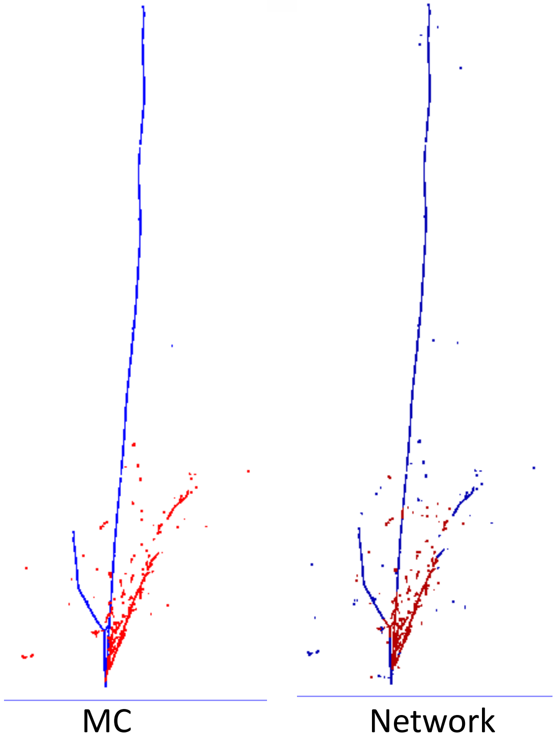


	Sensitivity	Specificity	Accuracy	#PFOs
CutFlow	0.6809	0.9197	0.7638	~461,000
CERN BDT TUNED	0.9268	0.8223	0.8906	~461,000

M. Rai

Hit-level track/shower classification

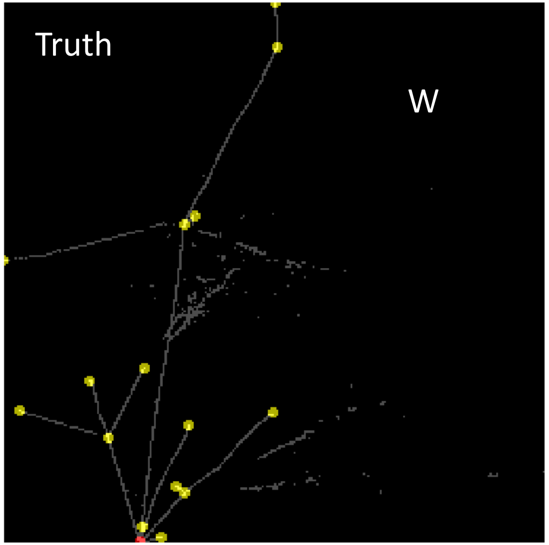
- Pandora library with LibTorch support currently undergoing review pending release
- Hit-level track/shower classification network will be released alongside this update
  - New algorithm will not run by default due to ongoing investigation into computational performance of the LibTorch UPS product
- Alongside LArContent (larpandoracontent in LArSoft context) there is now a LArDLContent (larpandoradlcontent) library
- LArContent handles all of the standard algorithms and has no knowledge of deep learning libraries
- LArDLContent has access to LibTorch features and LArContent algorithms



A. Chappell

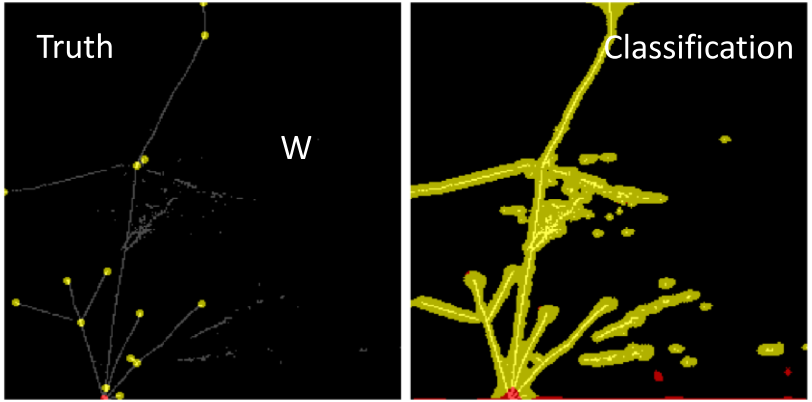
Vertexing network

- Alternative route to track/shower classification
  - Identify vertices and form skeleton of event topology
  - Use skeleton as basis for track/shower cluster identification
- Reuses semantic segmentation network for hit-level track/shower identification
- Ground truth is a 256x256 pixel image composed of MC particle endpoints projected onto respective views
  - Primary vertex is one class
  - All other vertices form a second class
- Small scale training set so far, 1000 events from MCC 11 FD 1x2x6 nu

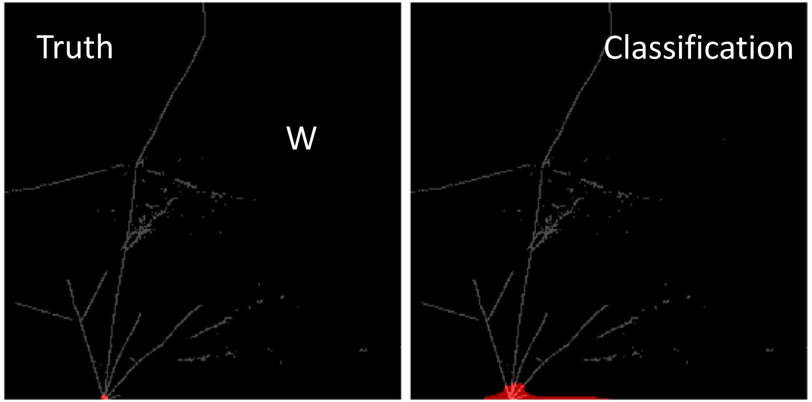


Vertexing network

- Truth identifies primary vertex and "other" vertices
  - This was a little too ambitious to start off



- Scaled back to primary only as proof of concept
  - At least some region finding capability from small training run
- Introduce endpoints of primaries in next stage



A. Chappell

External clustering and vertexing

WARWICK

- Updates to LArPandora to provide more flexibility in interacting with external producers
  - https://github.com/AndyChappell/larpandora/tree/feature/pandora_art_io
- Added algorithms to read vertices and clusters identified by external producers
  - LArPandoraInterface/ExternalVertexingAlgorithm.h
  - LArPandoraInterface/ExternalClusteringAlgorithm.h

Summary

- Recent releases
  - TrackInEMShower (LArContent v03_19_00)
  - Performance improvements (LArContent v03_19_03)
- In review
  - PFO-level track/shower characterisation BDT
  - Deep learning support
  - Further refinements to TrackInEMShower
  - External clustering and vertexing algorithm support
- In progress
  - Concave hull identification
  - Vertex identification network

Backup

Pandora correct event fraction definition

WARWICK

- To be tagged as correctly reconstructed, each primary particle in the event (in this case a single cosmic ray muon) needs to be matched to a single primary PFO that is above threshold
  - The PFO needs to have at least 5 shared hits, a completeness above 10% and a purity above 50%.
- Matches are determined by first matching each primary MC particle to its best primary PFO and then matching the remaining primary PFOs to the primary MC particles.
- For the MC particle to be deemed reconstructable it needs to have at least 5 hits in at least two views and at least 15 hits overall.
- And in this case the hierarchy is folded back. So the correct event fraction is for the cosmic ray muon and child showers i.e the hierarchy.