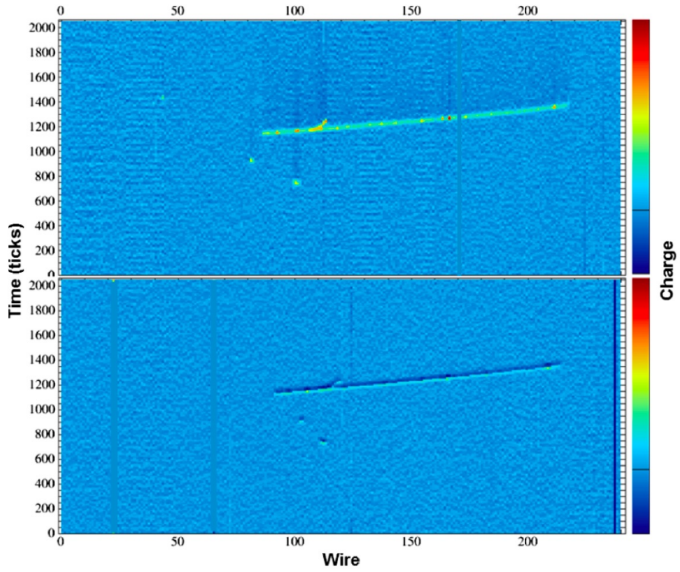
Blip reconstruction tools in the MicroBooNE LArTPC

PNS WG Meeting
June 7, 2023

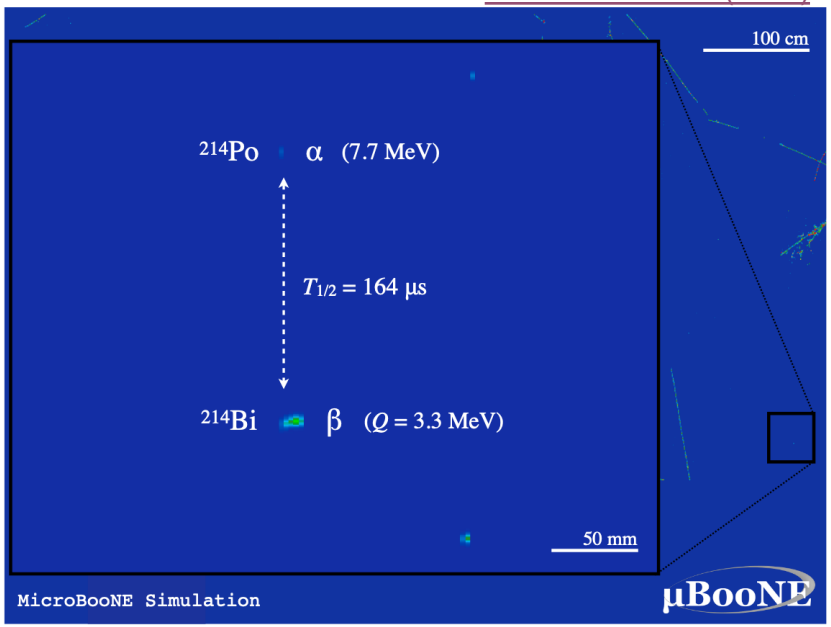
Will Foreman
Illinois Institute of Technology

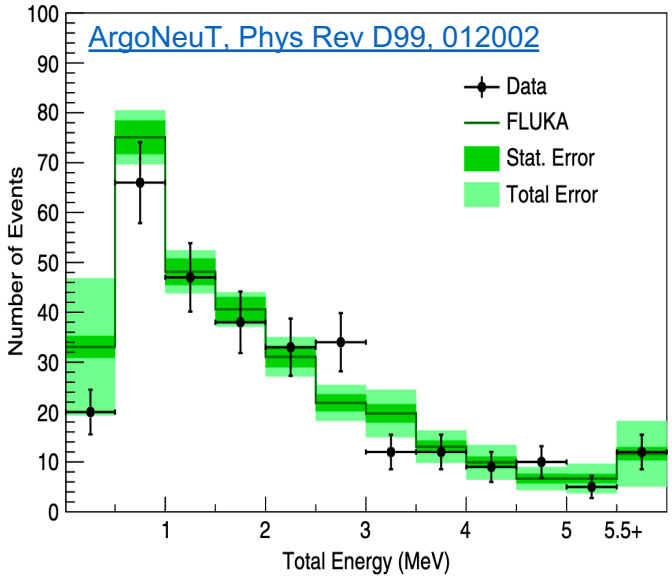
A little history

- ArgoNeuT: pioneered MeV-scale reconstruction in LArTPCs by measuring de-excitation γ's from ν_μCC in the NuMI beamline
- MicroBooNE: same reconstruction technique was employed to look at ^{222}Rn daughter decays (^{214}Bi → ^{214}Po) from special R&D
- See theses by Ivan Lepetic (link) and Avinay Bhat (link)



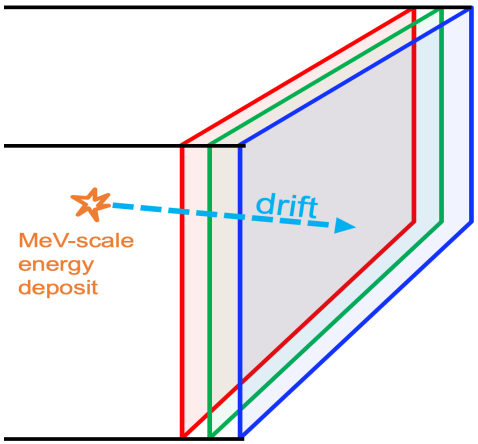
JINST 17 P11022 (2022)

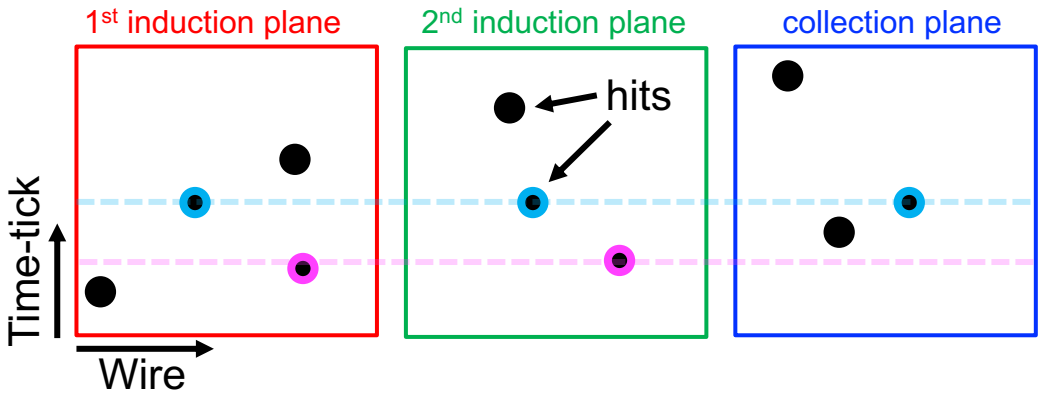


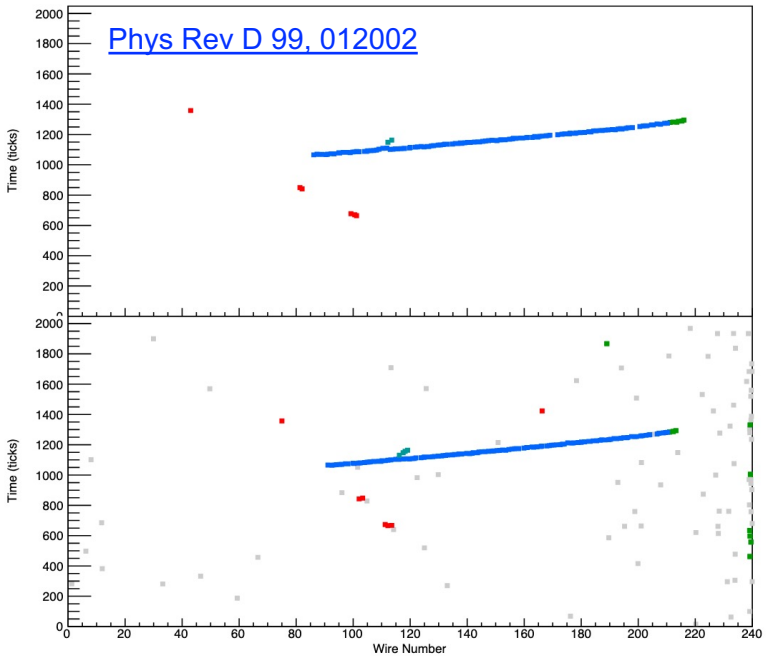


ILLINOIS INSTITUTE OF TECHNOLOGY

Blip reconstruction in a nut-shell

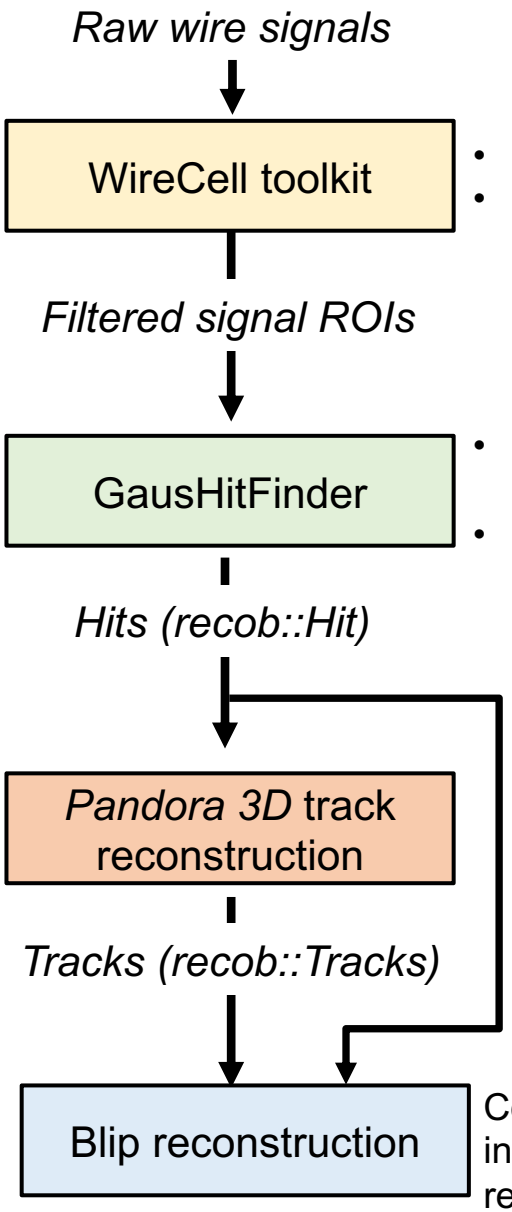






- Signals time-matched between wire readout planes
  - Wire intersections → YZ coordinate
- Easy with extended (multi-hit) signals
- More challenging at lower energy
  - Hit-finding thresholds
  - Noise hits create ambiguous fake matches

Reconstruction workflow in MicroBooNE



- De-convolution
- ROI-finding

- Search ROIs for ADC above threshold
- Fit to Gaussians

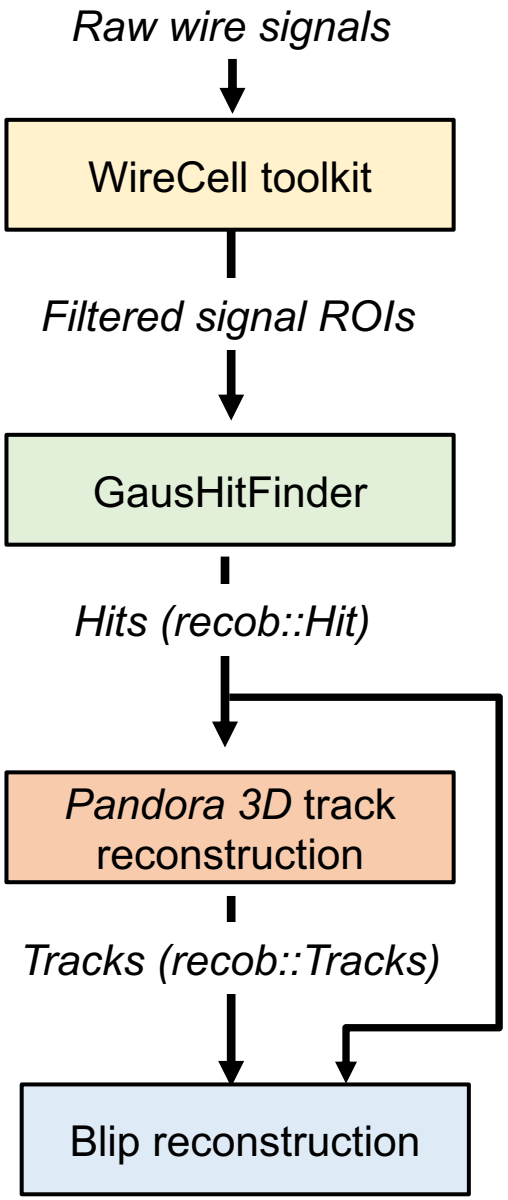
Combines Hit and Track information to identify and reconstruct MeV-scale signatures

Energy threshold influenced by a confluence of WireCell and GausHitFinder settings

	Parameter	Standard reco	"Low threshold"
WireCell	troi_ind_th_factor	3	3
	troi_col_th_factor	5	5
	r_fake_signal_low_th	500	1
	r_fake_signal_high_th	1000	1
	r_fake_signal_low_th_ind_factor	1 (×500 = 500*)	375 (×1=375*)
	r_fake_signal_high_th_ind_factor	1 (×1000 = 1000*)	750 (×1=750*)
GausHit Finder	gaushit ROI threshold: plane 0	2.9	1.5
	gaushit ROI threshold: plane 1	2.6	1.5
	gaushit ROI threshold: plane 2	3.5	1.0

*WireCell *collection* "threshold": `r_fake_signal_[low/high]_th`
WireCell *induction* "threshold": `r_fake_signal_[low/high]_th` × `r_fake_signal_[low/high]_th_ind_factor`

BlipReco toolkit in MicroBooNE



- Tools for MeV-scale reconstruction developed in MicroBooNE: ***BlipReco***
  - Functionality factorized into dedicated algorithm class in the *ubreco* MicroBooNE LArSoft repository, allowing for flexible integration into other reco/analysis modules
  - Integration into production-level software in progress in preparation for next reconstruction campaign
  - Goal: experiment-agnostic LArSoft tool and eventual data object
- This toolkit used in follow-up study to the radon mitigation paper
  - In internal review
  - Expected publication within ~1 month

BlipReco overview

1. **Isolated hits identification**

   Hits *within* tracks > configurable length are vetoed; optional 2D masking in regions surrounding long tracks

2. **Hit clustering per plane**

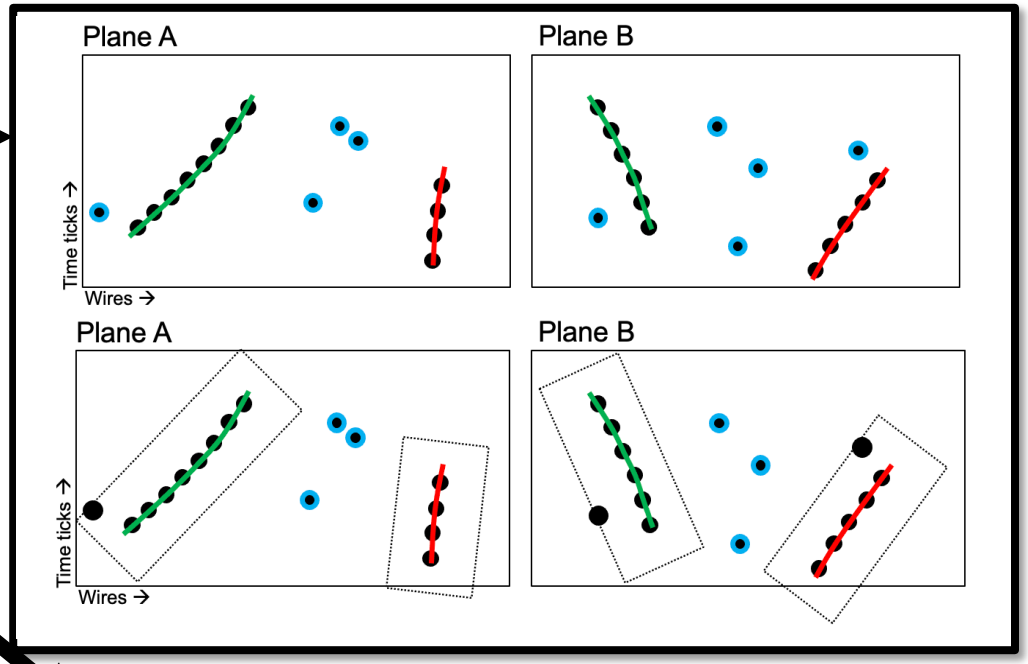
   Hit width ('RMS') defines proximity threshold for clustering in wire-time space

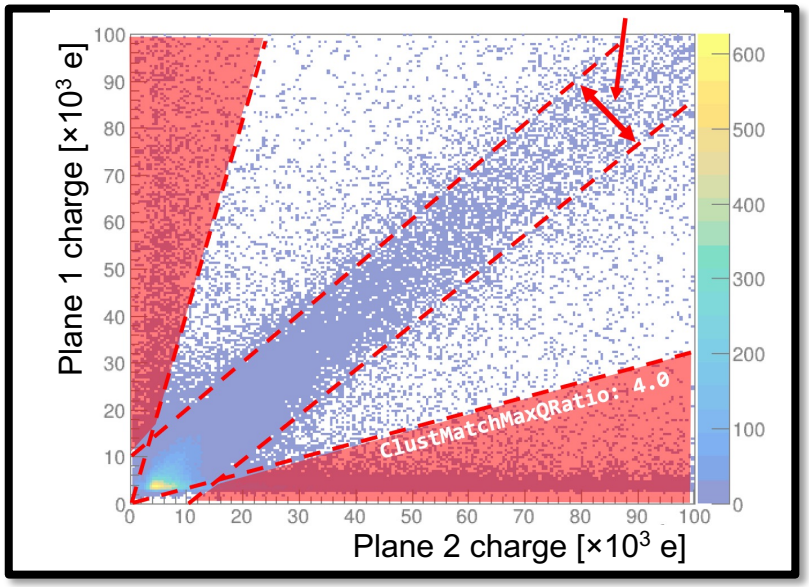
3. **Cluster time-matching**

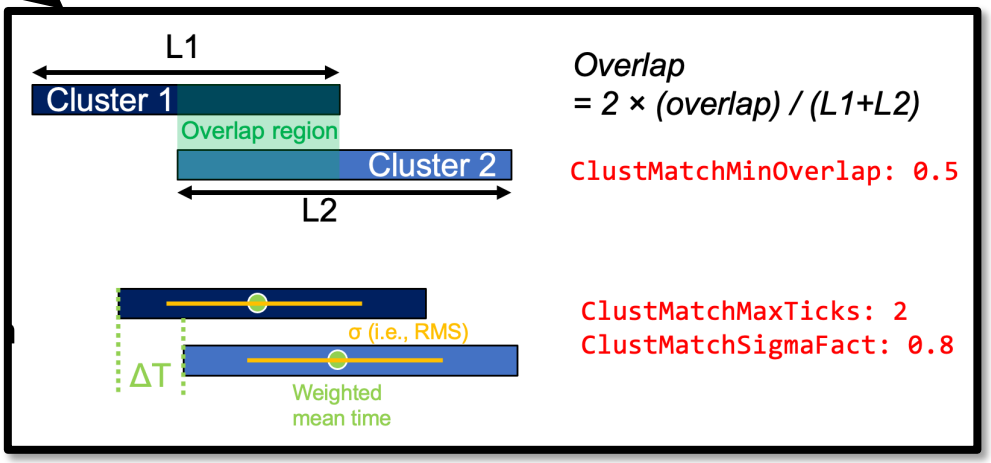
4. **Geometric requirement**

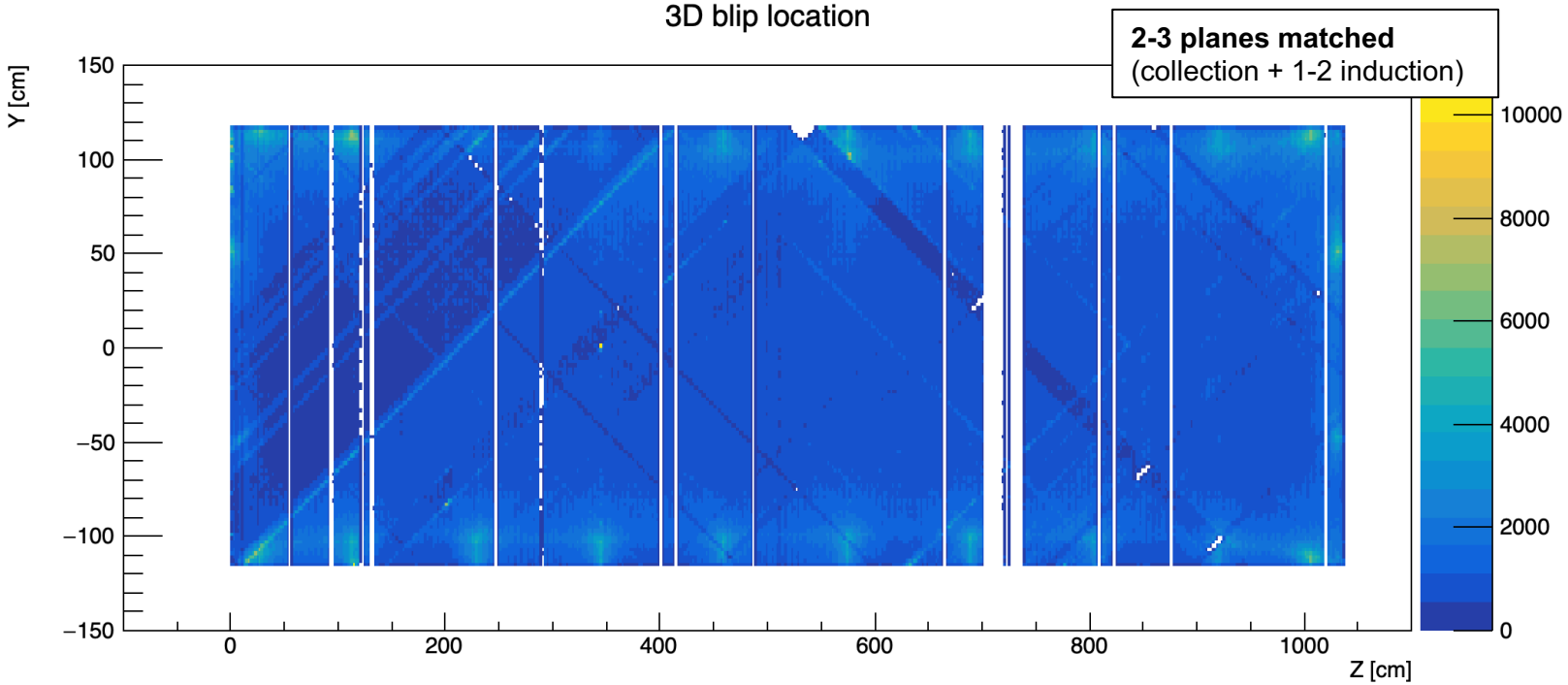
   Wires must cross!

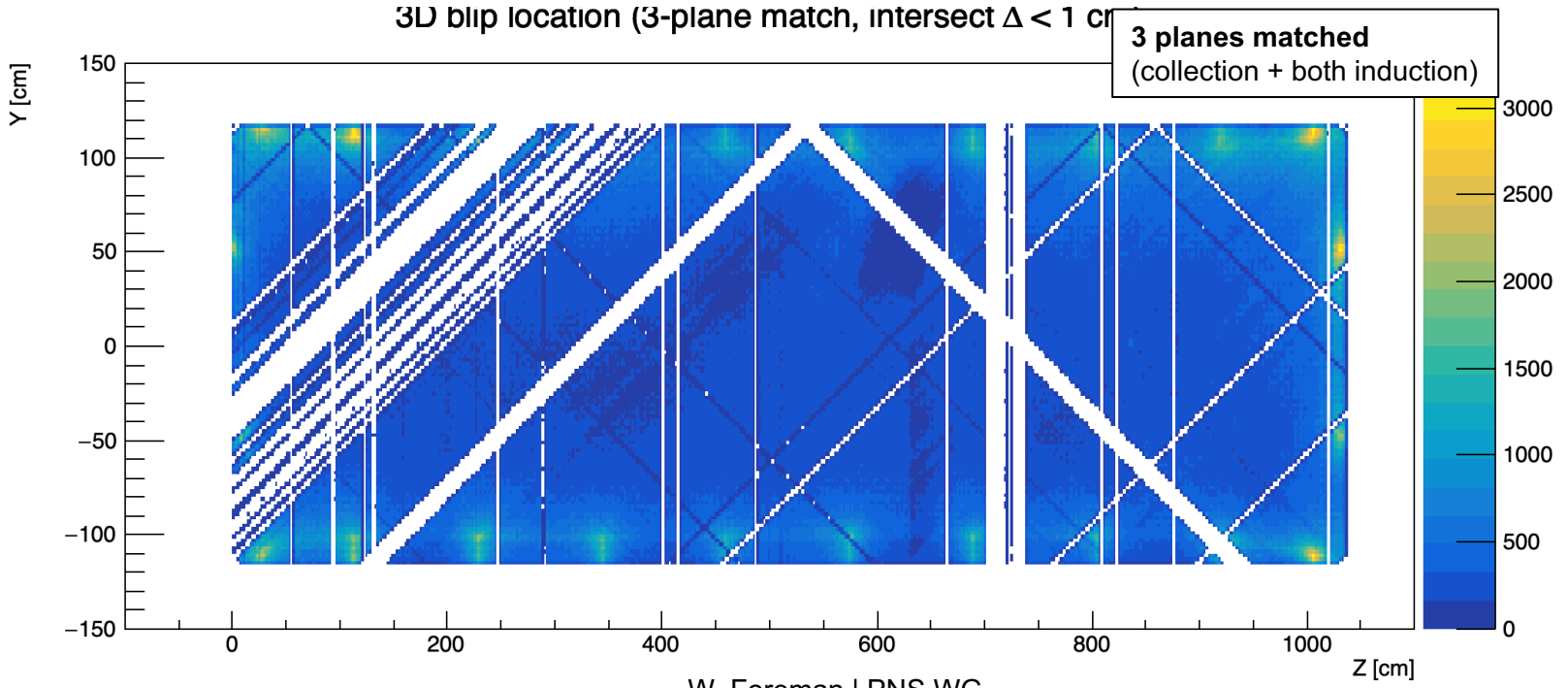
5. **Relative charge comparison**








3D blip location
2-3 planes matched
(collection + 1-2 induction)
Y [cm]
Z [cm]
150
100
50
0
−50
−100
−150
0
200
400
600
800
1000
10000
8000
6000
4000
2000
0


3D blip location (3-plane match, intersect Δ < 1 cm
3 planes matched
(collection + both induction)
Y [cm]
Z [cm]
150
100
50
0
−50
−100
−150
0
200
400
600
800
1000
3000
2500
2000
1500
1000
500
0

BlipReco overview

Some other features

- Electron lifetime corrections
- YZ plane non-uniformity corrections
- Space charge corrections: spatial offset + local E-field determination
- Charge-to-energy conversion using local E-field
  - Fcl-configurable dE/dx (default 2.8 MeV/cm, applicable for ~ 1 MeV electrons)
- "Dead channel" awareness: uses LArSoft's `'ChannelStatusService'` to locate non-functional wires and veto blips within proximity (fcl-configurable)
- "Bad channel" masking options
  - Exclude customized input list and/or noisy channels identified upstream by WireCell

Code structure

`ubreco/BlipReco` (3.3 MB total)

```
Alg
BlipAna_module.cc
blipreco_badchannels.txt
blipreco_configs.fcl
BlipRecoProducer_module.cc
CMakeLists.txt
job
ParticleDump_module.cc
TrackMasker_module.cc
Utils
```

Code structure

`ubreco/BlipReco` (3.3 MB total)

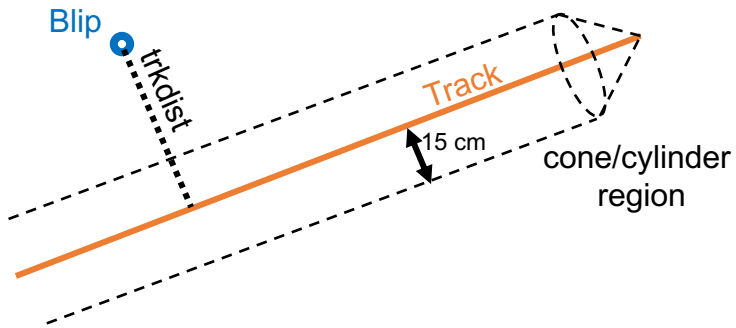
```
Alg
BlipAna_module.cc
blipreco_badchannels.txt
blipreco_configs.fcl
BlipRecoProducer_module.cc
CMakeLists.txt
job
ParticleDump_module.cc
TrackMasker_module.cc
Utils
```

Utils

```
BlipUtils.cc
BlipUtils.h
classes_def.xml
classes.h
CMakeLists.txt
DataTypes.h
```

DataTypes.h

```
struct Blip {

  int       ID              = -9;         // Blip ID / index
  bool      isValid         = false;      // Blip passes basic checks
  int       TPC             = -9;         // TPC
  int       NPlanes         = -9;         // Num. matched planes
  int       MaxWireSpan     = -9;         // Maximum span of wires on any plane cluster
  float     Charge          = -9;         // Charge on calorimetry plane
  float     Energy          = -999;       // Energy (const dE/dx, fcl-configurable)
  float     EnergyESTAR     = -999;       // Energy (ESTAR method from ArgoNeuT)
  float     Time            = -999;       // Drift time [ticks]
  float     ProxTrkDist     = -9;         // Distance to cloest track
  int       ProxTrkID       = -9;         // ID of closest track
  bool      inCylinder      = false;      // Is it in a cone/cylinder region?

  TVector3  Position;                     // 3D position TVector3
  float     SigmaYZ         = -9.;        // Uncertainty in YZ intersect [cm]
  float     dX              = -9;         // Equivalent length along drift direction [cm]
  float     dYZ             = -9;         // Approximate length scale in YZ space [cm]

  // Plane/cluster-specific information
  blip::HitClust clusters[kNplanes];

  // Truth-matched energy deposition
  blip::TrueBlip truth;

  // Prototype getter functions
  double X() { return Position.X(); }
  double Y() { return Position.Y(); }
  double Z() { return Position.Z(); }
```



"Blip" data object prototype (C++ struct)

- Encodes XYZ, charge, & energy of 3D blips
- Includes distance to nearest track & track cone-cylinder region flag

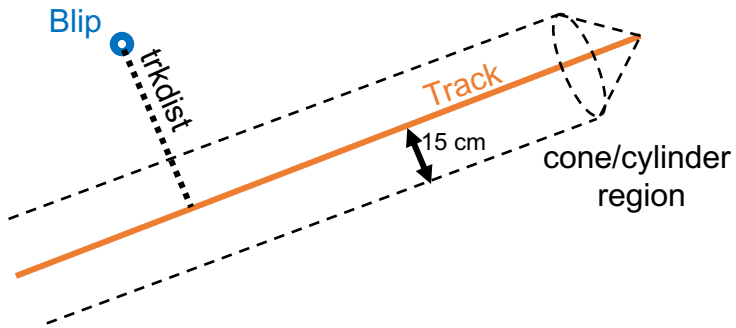
Code structure

`ubreco/BlipReco` (3.3 MB total)

```
Alg
BlipAna_module.cc
blipreco_badchannels.txt
blipreco_configs.fcl
BlipRecoProducer_module.cc
CMakeLists.txt
job
ParticleDump_module.cc
TrackMasker_module.cc
Utils
```

Utils

```
BlipUtils.cc
BlipUtils.h
classes_def.xml
classes.h
CMakeLists.txt
DataTypes.h
```



DataTypes.h

```
struct Blip {

  int       ID             = -9;        // Blip ID / index
  bool      isValid        = false;     // Blip passes basic checks
  int       TPC            = -9;        // TPC
  int       NPlanes        = -9;        // Num. matched planes
  int       MaxWireSpan    = -9;        // Maximum span of wires on any plane cluster
  float     Charge         = -9;        // Charge on calorimetry plane
  float     Energy         = -999;      // Energy (const dE/dx, fcl-configurable)
  float     EnergyESTAR    = -999;      // Energy (ESTAR method from ArgoNeuT)
  float     Time           = -999;      // Drift time [ticks]
  float     ProxTrkDist    = -9;        // Distance to cloest track
  int       ProxTrkID      = -9;        // ID of closest track
  bool      inCylinder     = false;     // Is it in a cone/cylinder region?

  TVector3  Position;                   // 3D position TVector3
  float     SigmaYZ        = -9.;       // Uncertainty in YZ intersect [cm]
  float     dX             = -9;        // Equivalent length along drift direction [cm]
  float     dYZ            = -9;        // Approximate length scale in YZ space [cm]

  // Plane/cluster-specific information
  blip::HitClust clusters[kNplanes];

  // Truth-matched energy deposition
  blip::TrueBlip truth;

  // Prototype getter functions
  double X() { return Position.X(); }
  double Y() { return Position.Y(); }
  double Z() { return Position.Z(); }
```

"Blip" data object prototype (C++ struct)

- Encodes XYZ, charge, & energy of 3D blips
- Includes distance to nearest track & track cone-cylinder region flag
- Truth-matching information also encoded

DataTypes.h

```
// True energy depositions
struct TrueBlip {
  int       ID             = -9;      // unique blip ID
  int       TPC            = -9;      // TPC ID
  float     Time           = -999e9;  // time [us]
  float     Energy         = 0;       // energy dep [MeV]
  int       DepElectrons   = 0;       // deposited electrons
  int       NumElectrons   = 0;       // electrons reaching wires
  float     DriftTime      = -9;      // drift time [us]
  int       LeadG4ID       = -9;      // lead G4 track ID
  int       LeadG4Index    = -9;      // lead G4 track index
  int       LeadG4PDG      = -9;      // lead G4 PDG
  float     LeadCharge     = -9;      // lead G4 charge dep
  TVector3  Position;                 // XYZ position
```

Using the tool

- Single call to algorithm is all that's required
  - Alg takes pointer to entire `art::Event` and does all the magic behind the scenes
  - Returns a vector of 'Blip' objects that the user is free to incorporate into their analysis or reconstruction as they see fit

```
//============================================
// Run blip reconstruction:
//============================================

fBlipAlg->RunBlipReco(evt);

//
//  In the above step, we pass the entire art::Event to the algorithm,
//  and it creates a single collection of blip 'objects', a special data
//  struct in the 'blip' namespace defined in BlipUtils.h.
//
//  We can then retrieve these blips and incorporate them into
//  our analysis however we like:
//
//    std::vector<blip::Blip> blipVec = fBlipAlg->blips;
//
//  The alg also creates collections of 'HitInfo' and 'HitClust'
//  structs used in the blip reconstruction process, which can be
//  accessed in the same way as blips.
//
//    * HitInfo simply saves some calculations for each hit that aren't
//      present in the native recob::Hit object, like drift time, associated
//      G4 particle IDs, etc.
//
//    * HitClust is just a cluster of hits on a specific plane; these are
//      used to create 3D blips by plane-matching.
//
```

Example of looping through blips and filling histograms of XYZ, energy, and true energy

```
for(auto& blip : blipVec ) {
  h_histogram_X     ->Fill( blip.x );
  h_histogram_Y     ->Fill( blip.y );
  h_histogram_Z     ->Fill( blip.z );
  h_histogram_E     ->Fill( blip.Energy );
  h_histogram_TrueE ->Fill( blip.truth.Energy );
}
```

Summary

- ***BlipReco* toolkit developed in MicroBooNE for standardized MeV-scale reconstruction in LArTPCs**
  - Ideas pioneered by ArgoNeuT and earlier MicroBooNE analyses
  - New results using these tools in μB to be published this summer and presented to this group
- **Advantages:**
  - Flexible, lightweight, fast, user-friendly
  - Requires only a collection of hits (recob::Hit) to work, with track collection (recob::Track) recommended but optional
  - Built-in masking surrounding long tracks
  - Options for filtering hits based on quality metrics: amplitude, RMS, GOF, ...
- **Disadvantages**
  - Performance and sensitivity thresholds limited by upstream hit-finding and wire processing algorithms