New ideas for reconstruction/tracking in LAr

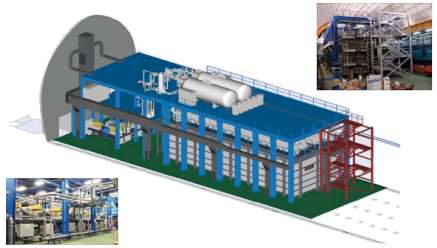
Gianluca Petrillo

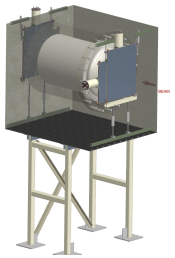
University of Rochester/Fermilab

November 4th , 2014,
International Workshop
on Next generation Nucleon Decay and Neutrino Detectors,
APC Laboratory, Paris, France

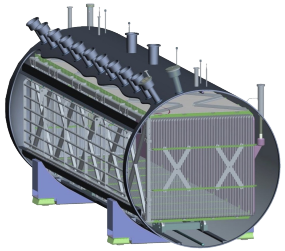
Reconstruction in liquid argon TPC

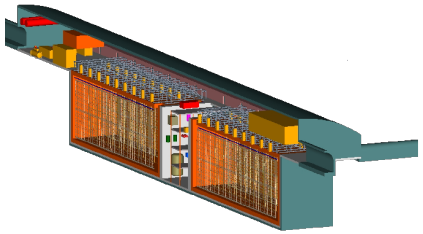
Liquid Argon TPCs can provide a high-definition 3D view of a physics event, together with particle energy estimation.
All LAr TPC detectors currently existing or planned share most design principles (box geometry, signal collection planes, uniform electric field, presence of optical detectors) and have to address the same fundamental problems.





Icarus T600, ArgoNeuT





MicroBooNE, LBNE 34kt

LArSoft

Neutrino experiments and prototypes based on LAr TPC have proliferated at Fermilab in the recent years:

- ArgoNeuT, Bo's, MicroBooNE, LArIAT, LBNE, LAr1ND...
- all share the same fundamental detector technology
- many of them are small collaborations...
- ... and many of the collaborators work on many of them

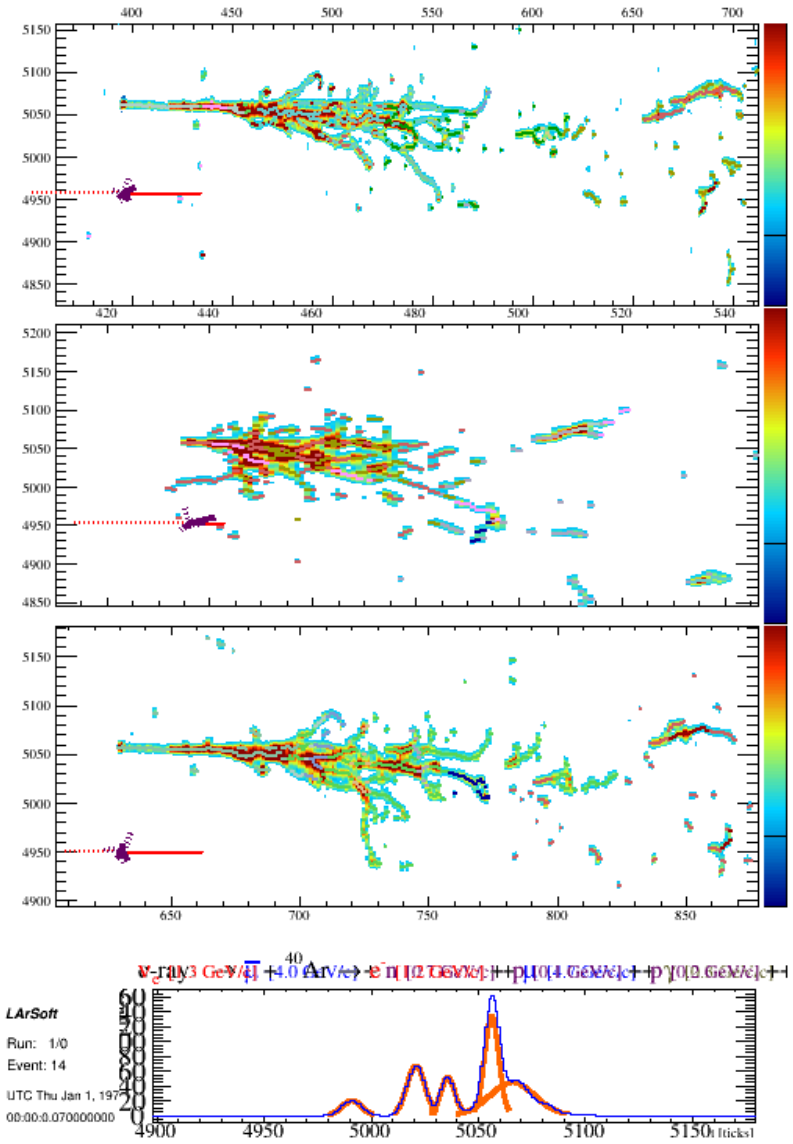
Fermilab has started the LArSoft project:

- a production-level-quality platform for simulation, reconstruction and analysis
- algorithms are general enough to be used by any of the experiments
- designed with enough flexibility to accommodate their specific needs
- partner experiments are at the foundation of the project:
  - contribute code and ideas
  - participate in the decisions

LArSoft content

LArSoft provides:

- **data:** signal processing, event reconstruction
- **simulation:** extensible interface with generators (e.g. Genie, CRY...), detector and readout simulation
- **analysis tools,** including event display
- **production:** serialization of data objects, interface with DB-based storage (e.g. SAMWeb)



Reconstruction of simulated $\nu_e(1.3\ \mathrm{GeV}/c) + {}^{40}\mathrm{Ar} \rightarrow e^- + 3p + 5n$ *in MicroBooNE* ↑

Why LArSoft?

The early, smaller experiments:

- got a high-quality underlying infrastructure for production
- shared the load of software development

Even the coming, larger experiments:

- get a complete infrastructure and a complete set of algorithms to simulate and analyze data immediately
- get a well tested and proofed software suite

LArSoft development

The project is in continuous development, and we strive to ahdere to good practises to have reusable and easy-to-develop software:

Facilitate early testing and validation of the new code

- provide an environment for test development
- provide automatic execution of test suites

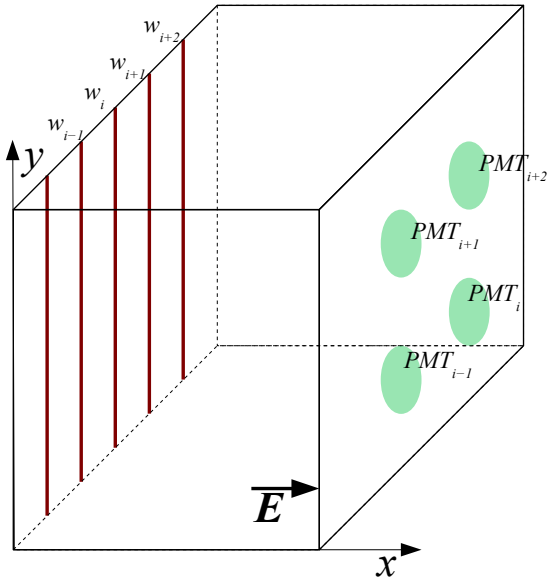
Keep "framework", "algorithms" and serialized data format independent

- allows LArSoft to be used in contexts where the `art` framework is not present: simple analysis code, a light-weight analysis framework, just a "ROOT macro", or any another production framework the experiments elect to use
- makes it easier to "import" existing code into LArSoft

Liquid Argon TPC in a nutshell

In a nutshell, a liquid argon TPC is made of:

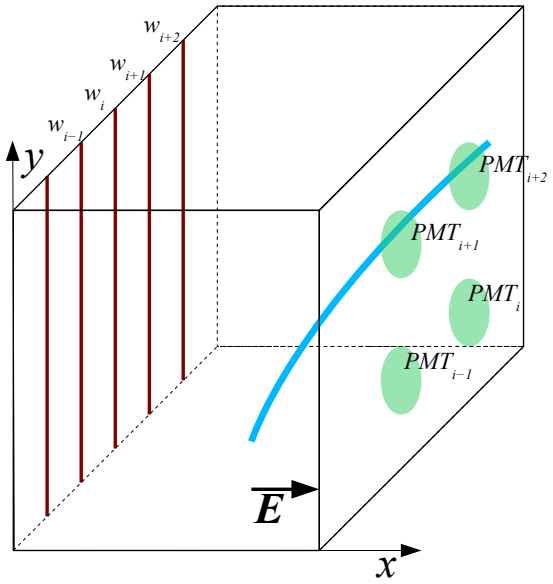
- target: the argon itself, mostly ^{39}Ar
- detection: argon is ionised, electrons drift at constant speed to the readout planes
- readout: currently planes of parallel wires, providing projection views for a stereoscopic image of the event
- event time: from ν beam source, and from photon detection



Liquid Argon TPC in a nutshell

In a nutshell, a liquid argon TPC is made of:

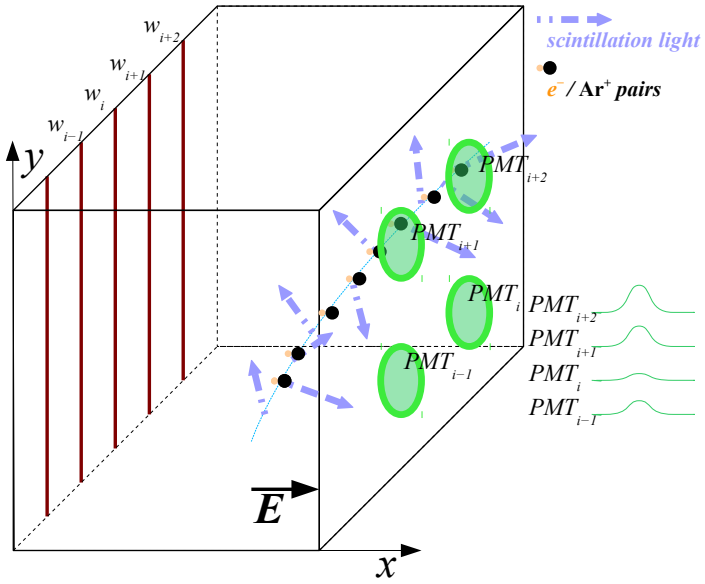
- target: the argon itself, mostly ^{39}Ar
- detection: argon is ionised, electrons drift at constant speed to the readout planes
- readout: currently planes of parallel wires, providing projection views for a stereoscopic image of the event
- event time: from ν beam source, and from photon detection



Liquid Argon TPC in a nutshell

In a nutshell, a liquid argon TPC is made of:

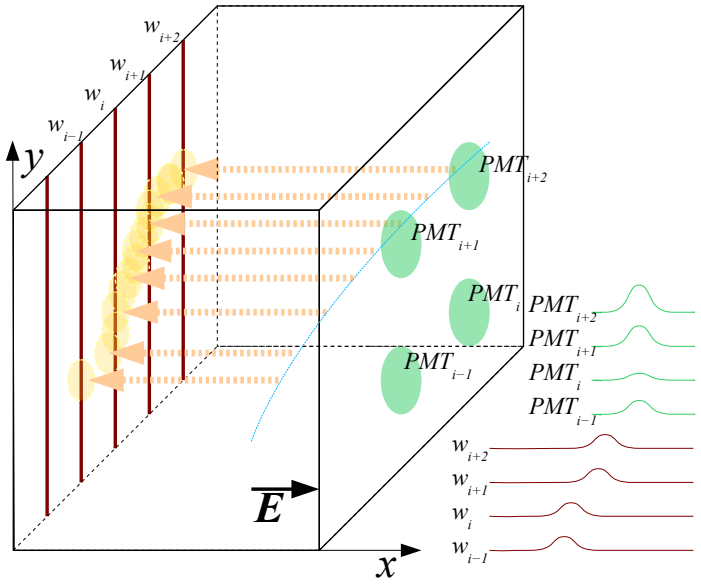
- target: the argon itself, mostly ^{39}Ar
- detection: argon is ionised, electrons drift at constant speed to the readout planes
- readout: currently planes of parallel wires, providing projection views for a stereoscopic image of the event
- event time: from ν beam source, and from photon detection



Liquid Argon TPC in a nutshell

In a nutshell, a liquid argon TPC is made of:

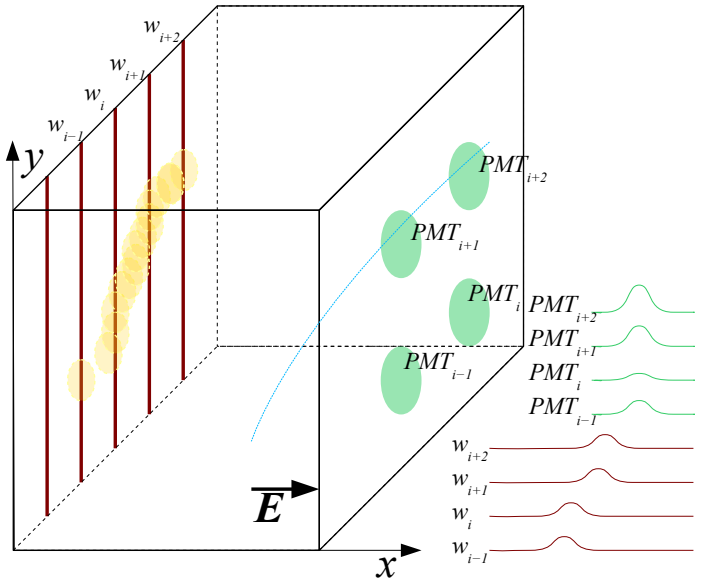
- target: the argon itself, mostly ^{39}Ar
- detection: argon is ionised, electrons drift at constant speed to the readout planes
- readout: currently planes of parallel wires, providing projection views for a stereoscopic image of the event
- event time: from ν beam source, and from photon detection



Liquid Argon TPC in a nutshell

In a nutshell, a liquid argon TPC is made of:

- target: the argon itself, mostly ^{39}Ar
- detection: argon is ionised, electrons drift at constant speed to the readout planes
- readout: currently planes of parallel wires, providing projection views for a stereoscopic image of the event
- event time: from ν beam source, and from photon detection



Typical reconstruction chain

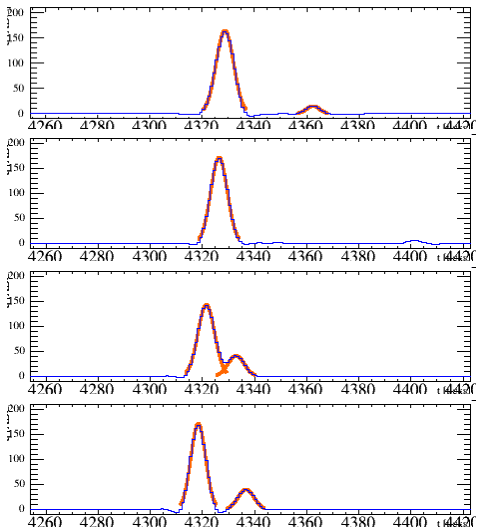
A typical reconstruction chain is made of multiple steps:

1. hit finding: signal induced on each wire from drifting ionization electrons
2. cluster finding: grouping hits lying on each wire plane
3. track finding: deducing a linear, 3D particle-like path
4. shower finding: outline a 3D electromagnetic shower

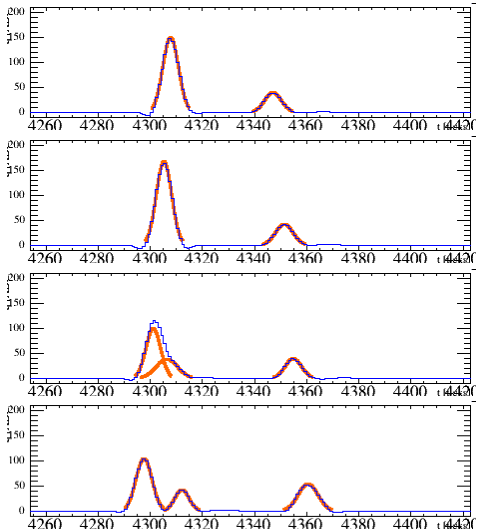
(x) vertex finding: deducing particle interaction points

Hit finding

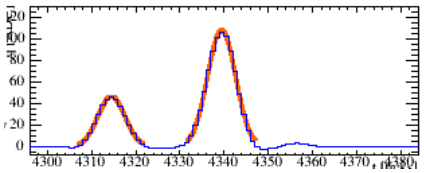
- from the signal on a wire (reshaped)
  in the plots: detected charge vs. time in ADC and TDC count units
- well-behaved hits have gaussian-shaped signal
- it is rare to see no signal where a hit is expected (not so in silicon trackers)
- overlapping may happen...
  - by chance (particles passing over the same wire at the same x)
  - by topology (close hits from interaction)



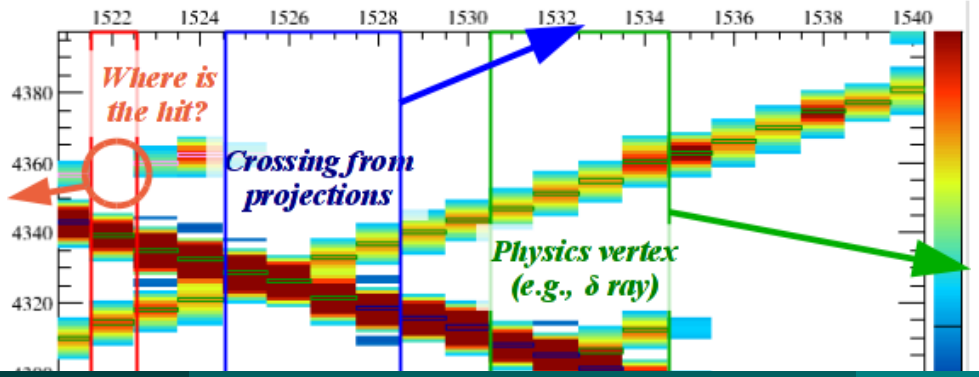
Crossing of projections



Physics vertex



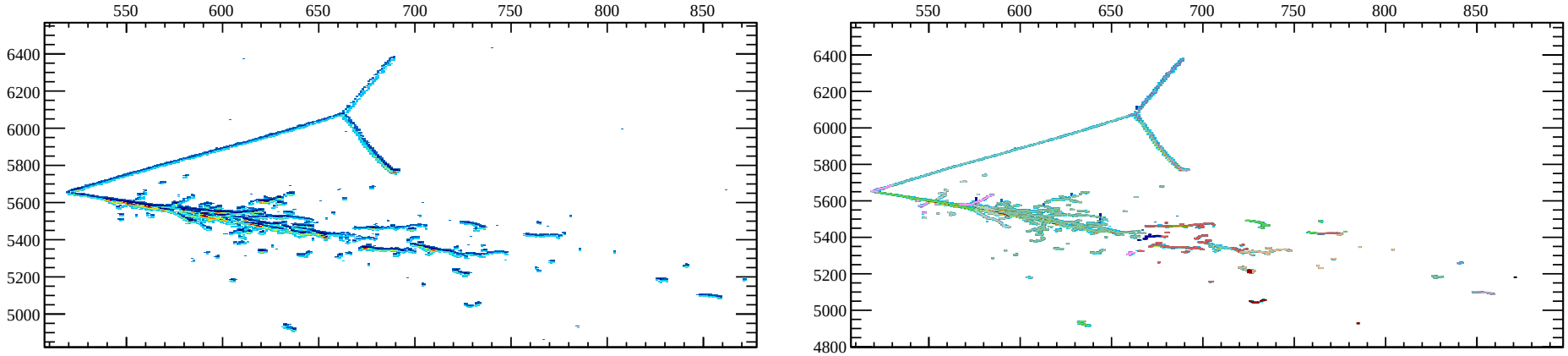
Hit not reconstructed (signal is there, but tiny)



Clustering

Clustering is the identification of hit structures on the 2D wire plane:

- ⇒ simplifies the input for the tracking algorithms: from $\mathcal{O}(1000)$ hits to $\mathcal{O}(10)$ clusters
- ⇒ helps correlating objects between planes



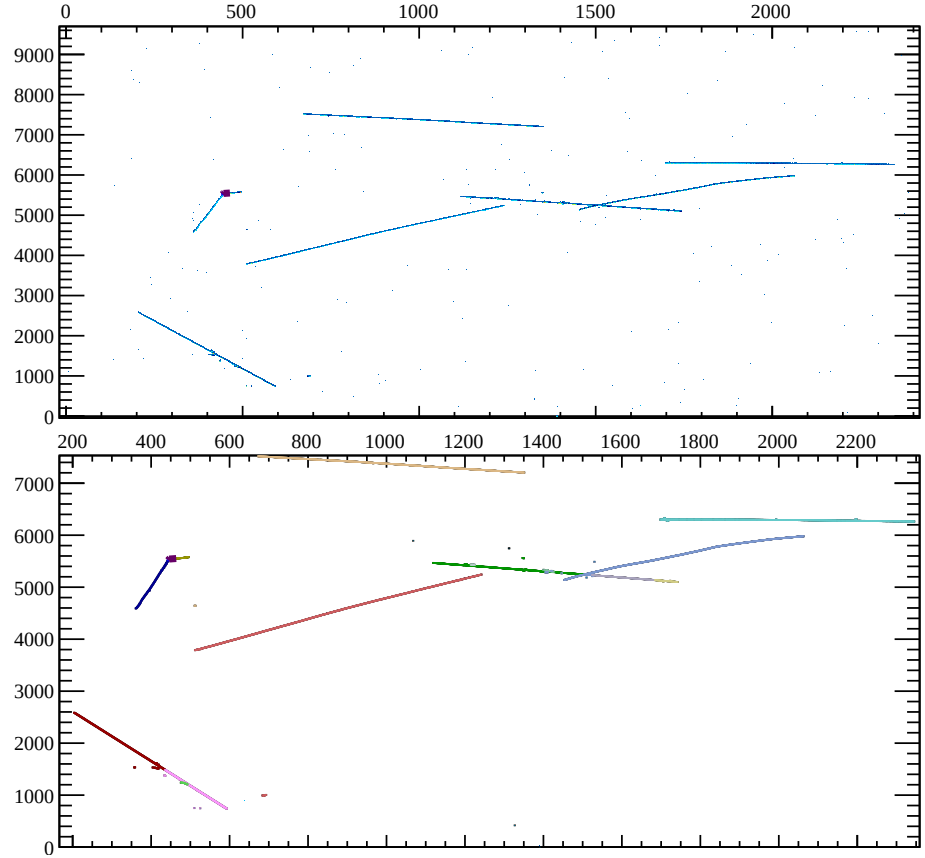
Detail of a $\nu_e + {}^{39}\mathrm{Ar}$ interaction in MicroBooNE simulation, from hits to clusters
The event shows both tracks of single ionizing particles and an electromagnetic shower.

Line finders

Hough transform is used in image processing to find straight lines:

- each hit "votes" a set of lines passing through it
- the lines with most votes become clusters
- a second pass is typically needed to stitch together almost-parallel segments
- purely geometric

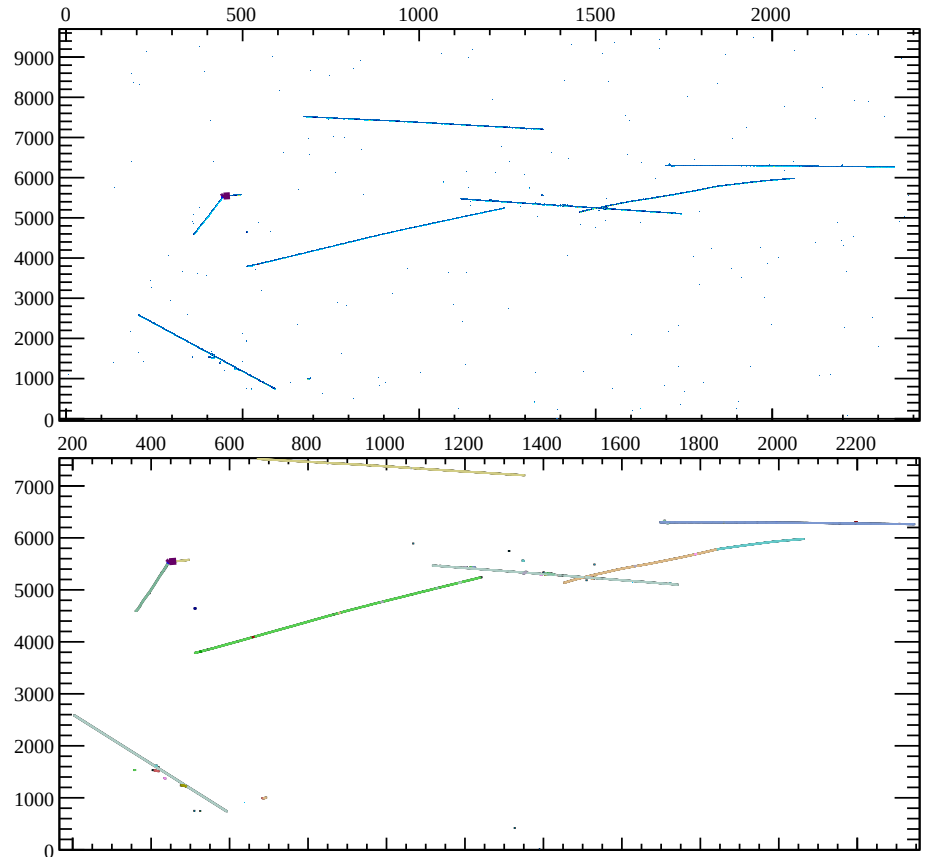
At the precision we need, it takes a lot of memory.



Simulation of a ν_μ +^{39}Ar event in MicroBooNE, with clustering by FLAME + Hough line finder + line stitcher

"Cluster crawler" *(B. Baller, MicroBooNE)*

- start from "downstream", where the event is cleaner
- three aligned hits constitute a seed cluster
- hits are added to the seed cluster if aligned
- the charge of new hits must be similar enough to the previous ones
- some allowance for short gaps and small curvature is included

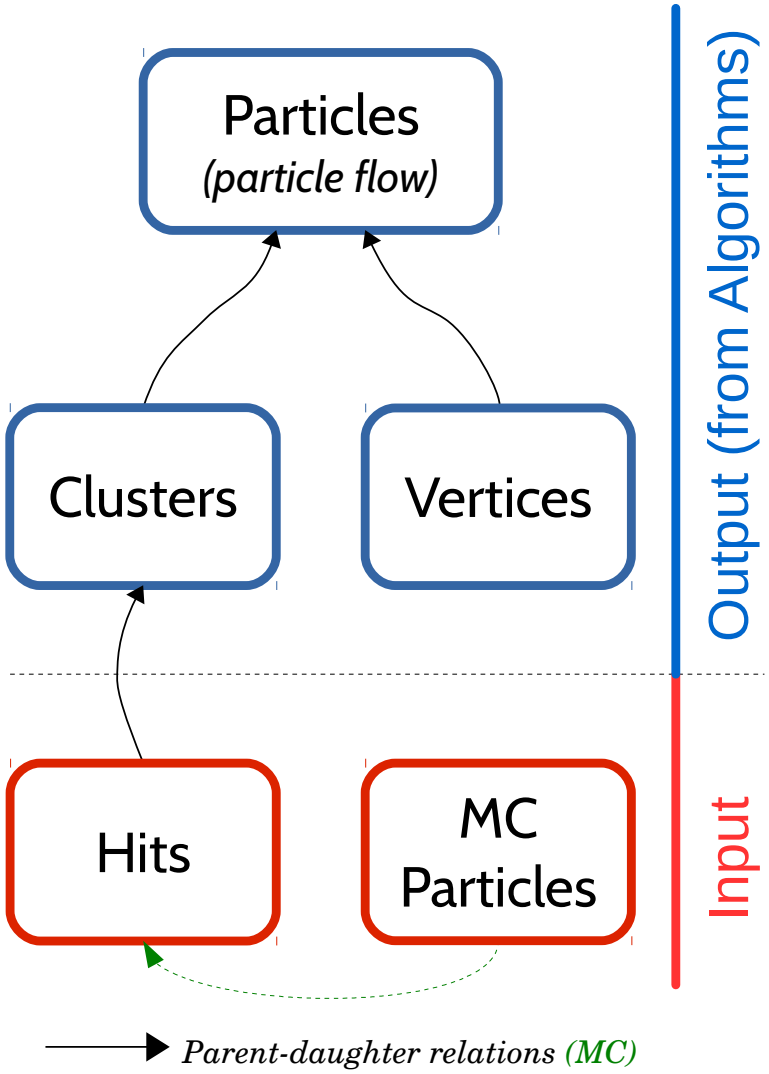


Simulation of a ν_μ + ^{39}Ar event in MicroBooNE, with clustering by cluster crawler

Pandora *(J. Marshall, M. Thomson, A. Blake)*

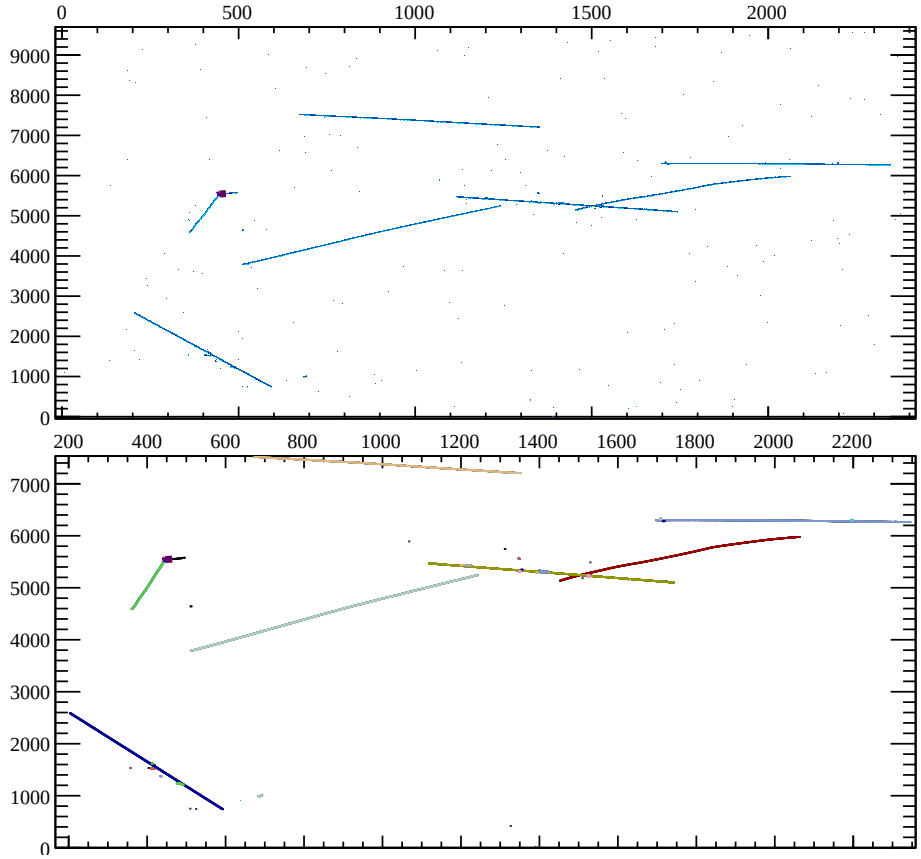
- a Software Developer Kit designed to simplify the creation of reusable pattern-recognition algorithms
- reconstruction is performed by (typically) large numbers of Pandora Algorithms
- each Algorithm tries to address a particular event topology

Pandora event model



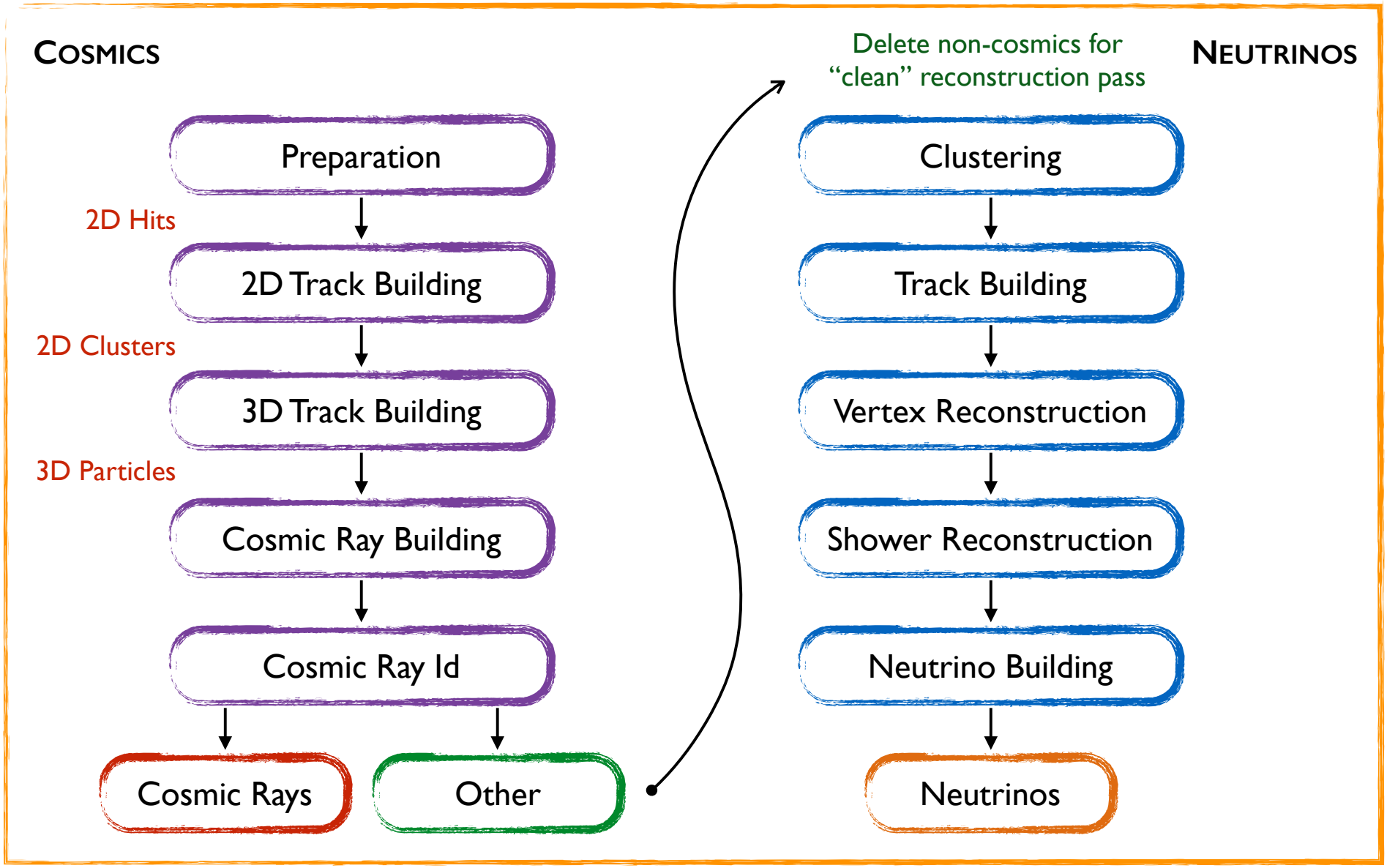
Pandora algorithms

- examples of algorithms:
  - find all straight segments
  - split segments with kinks
  - stitch aligned segments
  - fix splitting (δ ray-like) ...
- the chain to reconstruct cosmic rays includes about 70 calls to *small* algorithms
- algorithms try hard not to make mistakes, since it's very hard to recover
- in doubt, more alternatives can be carried on until the choice can be made



Simulation of a ν_μ + ^{39}Ar event in MicroBooNE, with clustering by Pandora

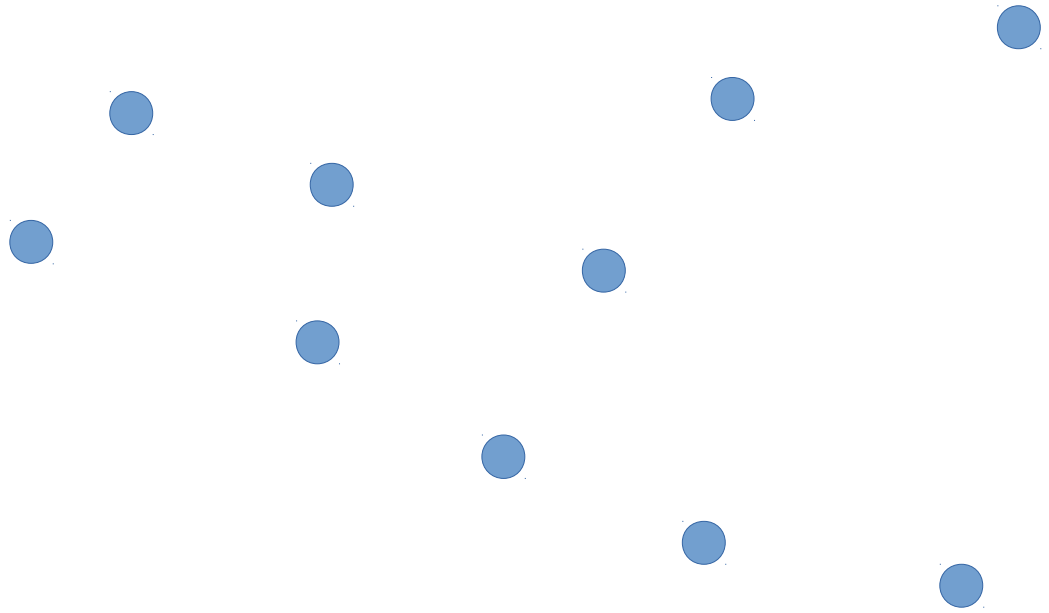
Pandora: reconstruction chain in LArSoft



Simple, fast cellular automata

Cellular automata models can be used to detect patterns:

- based on a graph of neighbouring hits and on an "evolution rule" refining the state of its elements until stability is reached
- evolution is simple and fast
- example from *Eur. Phys. J. C* (2013) 73:2591:

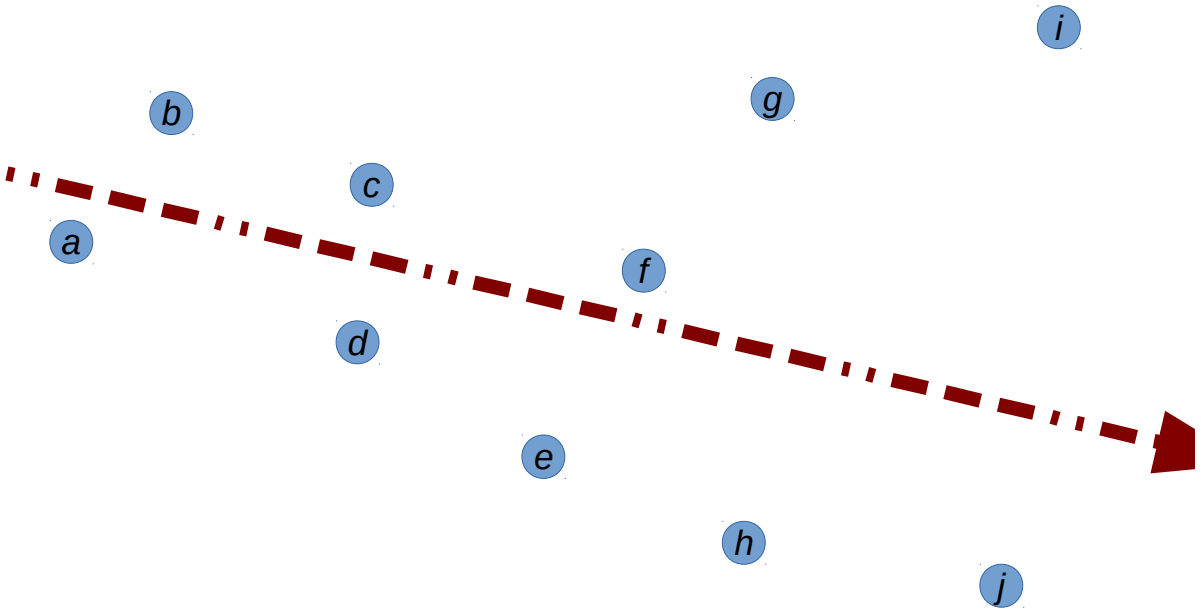


Start from the hits

Simple, fast cellular automata

Cellular automata models can be used to detect patterns:

- based on a graph of neighbouring hits and on an "evolution rule" refining the state of its elements until stability is reached
- evolution is simple and fast
- example from *Eur. Phys. J. C* (2013) 73:2591:

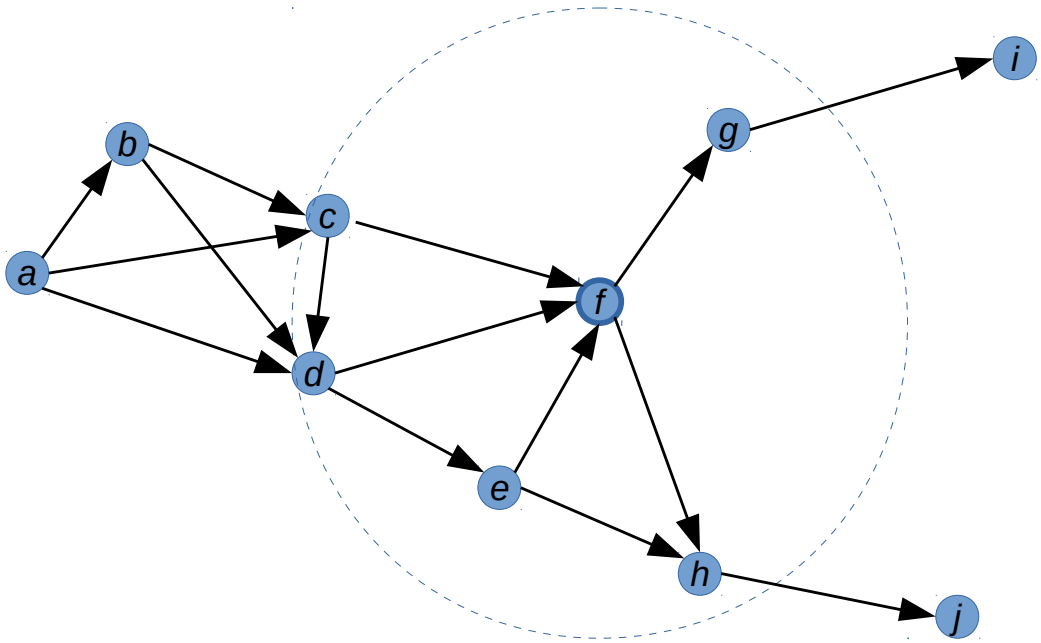


Sort them according to a direction

Simple, fast cellular automata

Cellular automata models can be used to detect patterns:

- based on a graph of neighbouring hits and on an "evolution rule" refining the state of its elements until stability is reached
- evolution is simple and fast
- example from *Eur. Phys. J. C* (2013) 73:2591:

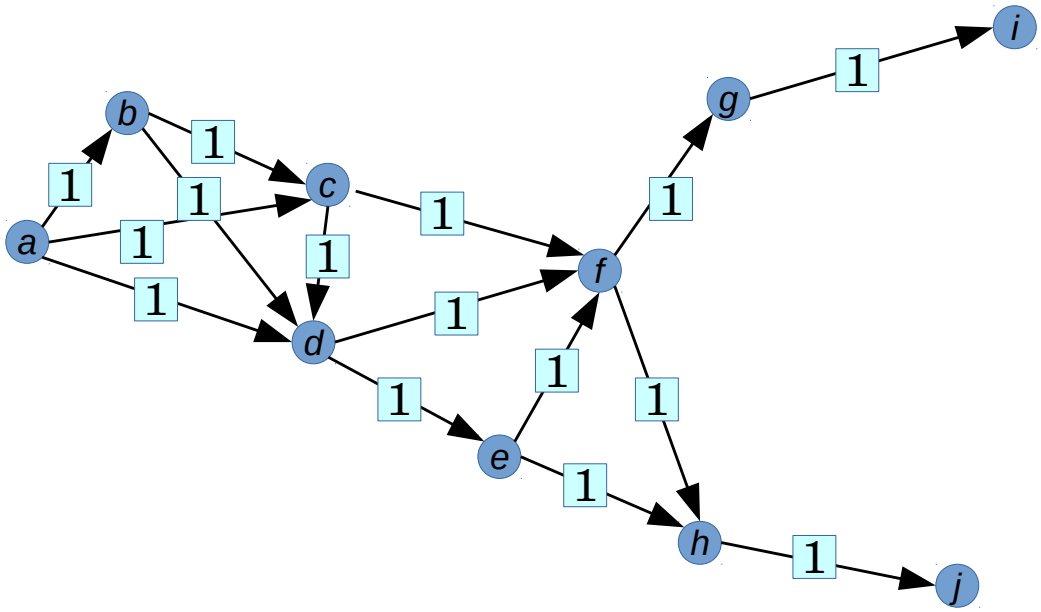


Establish neighbour relations (forward) and the "cells"

Simple, fast cellular automata

Cellular automata models can be used to detect patterns:

- based on a graph of neighbouring hits and on an "evolution rule" refining the state of its elements until stability is reached
- evolution is simple and fast
- example from *Eur. Phys. J. C* (2013) 73:2591:

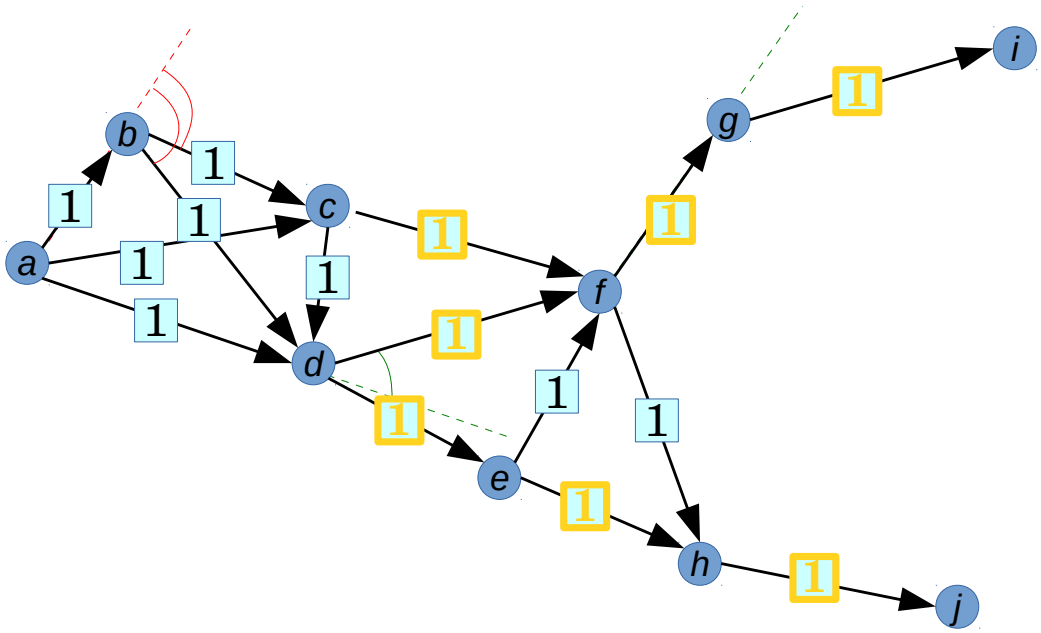


Assign the cells an initial status

Simple, fast cellular automata

Cellular automata models can be used to detect patterns:

- based on a graph of neighbouring hits and on an "evolution rule" refining the state of its elements until stability is reached
- evolution is simple and fast
- example from *Eur. Phys. J. C* (2013) 73:2591:

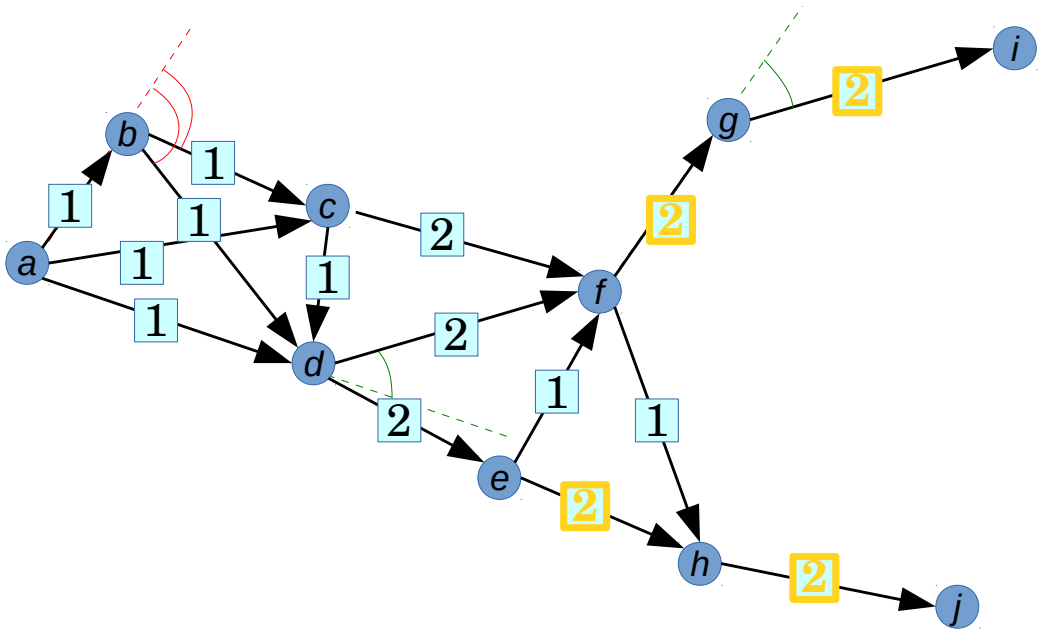


Increase the status of cells following others with same status

Simple, fast cellular automata

Cellular automata models can be used to detect patterns:

- based on a graph of neighbouring hits and on an "evolution rule" refining the state of its elements until stability is reached
- evolution is simple and fast
- example from *Eur. Phys. J. C* (2013) 73:2591:

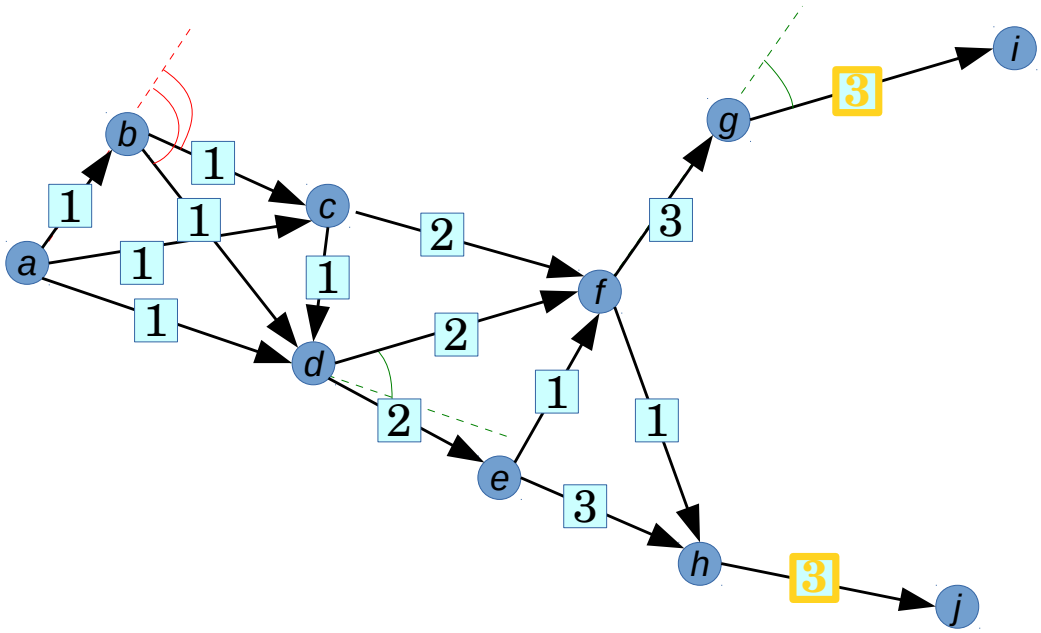


Repeat...

Simple, fast cellular automata

Cellular automata models can be used to detect patterns:

- based on a graph of neighbouring hits and on an "evolution rule" refining the state of its elements until stability is reached
- evolution is simple and fast
- example from *Eur. Phys. J. C* (2013) 73:2591:

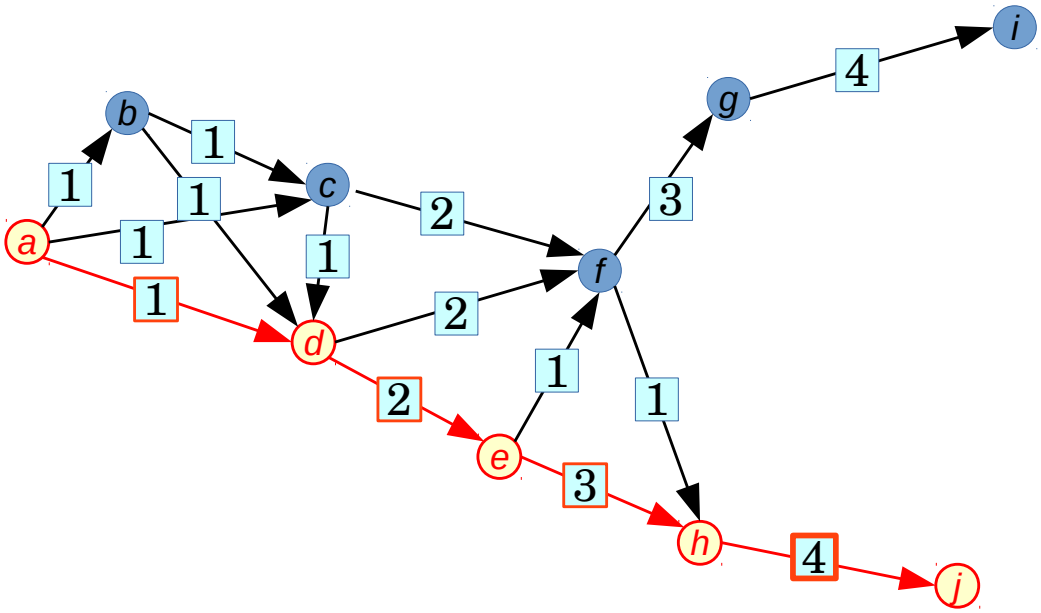


... until stable

Simple, fast cellular automata

Cellular automata models can be used to detect patterns:

- based on a graph of neighbouring hits and on an "evolution rule" refining the state of its elements until stability is reached
- evolution is simple and fast
- example from *Eur. Phys. J. C* (2013) 73:2591:

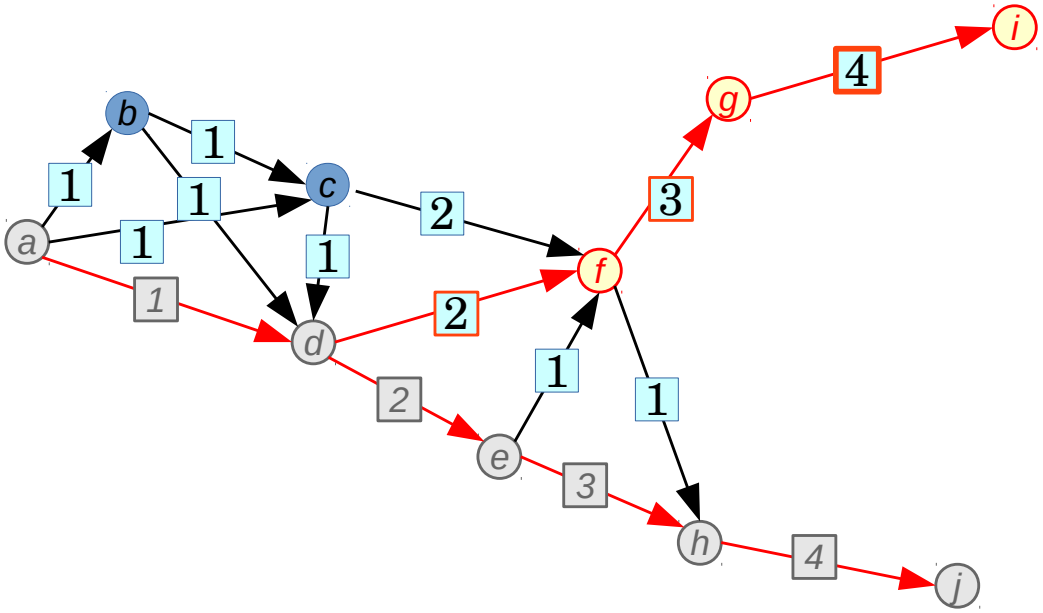


Find the longest path backward

Simple, fast cellular automata

Cellular automata models can be used to detect patterns:

- based on a graph of neighbouring hits and on an "evolution rule" refining the state of its elements until stability is reached
- evolution is simple and fast
- example from *Eur. Phys. J. C* (2013) 73:2591:



Find all the other paths

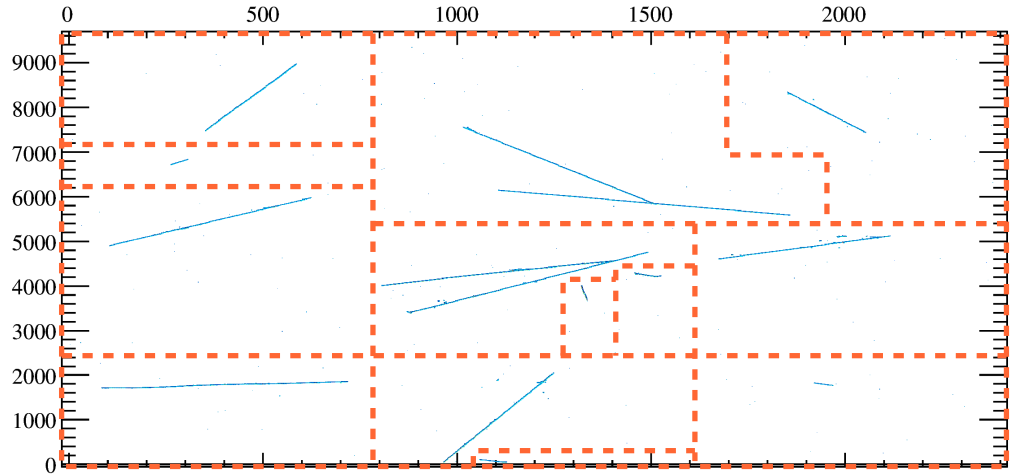
Simplifying clustering input: "preclustering"

As the input becomes larger:

- added combinatorics and noise deteriorate clustering performance
- required resources (time, memory) increase

⇒ it is convenient to partition the input space into independent regions

Partitioning can be effective in large detectors, where cosmic ray activity is sparse.



Example partition of hit activity in a wire plane (this was created by hand for illustration only)

Density-based spacial clustering

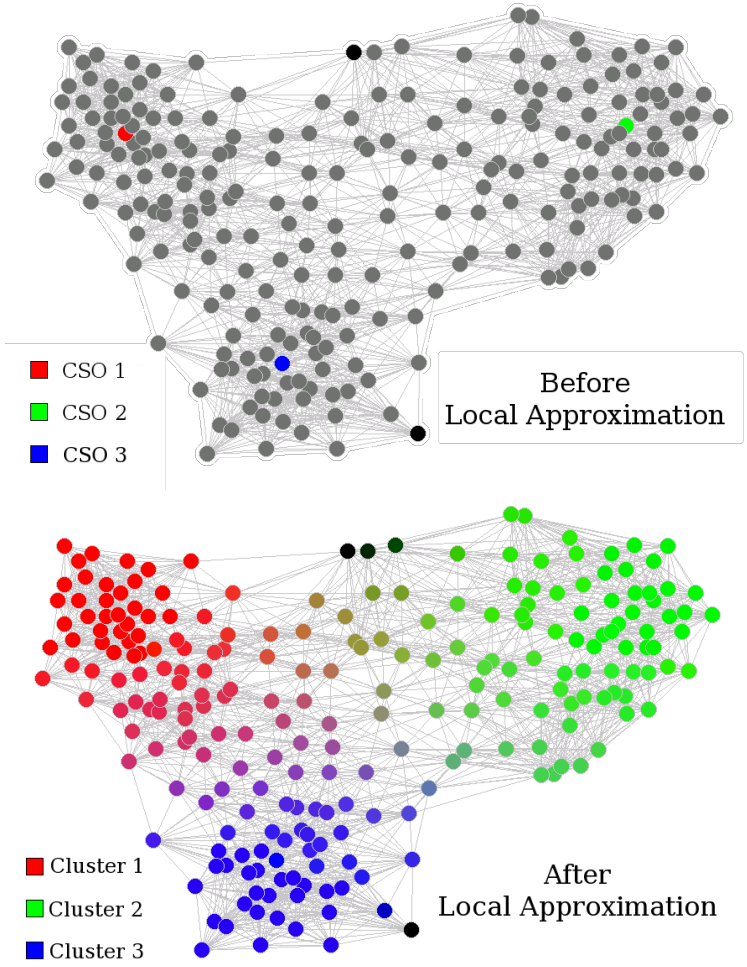
DBSCAN[3] is a well established algorithm, used for example by ArgoNeuT:

- parameters: distance ρ and minimum number of hits, n_{min}
- "dense areas" have at least n_{min} hits within distance ρ
- points in dense areas are clustered
- points not in dense areas are considered noise and ignored
- can be used in any dimensionality (wire $\times$ time or full 3D)
- does not assume shapes (e.g., a linear cluster)

Distance-based spacial clustering

FLAME[4] is a newer algorithm, falling in the fuzzy clustering category:

- fuzzy clustering distributes "shares" of a hit to each cluster, as opposed to assigning it all to a single cluster; but in the end we assign each hit to the cluster owning the largest share of it
- each hit that has (local) maximum density seeds one cluster
- FLAME minimizes the sum of distances from the seed of each hit in the cluster
- it is stable and generates a well-defined number of clusters



Result of FLAME clustering (from Wikipedia)

New tool for clustering: GIMP

We can borrow some techniques from the popular image processing world:

- Hough transform is one we have borrowed already
- smoothing may be used to reduce random noise hits
- hit detection might even become unnecessary
- corner detection can be used to guess track ends

As in the case of line finding by Hough algorithm, these techniques alone will probably fall short on the side of precision, yet they can be valuable as bootstrap for precision reconstruction.

Tracking options

We now want to reconstruct the trajectory of a physical particle. Many of the clustering algorithms are already designed to detect track-like structures:

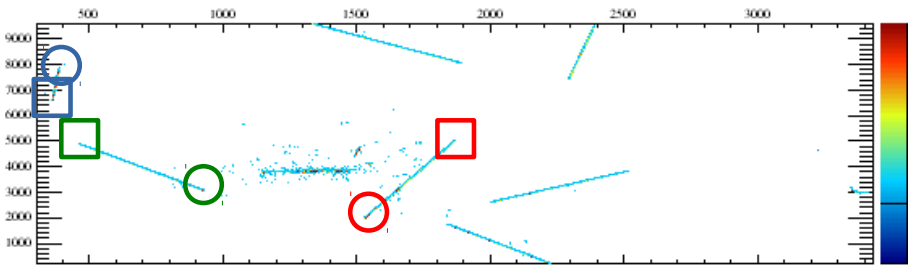
- clusters are often 2D "proto-tracks"
  $\rightarrow$ just match them between planes!
- or we can ignore clusters and go from scratch
  $\rightarrow$ need a starting point!
- clusters also represent regions of activity (especially from `DBSCAN` and fuzzy clustering)
  $\rightarrow$ roughly match them, vaporize them into hits and use the hits to build tracks

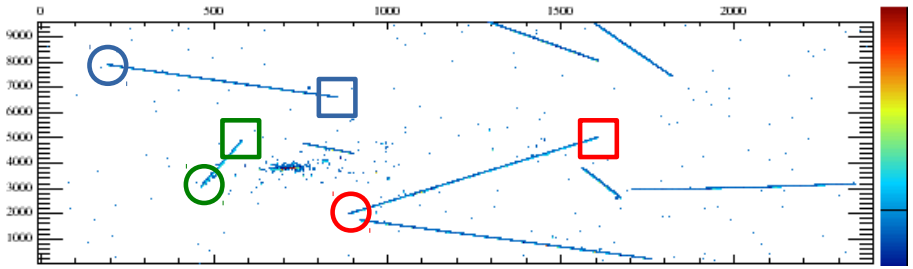
Tracks from spacial matching

Ideally, 2D clusters in different views from the same particle have the same distance from the wire plane, for both their ends:

- try all the combinations of two clusters from two different view
- try the longest first
- if they start and end time match, pair them
- if there are other views, match the clusters there too

This is simple, and greatly suffers if clusters are broken.



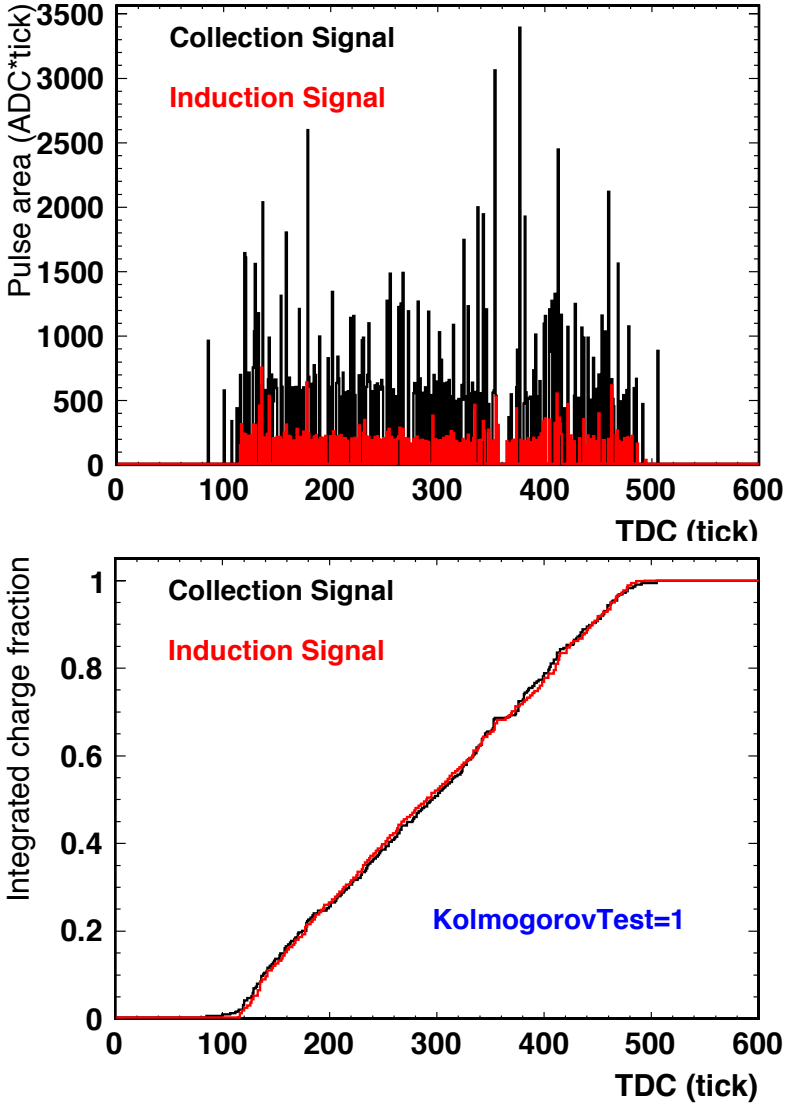


Charge distribution matching *(T. Yang, ArgoNeuT)*

A variation of this idea, already used by ArgoNeuT, relies on the fact that the different views see the same ionised charge:

- if the wires are well intercalibrated, they measure the same charge
- the distribution of total charge as function of time also matches
- a Kolmogorov test can decide on the goodness of the match

This allows matching of clusters with start and end time not perfectly matching.



Distribution and integral in time of measured charge from two matching clusters. Courtesy of ArgoNeuT.

Seeds

Many tracking algorithms need a hint of where to start: a seed!

$\Rightarrow$ a short group of aligned "space points"
(a space point is a 3D point from matching hits in different views)

This process can be extremely efficient: $> 99\%$ of tracks (but less for the very short ones) get at least one seed

A simple way of creating tracks is by interpolating seeds:

1. start from overlapping clusters
2. use their hits to find seeds
3. interpolate the seed segments

An implementation based on interpolation by Bezier curves is present in LArSoft *(B. Jones, MicroBooNE)*

Kalman filters

Kalman filters are a Tracking Tradition:

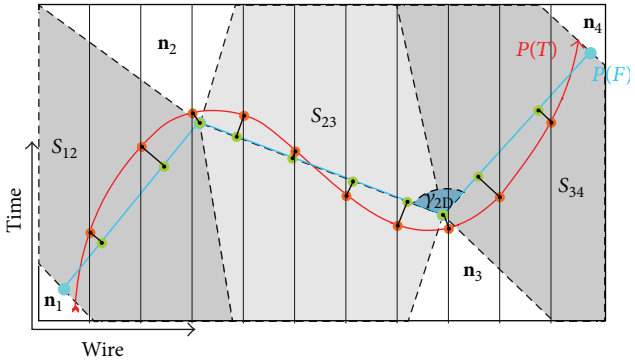
- the "filter" carries along track parameters and their covariance C
  - where the track is (u, v)
  - where it goes $(\partial u/\partial w, \partial v/\partial w)$
  - its momentum $(1/p)$, related to Coulomb scattering
- initial state from a seed segment
- merges the information of measurements one by one: either hits (2D) or "space points" (3D)
- learns to make better predictions as it is fed measurements

Polygonal Line Approximation

Icarus, [1]

Icarus has proposed an algorithm based on the simultaneous optimization of the 2D projections of the track to the cluster hits.

- a track is represented by a polygonal line
- start from a segment (two "nodes"), iteratively add nodes
- on each iteration, the position of the nodes is reoptimized:
  - minimize distance of the hits from the projections of the track
  - minimize distance from 3D vertices, if available
  - constrain angle between segments of the track



Projection of track fit (red) and observed hits (green).

Outlook: reconstruction in multiple passes

- the signal in a wire is correlated to the neighbouring ones
- charge and width of a hit from a particle almost parallel to a wire are larger
- close to a vertex, hits get mixed up and superimpose

Event topology provides additional, exploitable information: two- or multiple reconstruction iterations can improve reconstruction

- ongoing work on "Cluster Crawler" algorithm to include a second-pass *hit refining*
- the reconstruction chain of pandora for MicroBooNE tackles the eventual neutrino interaction starting over after having removed the hits from cosmic rays

This approach may be extended to improve all steps from wire signal processing to the final tracking and particle identification.

Outlook: hardware acceleration

Resource utilization efficiency is going to be a concern for experiments with large number of channels (e.g. LBNE, 240 TPC and >300k channels).

- zero suppression and fast detection of regions of activity help reducing the complexity
- parallelization can make the difference:
  - \+ promising studies in ArgoNeuT for Hough transform acceleration through GPUs
  - \+ GPU are designed for image processing, that we might exploit
  - – how many execution nodes have a GPU? and which GPU?
  - – how better N-thread jobs perform than N single-thread jobs?
  - – available memory per thread is going to drop!

As usual, well-designed algorithms are the best approach.

Summary

- liquid argon TPCs provide fine-grain information of events
- reconstruction needs to mix geometrical and physical information
- most of the described algorithms have been integrated by the authors into a single software suite, LArSoft, available to any experiment
- fully automated reconstruction software will be a necessity, and it is in our reach

References and supplemental material

References

[1] M. Antonello *et al.*,
"Precise 3D Track Reconstruction Algorithm for the ICARUS T600 Liquid Argon Time Projection Chamber Detector",
Adv. in High En. Phys. **2013**, 260820, 01/2013

[2] J. J. Back, G. J. Barker, A. J. Bennieston, S. B. Boyd, B. Morgan and Y. A. Ramachers,
"Implementation of a cellular automaton algorithm for neutrino interaction reconstruction in a liquid argon volume",
Eur. Phys. J. C (2013) 73:2591

[3] M. Ester, H-P. Kriegel, S. Jörg, X. Xu,
"A density-based algorithm for discovering clusters in large spatial databases with noise",
Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96) (1996) 226

[4] L. Fu, E. Medico,
"FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data",
BMC Bioinformatics 2007, **8**:3

Liquid Argon TPC from reconstruction perspective

muons from cosmic rays, or from ν_μ interaction: long track, minimum ionization, can kick "δ rays", can decay in an electron

electrons from ν_e interactions, "δ rays", muon decay: may develop in a electromagnetic shower

neutrons only visible if they kick around ^{39}Ar nuclei

protons, $\pi^\pm$ longish tracks, higher ionization than muons; pions may decay, protons... we'll see

photons, π^0 likely to generate one or two electromagnetic showers after a short space from interaction