

# Cluster3D Reconstruction And a Brief Overview of SpacePointSolver

Tracy Usher

Workshop on Calibration and Reconstruction  
for LArTPC Detectors

December 11, 2018

# Overview

- Motivation
- 3D Space Point Building
  - "Simple" Cluster3D approach
  - SpacePointSolver
- Event Slicing
- Towards Pure 3D Pattern Recognition

# Motivation

- Wire readout LArTPCs provide exquisite high resolution 2D images of events
- Stereo images allow for 3D reconstruction
  - Feature finding in 2D which are then matched to get 3D tracks, showers, etc.
  - Use 2D "hit" information to make 3D "space points" and then perform 3D pattern recognition
- Pros and Cons to both approaches
  - 2D feature finding techniques highly advanced but must deal with many special cases (especially for surface TPCs)
  - In 3D the bulk of the special cases disappear and track/shower reconstruction should be "straightforward"... but of course first you need a way to unambiguously build 3D space points
- Believe 3D approaches ultimately superior and we should work on developing tools to enable them

Build Space Points:  
Simple Approach

# Simple Space Points

- Driving philosophy is that one builds and keeps space points from all "allowed" combinations of individual 2D hits
  - Want high efficiency for "true" space points, willing to accept to accept some level of "fake" space points to achieve goal
    - In fact, one has to accept that there are always ambiguous combinations
  - Assume 3D level algorithms will resolve allowed ambiguities
- What are "allowed" combinations
  - Hits on different planes, wires must intersect (cross) to make combination
    - 3 2D hits crossing must form the minimum size triangle
  - Hits must agree in time
    - Difference in peak times of 2D hits within range defined by widths
- The approach is simple, the work is in handling the combinatorics

# The Obvious Problems

- Creating space points depends critically on the 2D hit finding efficiency and quality
  - Missing 2D hits will result in either missing space points -or- (worse) the wrong space points
  - Obviously, the 2D hit finding depends critically on the signal processing
- Building the "correct" space points from 2D hits also requires understanding inter-plane timing offsets

# The Obvious Pathologies

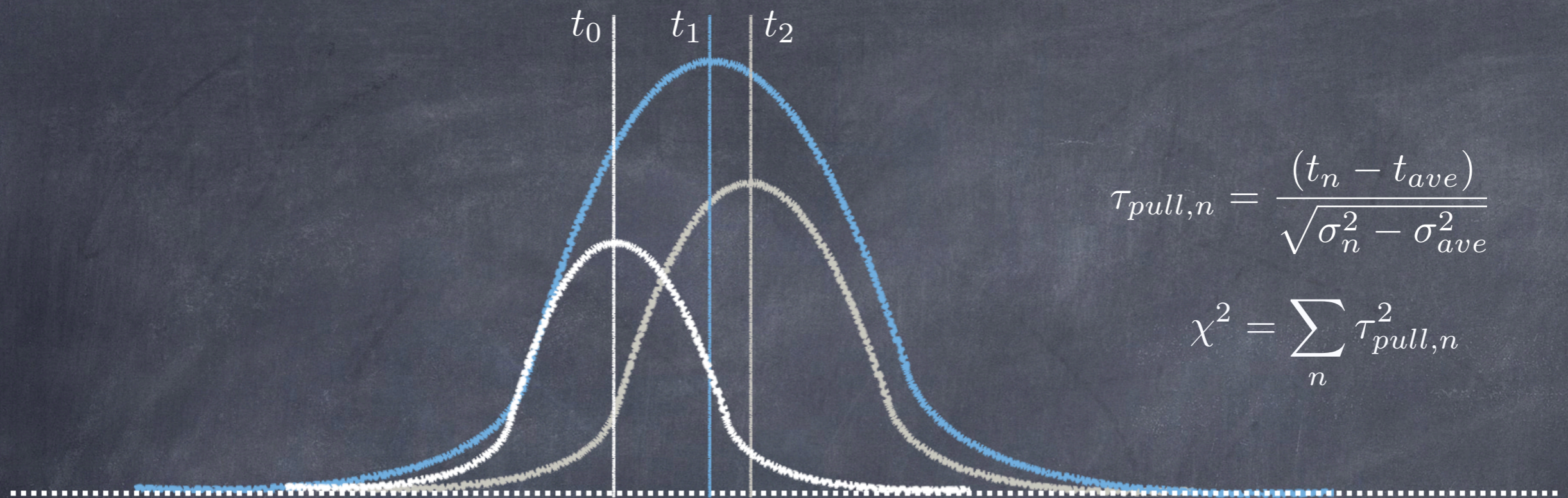
- Isochronous tracks

- Large numbers of 2D hits will agree in time and have good values for the metric above.
- Generally, once the hits start to have a separation on the order of the average width then the quality metric starts to have value in sorting these out

- Distorted waveforms

- Primarily an issue for tracks running along the x axis which create long pulse trains on small number of wires
- How best to handle these for the purposes of making 3D space points is still an open question

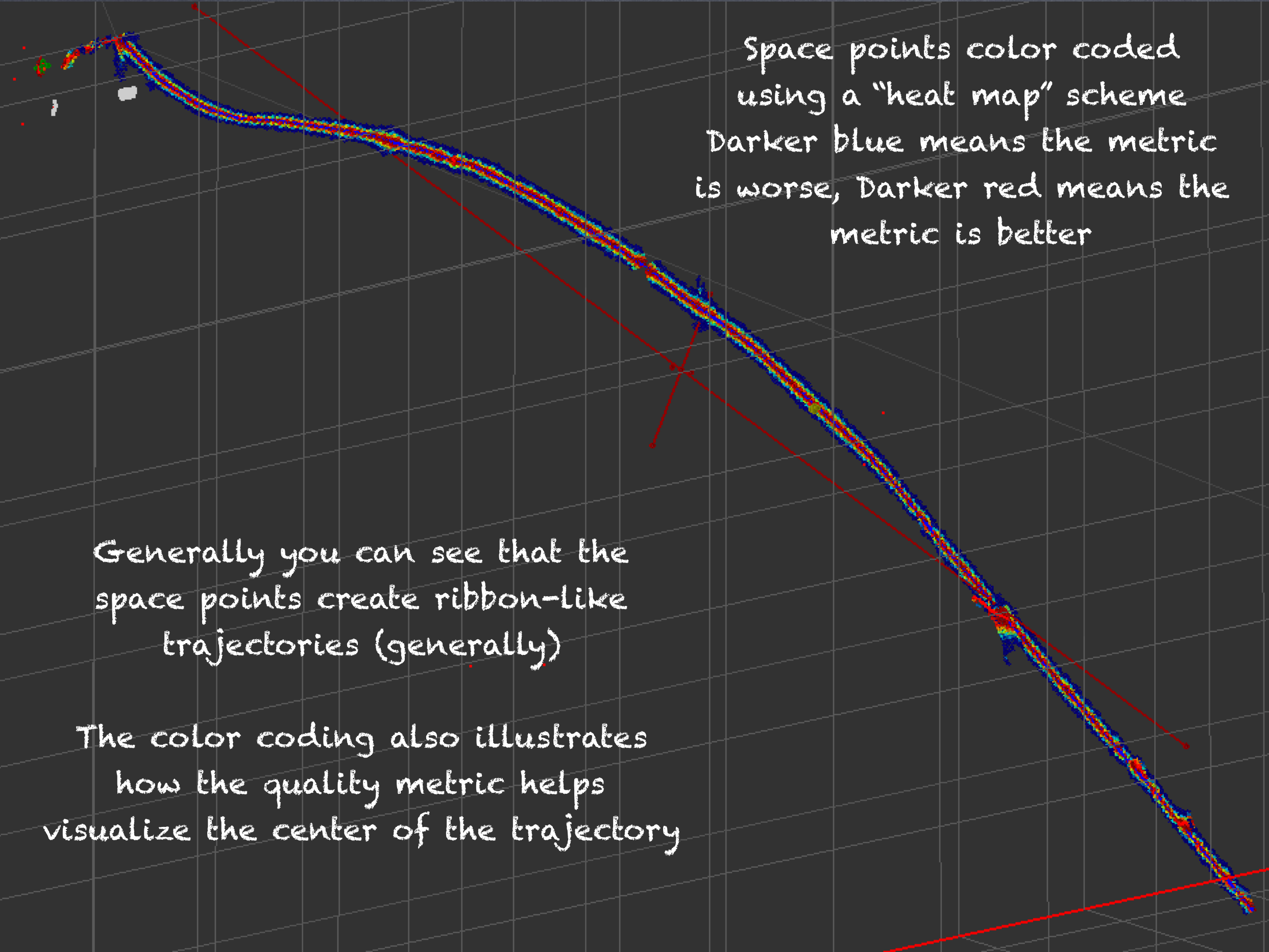
# Space Point Quality Metric



- Build a quality metric which can be useful in the downstream reconstruction:
  - First compute weighted average time of the three 2D hits, using this and the widths of the hits, form the sum of the squares of the "pulls" of the three hits
  - Can reject those with outright "bad" chi-square values
  - Can be very useful in downstream disambiguation

# Example Displays

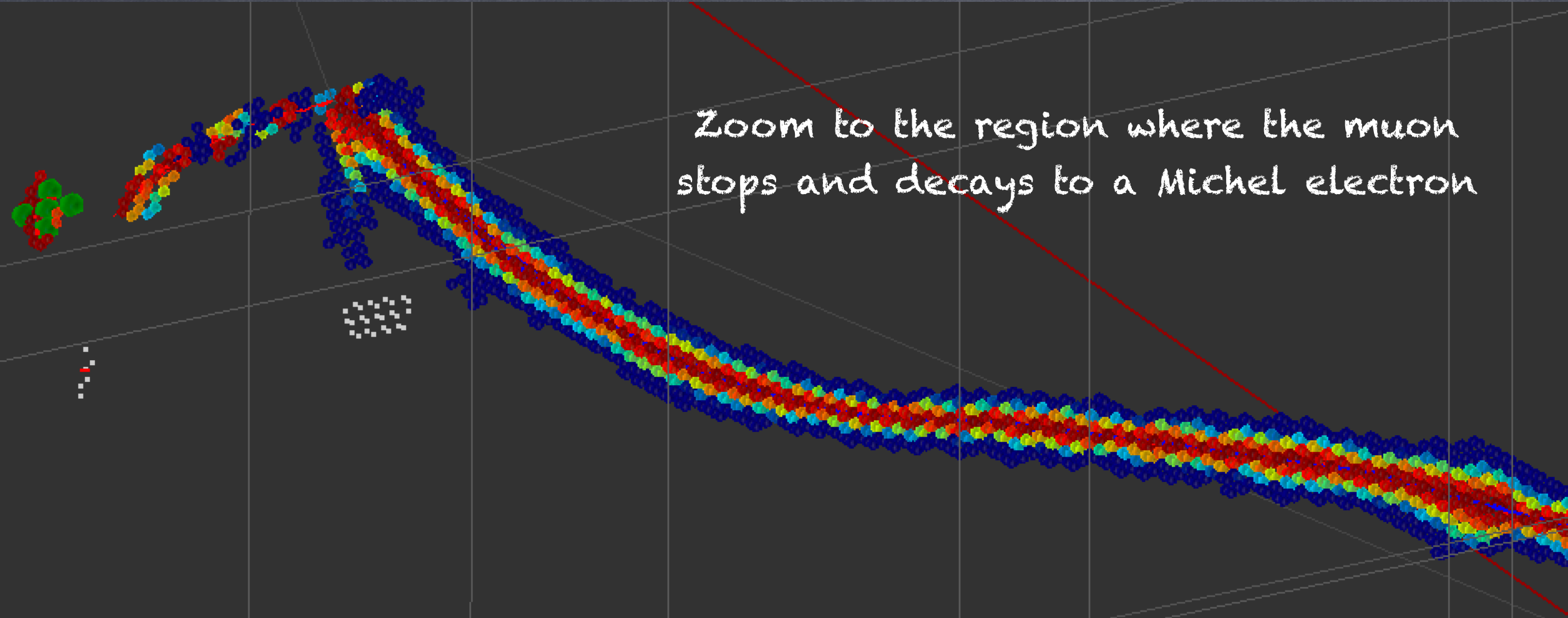
- Following example event displays utilize the ICARUS TPC simulation/reconstruction
  - Space point finding works with other TPCs
    - e.g. was originally developed using MicroBooNE
    - Also handles bad channels
    - ICARUS is a more interesting example because it is a multi-Cryostat and multi-TPC detector
- In the 3D event displays, space points are color coded according to the previously described metric
  - Using a "heat map" - better values of metric are at the red end of the spectrum, worse at the blue end



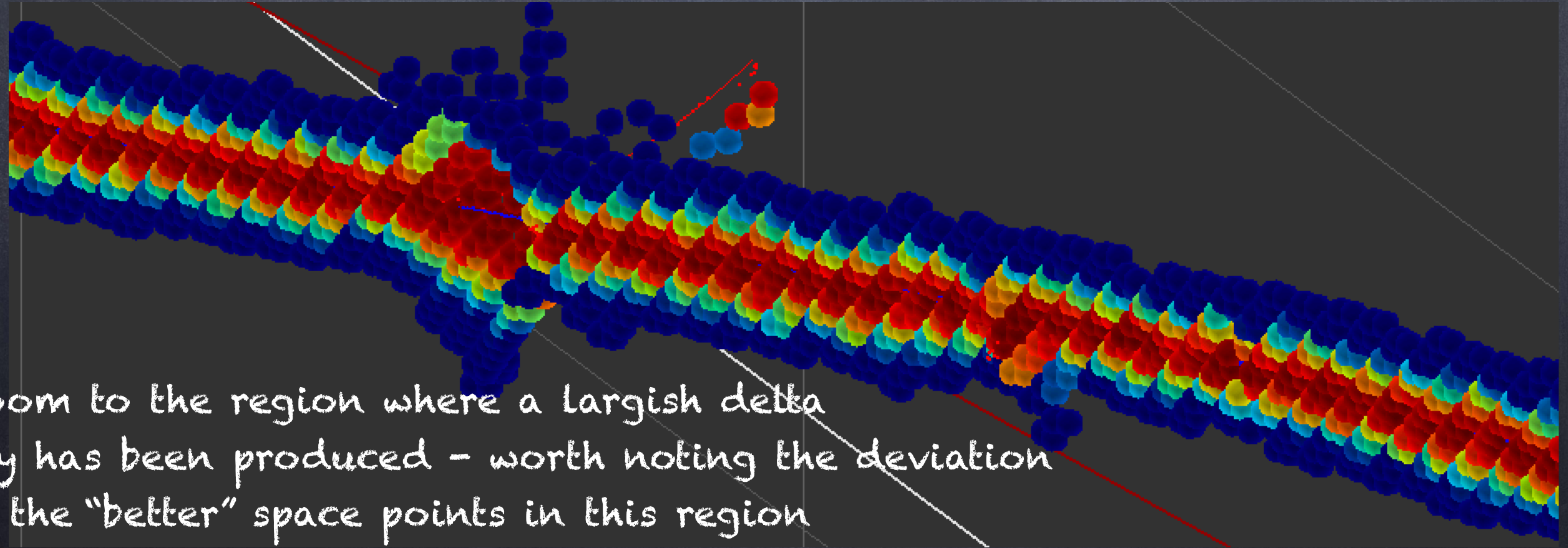
Space points color coded  
using a "heat map" scheme  
Darker blue means the metric  
is worse, Darker red means the  
metric is better

Generally you can see that the  
space points create ribbon-like  
trajectories (generally)

The color coding also illustrates  
how the quality metric helps  
visualize the center of the trajectory



Zoom to the region where the muon stops and decays to a Michel electron



Zoom to the region where a largish delta ray has been produced - worth noting the deviation in the "better" space points in this region

Build Space Points:  
Space Point Solver

# Space Point Solver

- Developed by Chris Backhouse ~Spring/Summer 2017
- Goal here is to prefer building of space points from the correct combinations of 2D hits - reduce the level of ambiguity from the simple approach
  - Starting point is the set of "simple" space points
    - SpacePointSolver uses different metrics for making initial 3D points
  - Then try to resolve ambiguous space points using the charge information of the hits
    - Assume the collection plane charges have the "true" deposition at that time and position
    - Employ a minimization technique to distribute this charge among the matched induction plane hits
- Originally motivated by the WireCell though use of 2D hits leads to a different and unique implementation

# Resolving Ambiguities

$$\text{minimize } \chi^2 = \sum_i^{\text{iwires}} \left( q_i - \sum_j^{\text{sites}} T_{ij} p_j \right)^2$$

$q_i$  = charge observed in the  $i^{\text{th}}$  induction hit

$T_{ij} = 0$  or  $1$ , does site  $j$  contribute to hit  $i$ ?

$p_j$  = charge predicted at site  $j$

subject to  $p_j > 0$  for all  $j$  and  $\sum_j^{\text{sites}} U_{jk} p_j = Q_k$  for all  $k$

$U_{jk} = 0$  or  $1$  – does site  $j$  contribute to collection wire  $k$ ?

$Q_k$  = total charge in the  $k^{\text{th}}$  collection hit to be distributed

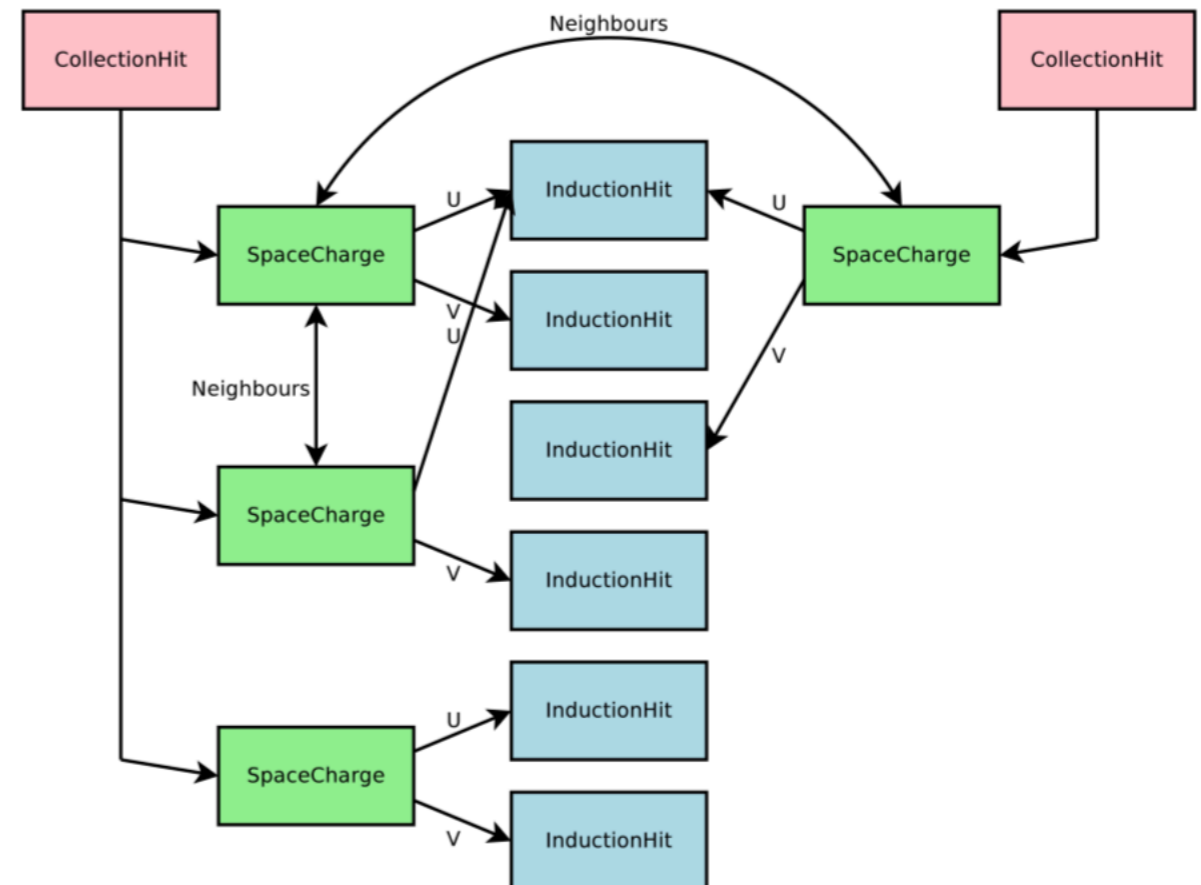
# Further Improvement

- ▶ Remaining freedom while maintaining a convex problem, add terms

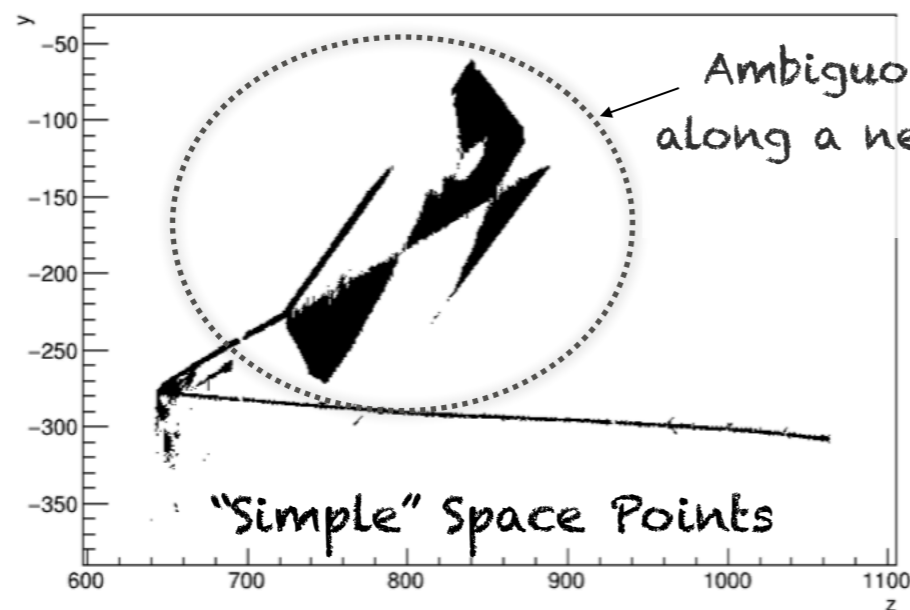
$$\chi^2 \rightarrow \chi^2 - \sum_{ij}^{\text{sites}} V_{ij} p_i p_j$$

- ▶ A term like  $\sum_i p_i$  has no effect due to overall charge conservation
- ▶  $V_{ij}$  terms amount to L2 regularization – prefer one large hit to two small ones

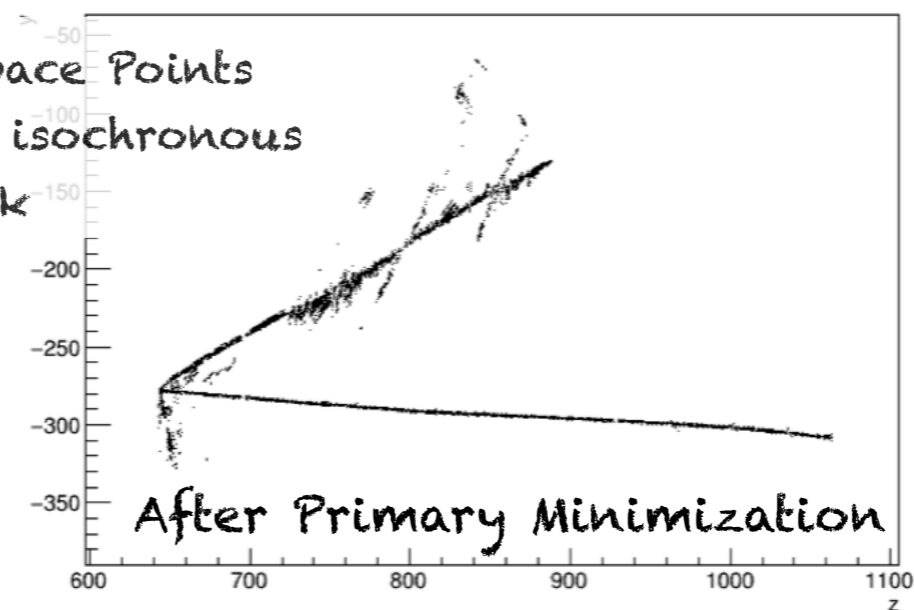
- ▶ Pick  $V_{ij} = 0.15e^{-r_{ij}/2\text{cm}}$  arbitrarily
- ▶ Rewards proximity of charges
- ▶ Overall strength set by trial and error



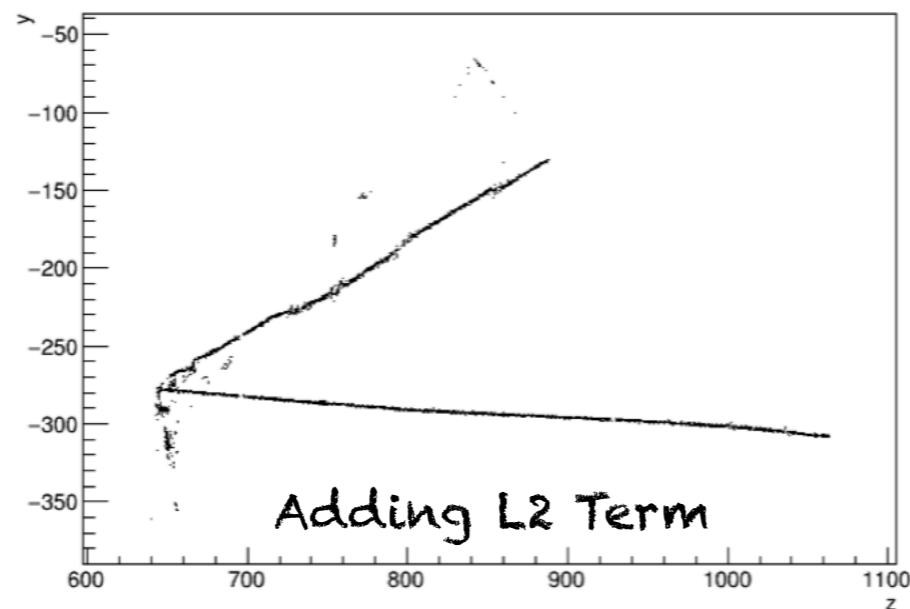
Nice example event with a nearly isochronous track to illustrate the impact of the minimization and regularization



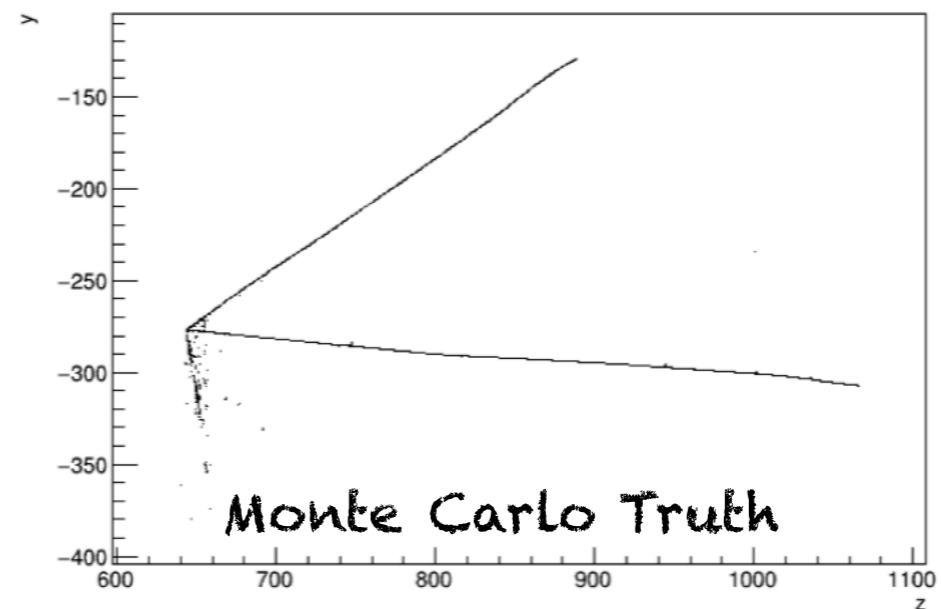
(a) All coincidences



(b) Without regularization



(c) With regularization



(d) True charge distribution

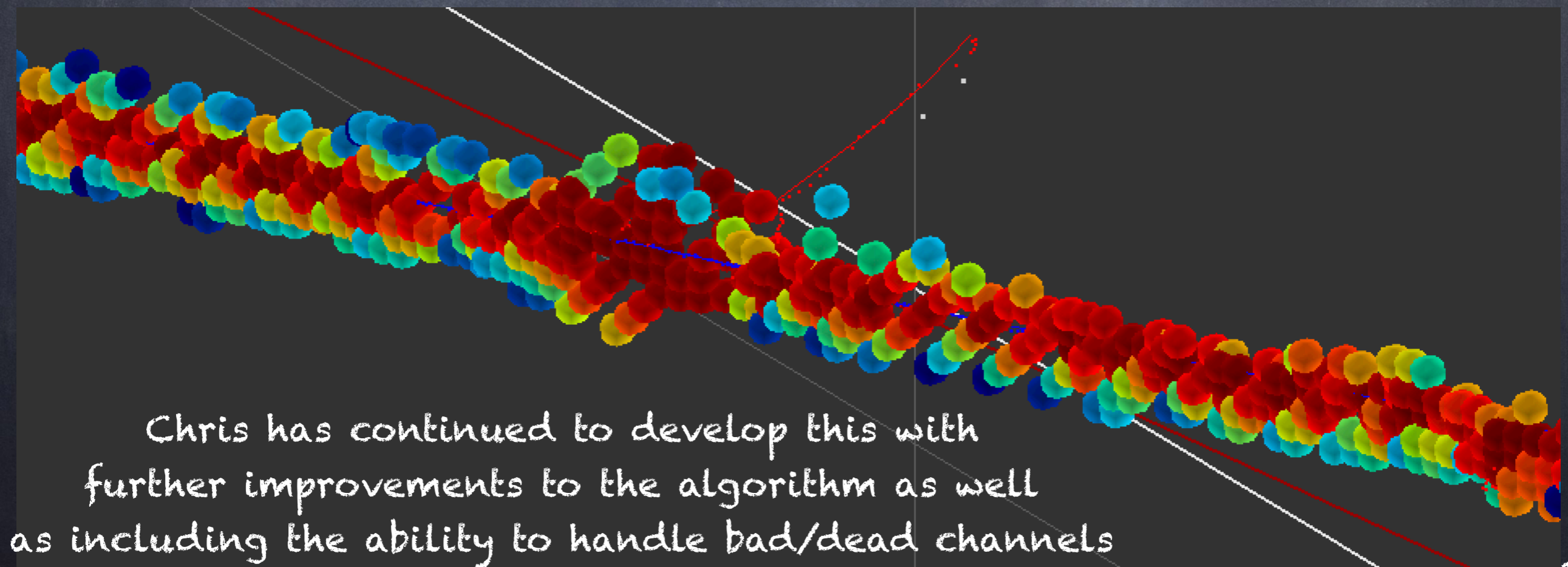
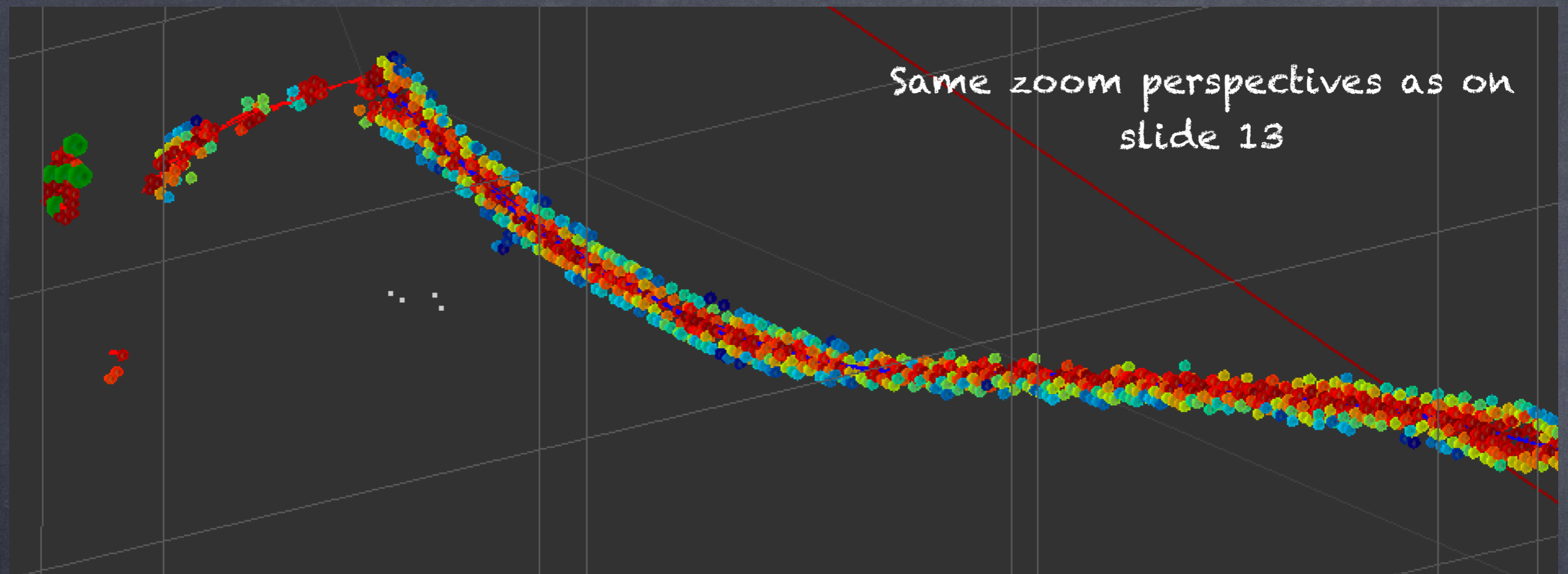
# Space Point Solver

- Chris Backhouse has made several presentations on this:
  - LArSoft Coordinating Meeting August 27, 2017: [Link](#)
  - Handling bad channels June 8, 2018: [Link](#)
- SpacePointSolver module is implemented in LArSoft with "standard" LArSoft data structures as output
  - Relatively straightforward to incorporate SpacePointSolver into the Cluster3D framework
  - Which makes it easy to compare against the standard Cluster3D



Same Event as on Slide 12  
Now drawn with Space Points  
Made with SpacePointSolver

Generally, the created Space Points  
now from a much narrower "ribbon"  
than those from the simple approach



# Event Slicing

# What is Event Slicing?

- Surface LArTPCs present two main problems for 2D reconstruction algorithms
  - Large number of tracks from cosmic ray background means significant volume of data to process - uses significant cpu/memory
  - Worse, cosmic ray tracks and debris often overlap signal events and require significant special handling - leads to misconstruction and efficiency problems
- When viewed in 3D the phase space for overlap is significantly reduced
  - An algorithm that can look at 3D space points can separate out the information specific to each of the interactions in the TPC
  - Can significantly improve performance - both cpu/memory and overall efficiency

# How to Slice an Event

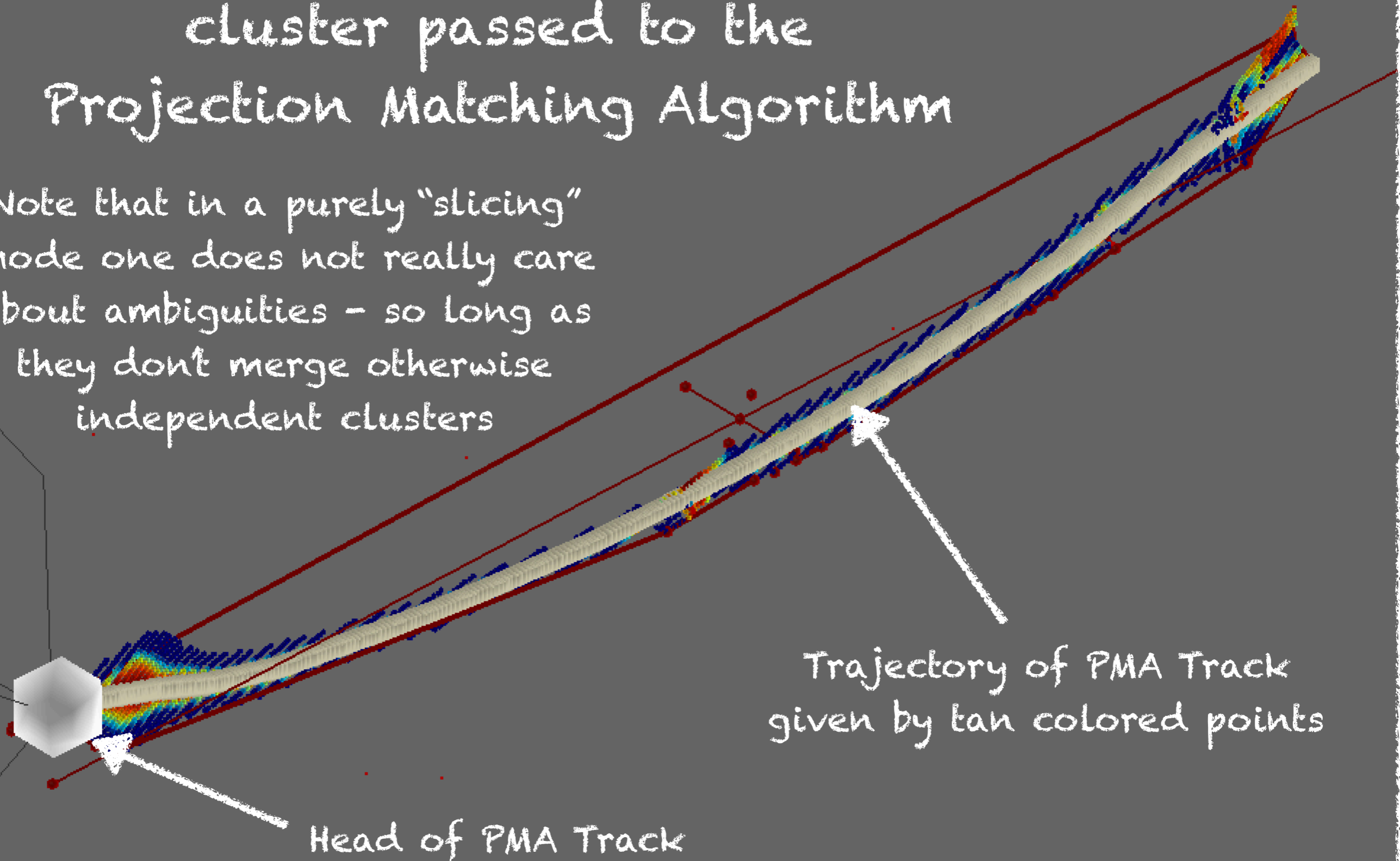
- Start with a source of 3D hits
  - Currently either simple or SpacePointSolver
- Cluster formation
  - Initial clustering of space points - DBScan/kdTree
  - Initial "analysis" of the event via Principal Component Analysis
  - Enables Merging "matching" clusters, arbitration of ghost clusters, general cleanup
- Resulting clusters represent "slices" of the overall event and the corresponding 2D hits for each slice can now be input to a 2D reconstruction algorithm
- Important to note that 2D reconstruction algorithms don't care about 3D space point ambiguities
- Note: A similar approach developed by Tingjun using SpacePointSolver for 3D hits, DBScan for clusters, slices then input to TrajCluster

# Projection Matching Algorithm

- The Projection Matching Algorithm (PMA) was originally developed to determine the trajectory of tracks based on 2D hits in each projection
  - Concept of Principal Curves: here it simultaneously minimizes the projection of a candidate track to hits in each view but, importantly, does not try to get the order of hits correct between the views
- It has since been developed to include "micro scale" pattern recognition
  - Example: given all the hits associated to a CR muon (and its daughters) it should return the CR track trajectory and any delta rays that are also "trackable".
- This is the situation after the first round of clustering with the 3D clustering
  - All hits associated to a common object are in the "super cluster", can use the PMA to do the fine level pattern recognition to reconstruct the event
  - Well... this is currently MOSTLY true... some work needs to be done here...
- It does not care about ambiguous 3D hits

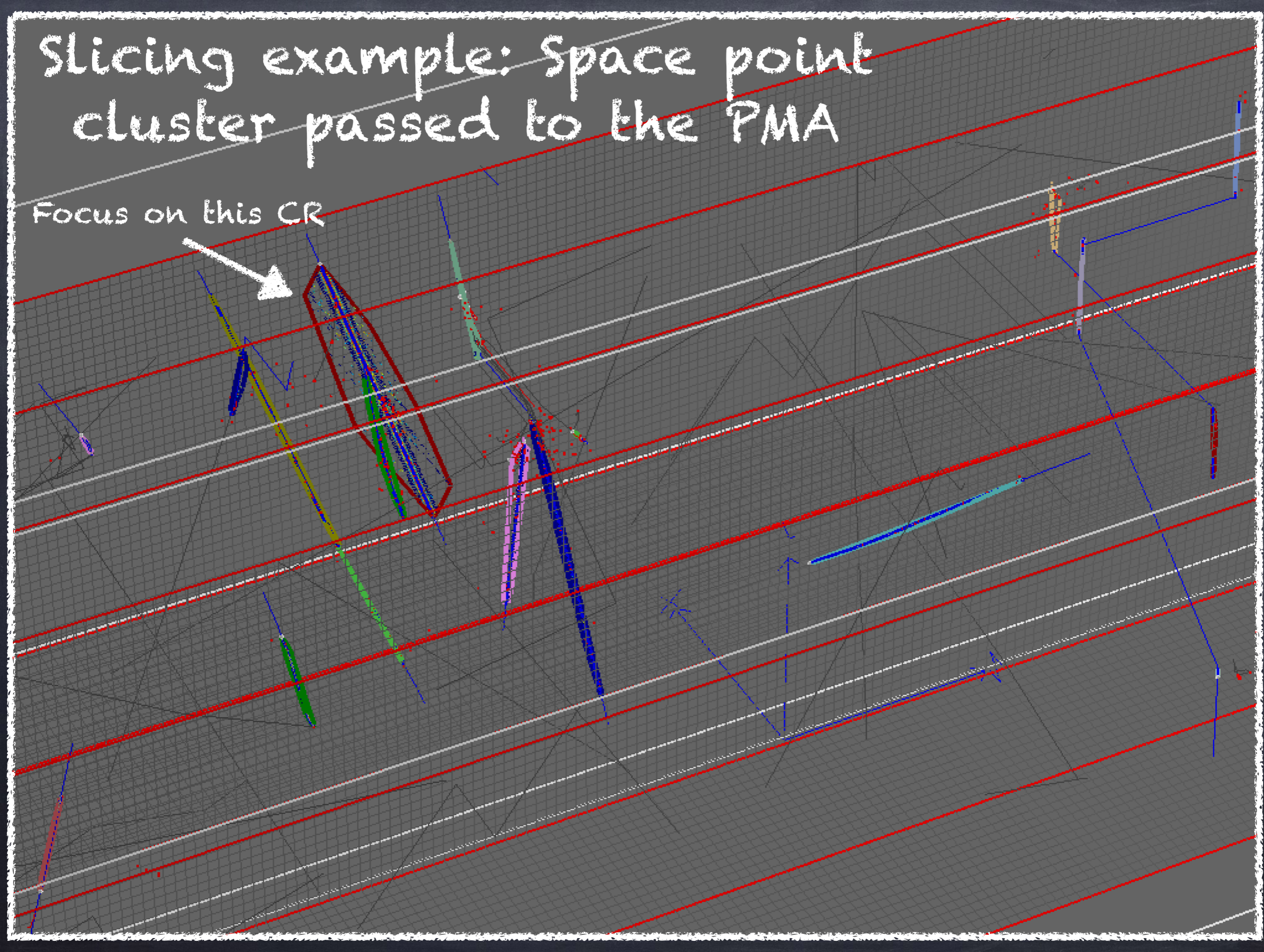
# Slicing example: Space point cluster passed to the Projection Matching Algorithm

Note that in a purely "slicing" mode one does not really care about ambiguities - so long as they don't merge otherwise independent clusters



Slicing example: Space point cluster passed to the PMA

Focus on this CR

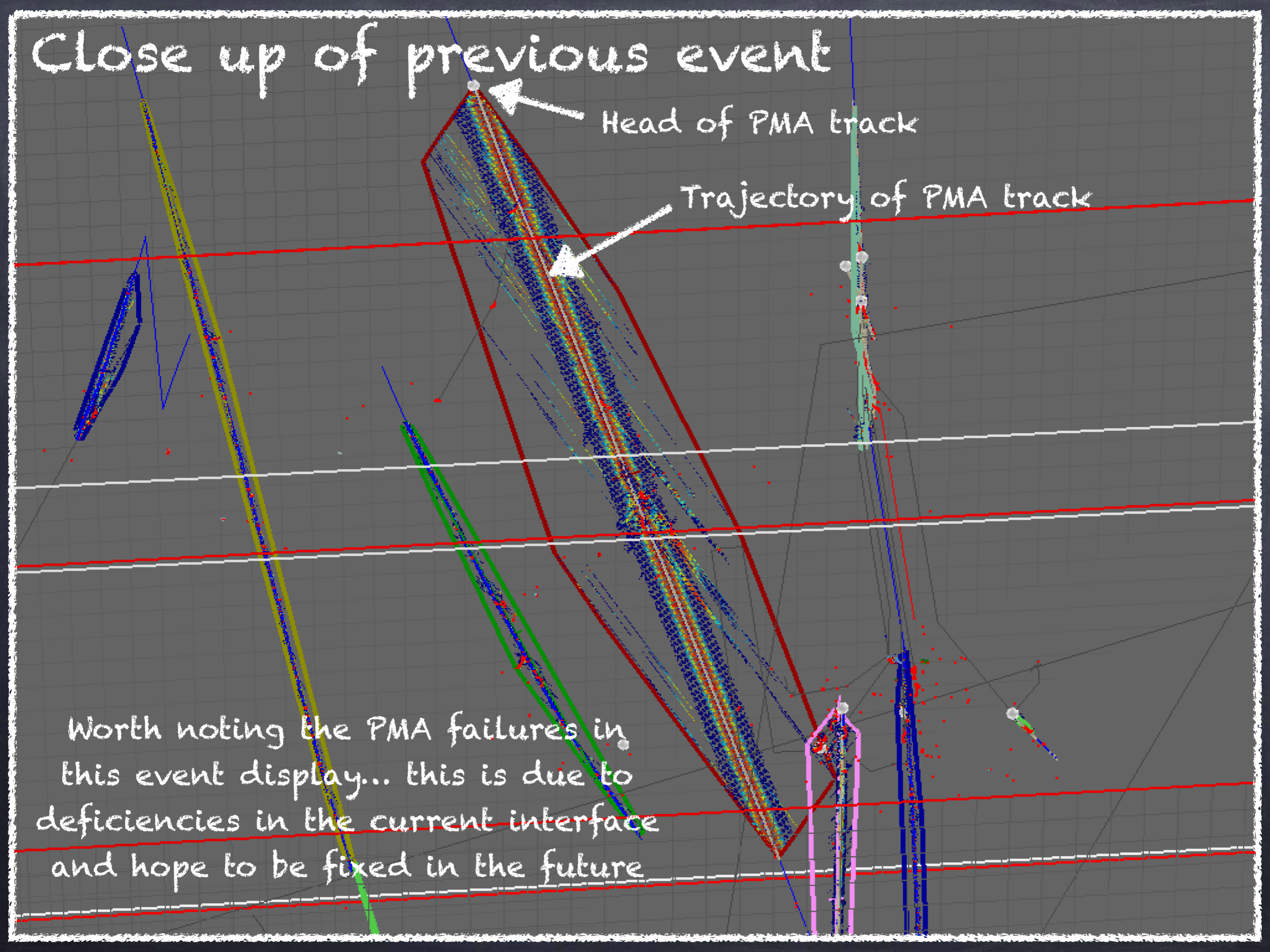


# Close up of previous event

Head of PMA track

Trajectory of PMA track

Worth noting the PMA failures in this event display... this is due to deficiencies in the current interface and hope to be fixed in the future



# Example of Cluster Merging with PCA Working Across TPC Volumes



Single isotropic Muon Simulation

# Towards Pure 3D Pattern Recognition Techniques

# Toward 3D Pattern Rec

- Ultimate goal is to have a purely 3D pattern recognition approach to solving the problem
- The recently rekindled effort in this area has focused on tracking with an idea to create a purely 3D version of the Projection Matching Algorithm
  - Underlying concept of "principal curves"
- This is facilitated by trying to adapt concepts from Computational Geometry:
  - Principal Components Analysis
  - Convex Hull - making use of defect points
  - Eventually hoping to employ Voronoi diagrams

# Current Algorithm Flow

- Step 1: Start with a sliced 3D cluster
- Step 2: Compute the Convex Hull
  - Computed in 2D by projecting the 3D points into the plane of maximum spread as defined by the Principal Components Analysis
    - 2D for now to facilitate visualization, this should be made 3D in the future
  - With the hull, use the "defect" points to identify candidate kinks in the trajectory
    - Use the angle of the two edges to identify kinks

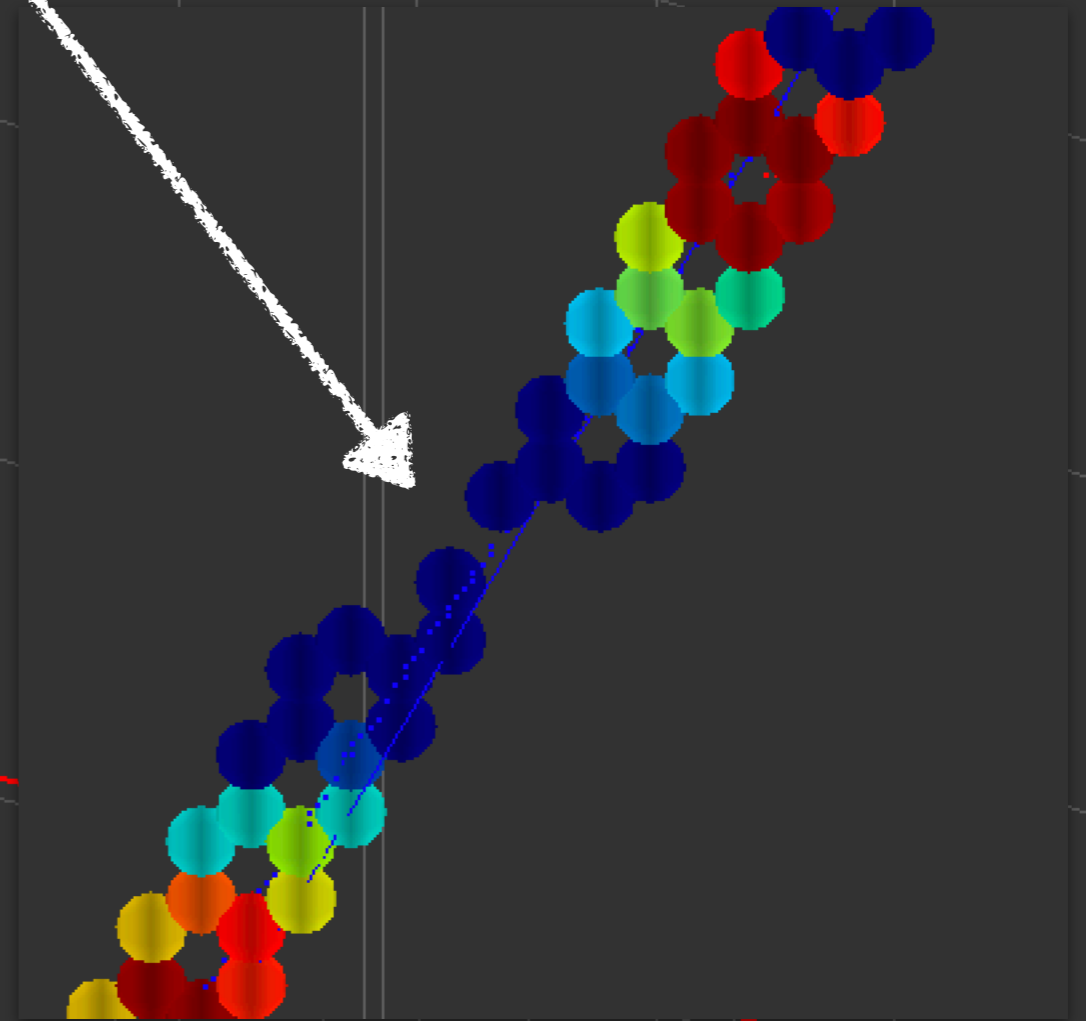
# Simulated Muon in ICARUS

## Principal Components Axes

3D Space Points displayed  
color coded by a measure  
of their "goodness":

darker red - "better"

Darker blue - "worse"



# Simulated Muon in ICARUS

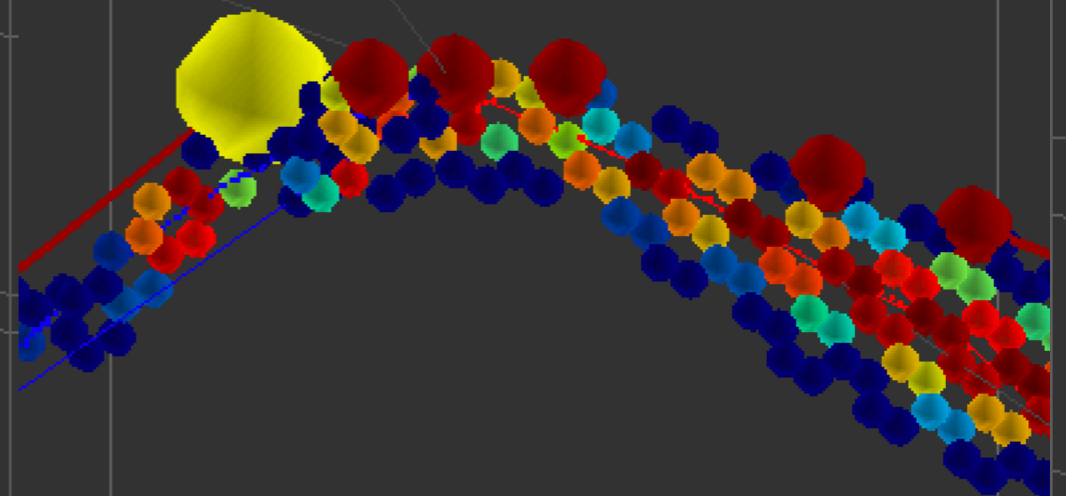
Project points to plane of  
maximum spread by PCA,  
Then compute the convex hull

Convex Hull  
Represented by red  
polygon enclosing cluster

Large (sort of) red balls  
are "defect" points of the  
convex hull

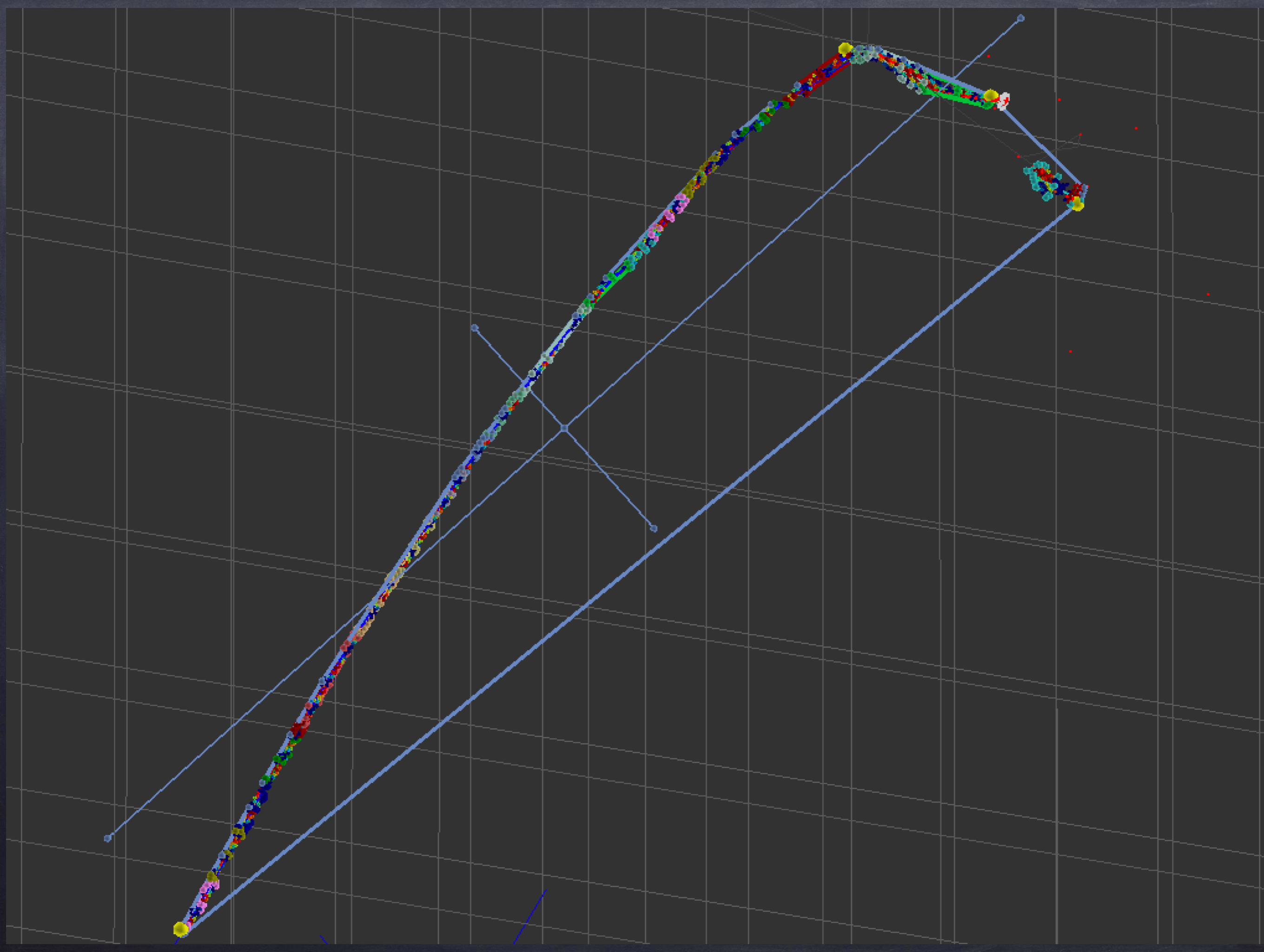
Ambiguous hits mean kinks can  
get rounded, kink points ultimately  
represent centers of interest and  
will need further investigation

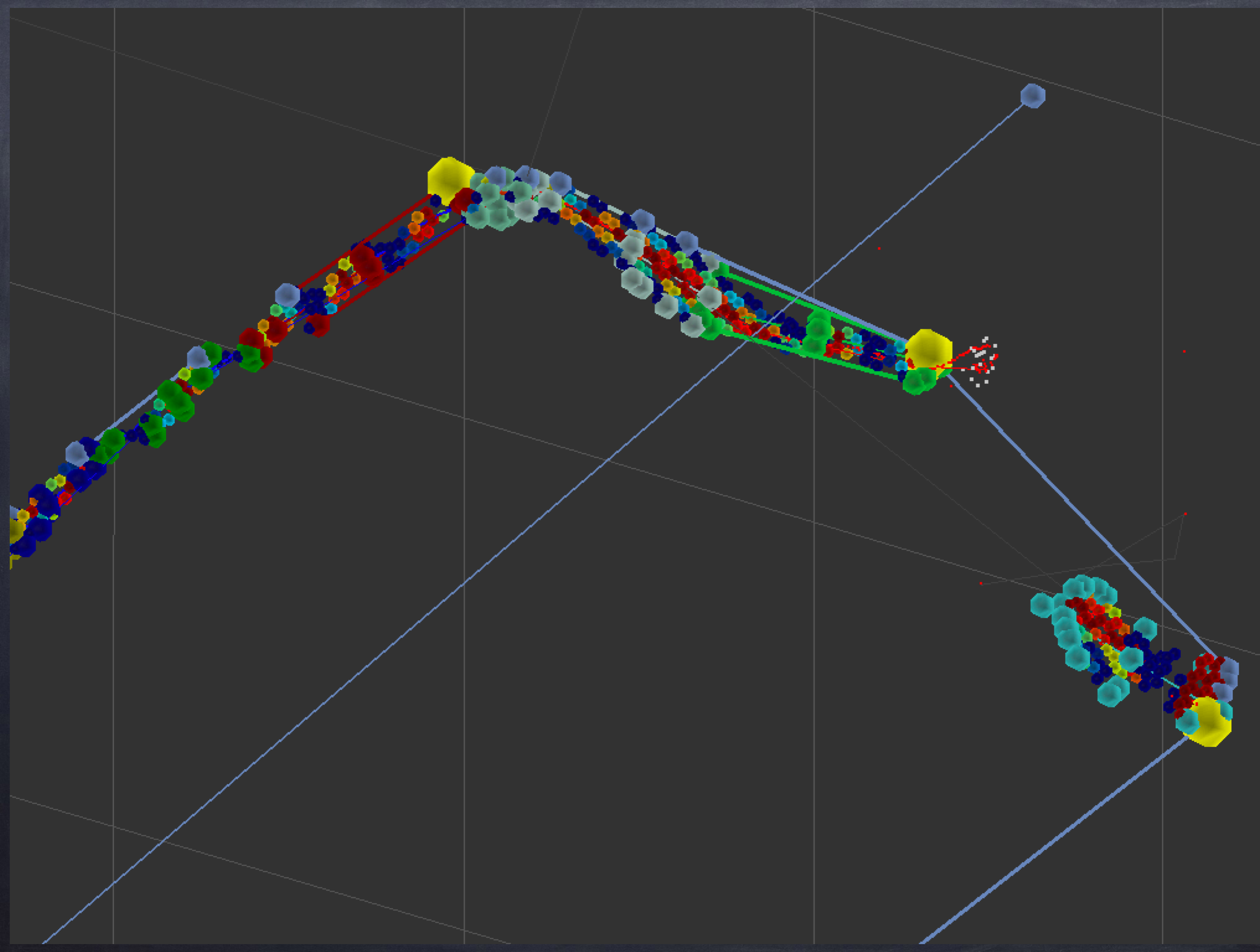
Large yellow balls are  
kink points on the hull



# Now the Real Work Begins

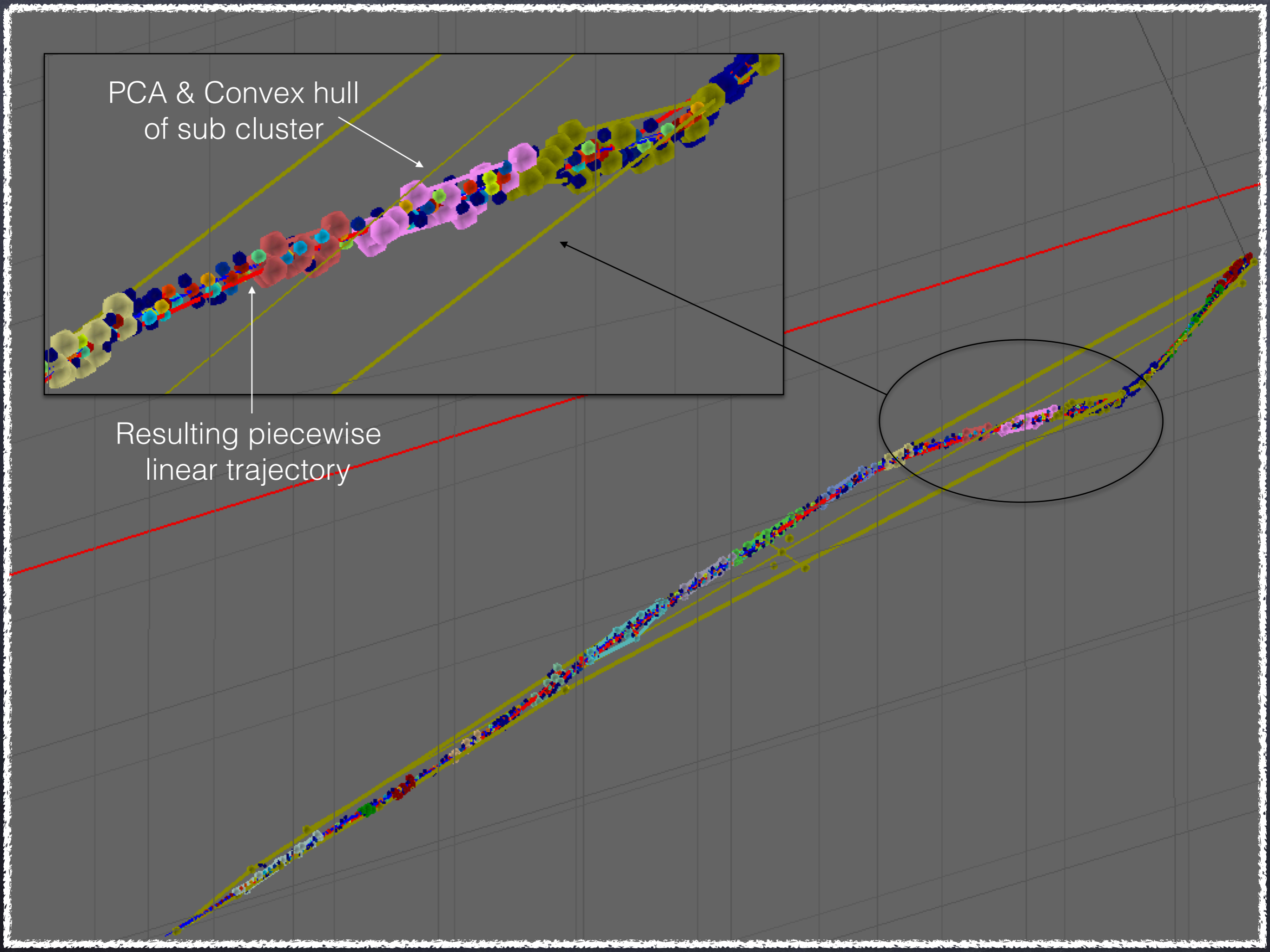
- Now have a gross picture of our cluster
  - Step 3 is to start breaking it into smaller pieces.
- Possible methods:
  - Break at kink points (if any)
  - Break at defects points with large deviations from the primary PCA
  - Break at "large" gaps
  - If lots of hits simply break in half
- Employ a recursive algorithm:
  - Break cluster by one of the above into two components
  - Recompute PCA, convex hull and find kink points for each
  - Rinse and repeat until "no further gain"





PCA & Convex hull  
of sub cluster

Resulting piecewise  
linear trajectory



Drawing the individual primary principal components  
axes of the subclusters in a connected fashion

Red curve: Linked PCA's

Blue curve: the simulated particle trajectory

Note that at this point we have made NO  
assumptions about the ordering of the 2D Hits  
or 3D Space Points

# 3D Techniques

## Status and Plans

- So far approach seems very promising for developing track trajectories and finding the main kink points
- This only increases the To Do List:
  - 2D convex hull to 3D, better kink resolution?
  - More powerful computation geometry techniques?
    - My favorite is Voronoi Diagrams
  - Refinement of kinks - find actual vertices
  - How will this work for shower reconstruction?
- The main problem?
  - Time, particularly before Deep Learning overwhelms all of this...

Extra Stuff

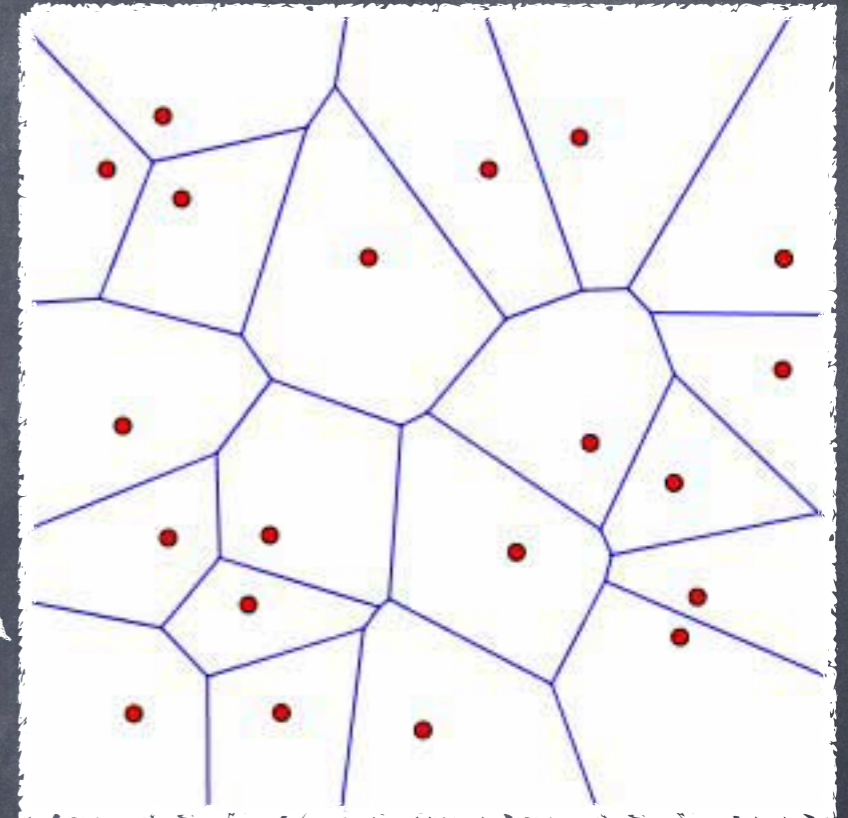
(Not For The Faint of Heart)

# Where Next?

- Two observations
  - When viewing the event displays it is intriguing to note the path defined by the 3D hits with good (small) values of the metric
  - Also note the density of hits along the particle path
- Is there a way to try to "follow" the path more directly?
  - Can use the end points of the convex hull as path finding end points
  - What might be useful for looking at the density and linking hits together to form a path?

# Voronoi Diagram

- In a nutshell - given a set of "site points" the voronoi diagram consists of the collection of voronoi cells, each cell is bounded by the set of points that are equal distance from the interior site point and each of its nearest neighbors
- Turns out to be very fast to calculate -  $n \log n$  in the site points
- What is it good for?
  - Fast lookup of nearest neighbors
  - Finding high density regions within a given collection of points
  - Can quickly (linear time) get the convex hull
  - Can quickly compute the minimum spanning tree
  - Can use to do "nearest neighbor" averaging...
- Investigate if a good starting point for a 3D pattern recognition algorithm



# Implementation in Cluster3D

- Can compute the Voronoi Diagram in the Cluster3D code
  - Unfortunately, the LArSoft event display has defied all attempts to actually display it!
- Previously had explored using the MST in Cluster3D
  - Followed path using a distance metric but did not handle curvature with ambiguous hits well
  - Using Voronoi Diagram, select nearest neighbor with best metric.