

GArSoft Simulation and Tracking

Tom Junk

DUNE LArTPC Pixel Workshop

September 29, 2018

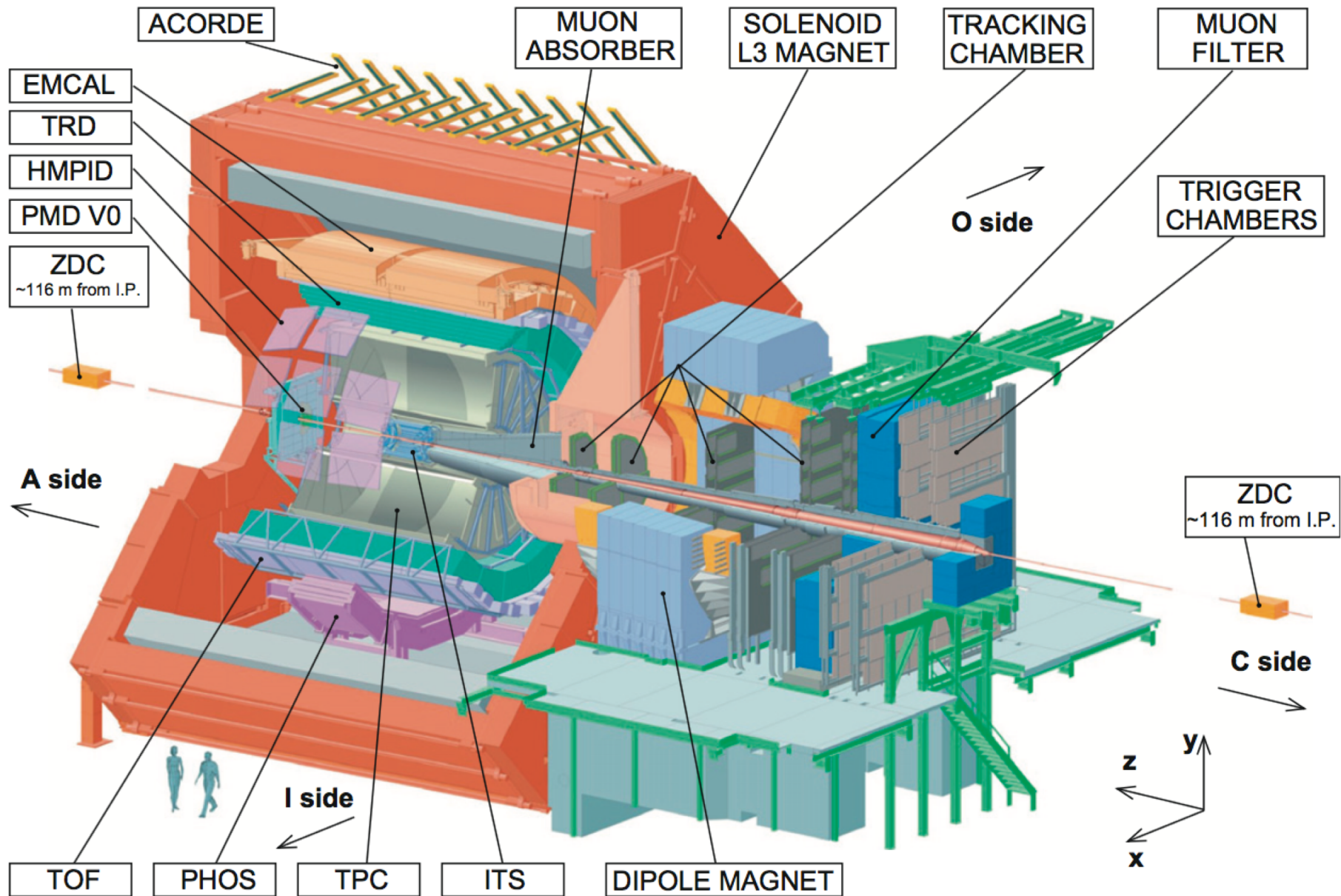
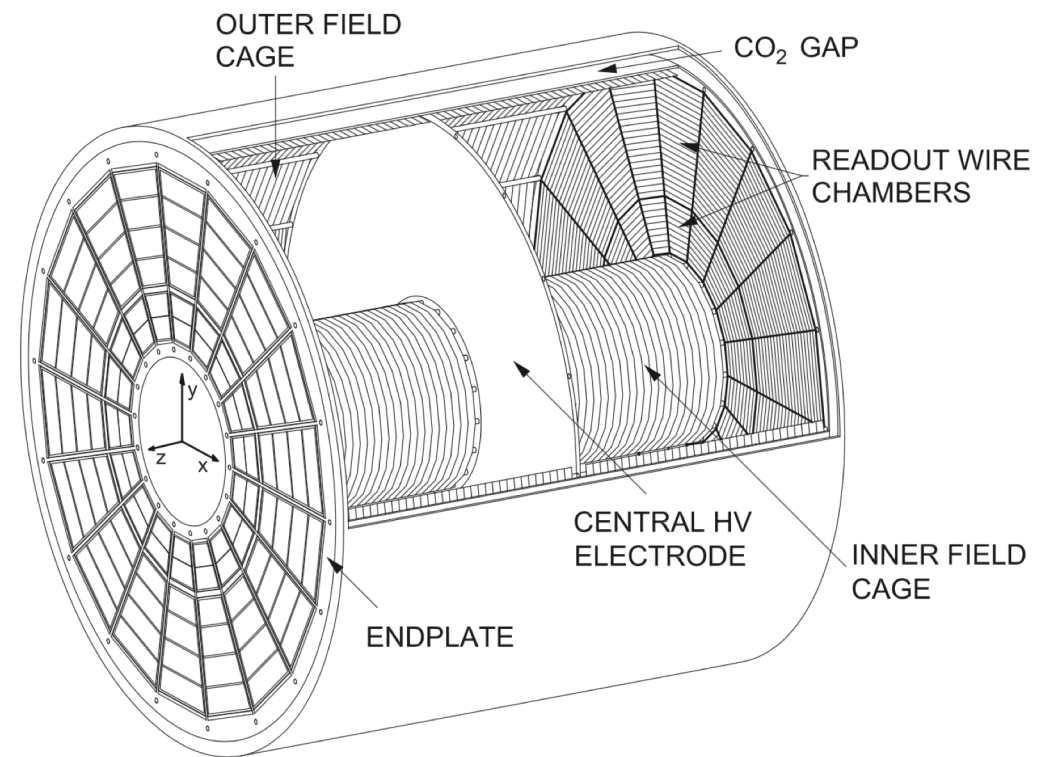


Fig. 1. ALICE schematic layout [2].

The ALICE TPC

- 18 Sectors, two sides. Anodes on either end, cathode in the middle.
- Outer radius: 2.58 meters, inner radius 0.788 meters (1.7 meters active).
- Length: 5.19 meters
- Gas 1 Atm Ne (+CO₂+N₂)
- B = 0.5 Tesla
- Alice will upgrade the readout chambers for HL-LHC.
- Pad readout: total 557,568 channels

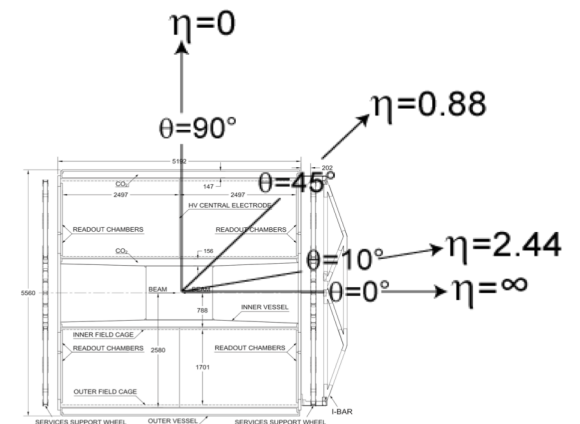
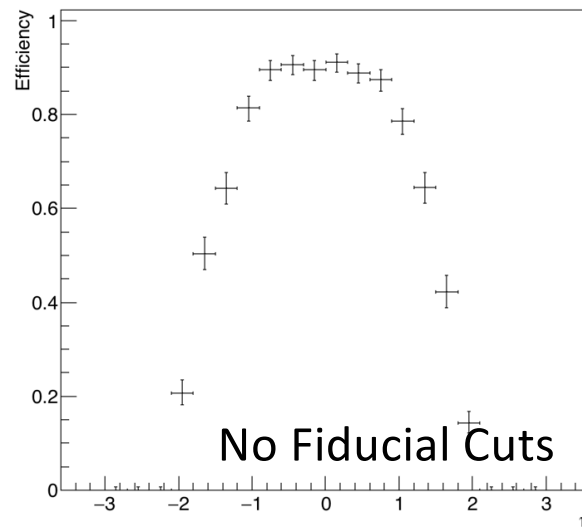
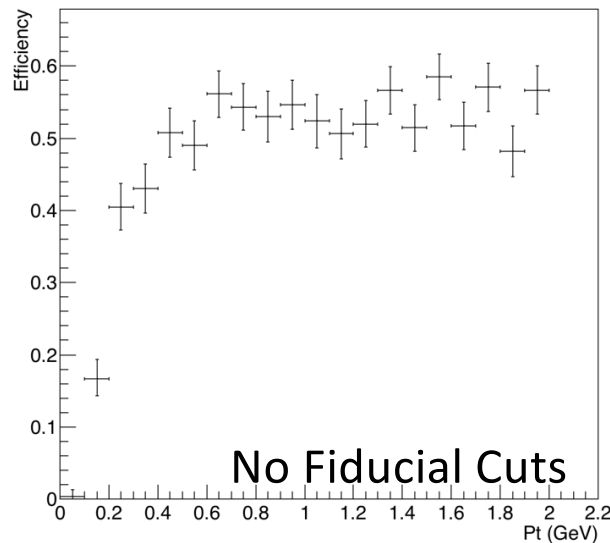


ALICE Software Performance: Tracking Efficiencies in p_T and η

using "beam-off" ALICE reco. Thanks to Ruben Shahoyan of ALICE for help!

arxiv:1507.07577 (high-mult cosmic-ray paper) 50-degree cut wrt zenith: $|\eta| < 1$

QA plots <https://twiki.cern.ch/twiki/bin/view/Main/QA:ChargedTracks> don't go out past $|\eta| = 1$



ALICE Software is optimized for pp and heavy-ion events. Isotropic efficiency and low- P_T reco not optimized

We Need our Own Software

- ALICE software is great
 - GEANT4 (*and* GEANT3!)
 - Calibrated with data, used for physics
 - Includes lots of subtle effects
 - Efficient, compact
- But it doesn't do what we need
 - We need 10 bar Ar (c.f. 1 bar mostly Ne)
 - Hole in the middle needs to be filled
 - We need to reconstruct low-momentum tracks
 - We need to reconstruct tracks pointing in arbitrary directions, starting anywhere in the chamber
 - We need to integrate it with the LAr Pixel detector, the ECAL, 3DST, and muon catchers.

Ways Forwards

1. Fork ALICE software and develop it as our own

- Great starting point
- Terminology and techniques differ from FD (and ND) (e.g. doesn't use *art*). It's just based on ROOT
- May find some things targeted at ALICE physics baked in very deep in the code.
- More tricky to use different framework from the Pixel LArDetector
- I/O structure mixed in with calculation

2. Modify LArSoft

- PixLAr ran into computer resource issues. LArSoft likes to save `raw::RawDigits` for example
- LArSoft's G4 interface is hard to modify away from the rectangular prism (Difficult to add the CPA frames to the ProtoDUNE geometry for example as they cut small notches in the active volume)
- Data products aren't appropriate. Tracks e.g. are helices and not just arbitrary hit collections.
- Need a fully-simulated ECAL

3. Build Our Own

- STT did this – HISOFT (T. Alion et al). Repo: `dunefgt`
- Lots of work
- Reco algs would have to be all our own anyway, even in options 1 and 2
- Brian Rebel already went down this path: GArSoft!

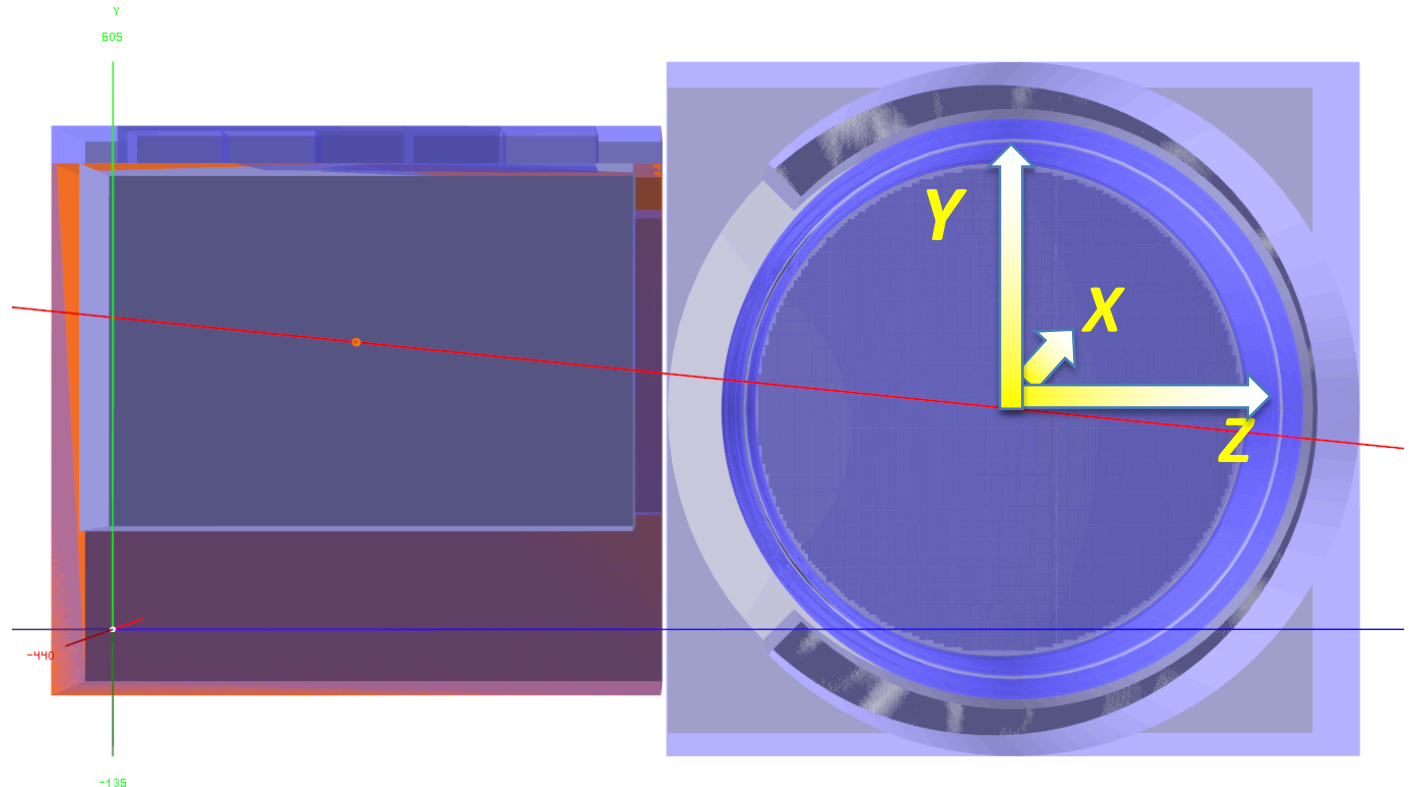
Software Status

- Simulation is with GEANT4.
- GENIE and single-particle interface
- Data products produced:
 - MC Particles (and MC Truth for the neutrino interaction)
 - Simulated energy deposits (just newly introduced into LArSoft, in GArSoft from the beginning)
 - Raw Digits (waveforms on each pad) – zero suppressed by default (as ALICE does)
 - Hits (in LArSoft terminology) – reconstructed pulses
 - Tracks
 - Vertices
 - Other data products, like flux and trigger simulations have initial placeholders but no work yet on them.
 - Analysis tree module written and tested. Saves: MCTruth, MCParticles, hits, tracks, vertices
- 3D Event display shows: MC Particle trajectories, raw digits (thresholded), hits, hits grouped on tracks.

Geometry

- From Mike Kordosky. Eldwan Brianne has been working on ECAL simulation in GArSoft using this GDML:

Coordinates:
X points along
E and B
Y points up
 $Z = X \times Y$



- GArSoft now can handle non-(0,0,0) locations of the center of the gas TPC

ALICE TPC Readout Plane

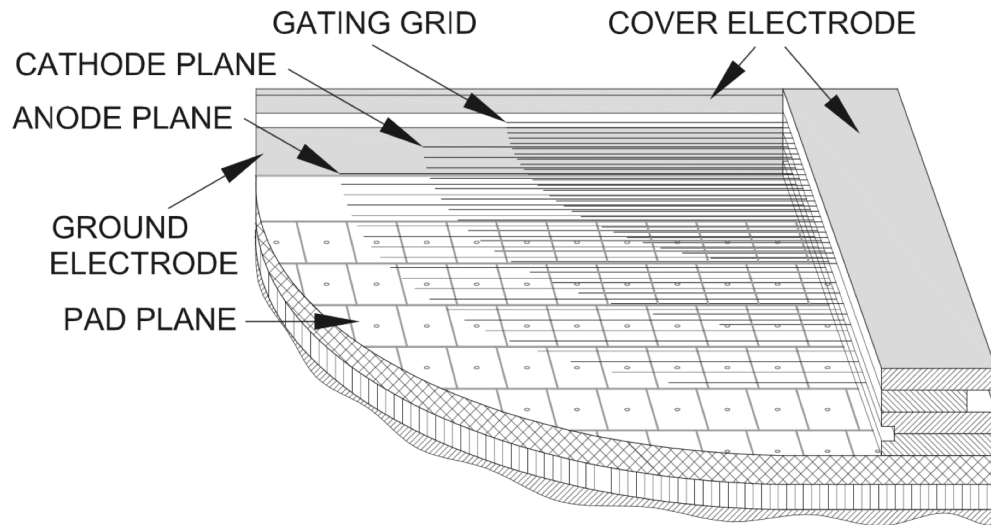


Fig. 9. Cross-section through a readout chamber showing the pad plane, the wire planes and the cover electrode.

For one sector
(1/18 of one side):

Inner readout chamber geometry	Trapezoidal, $848 < r < 1321$ mm active area
Pad size	$4 \times 7.5 \text{ mm}^2 (r\phi \times r)$
Pad rows	63
Total pads	5504
Outer readout chamber geometry	Trapezoidal, $1346 < r < 2466$ mm active area
Pad size	6×10 and $6 \times 15 \text{ mm}^2 (r\phi \times r)$
Pad rows	$64 + 32 = 96$ (small and large pads)
Total pads	$5952 + 4032 = 9984$ (small and large pads)

From the ALICE TPC TDR

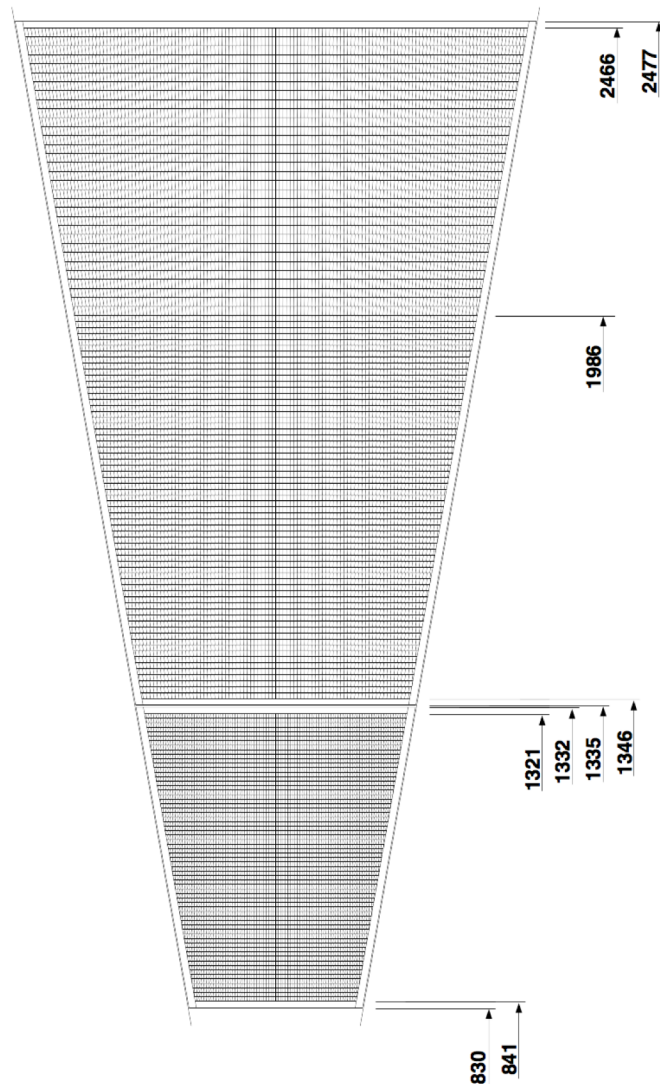
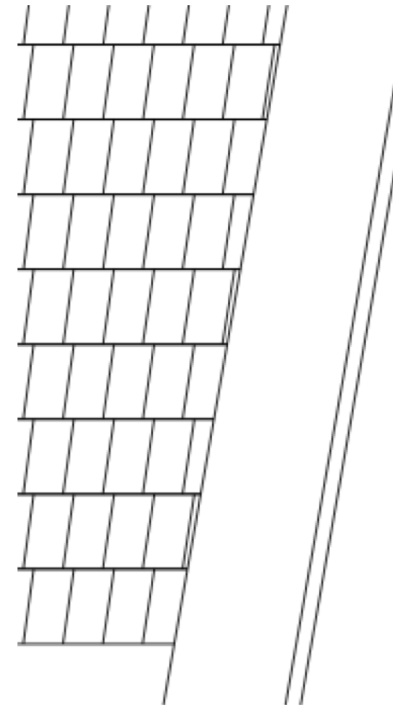


Figure 4.12: Pad layout of the ALICE TPC readout chambers. Distances from the beam axis are in mm.

Zooming in on the edge

Are these all
real channels?
Some are
awfully thin!



- I suspect these are just being blocked by the 12 mm cover electrode.
- There's a 3 mm gap between sectors (assume constant width). I defined channels all the way to the edge (going to have some extras).
- Partially obscured channels may collect less charge.

A First Attempt at a Channel Grid

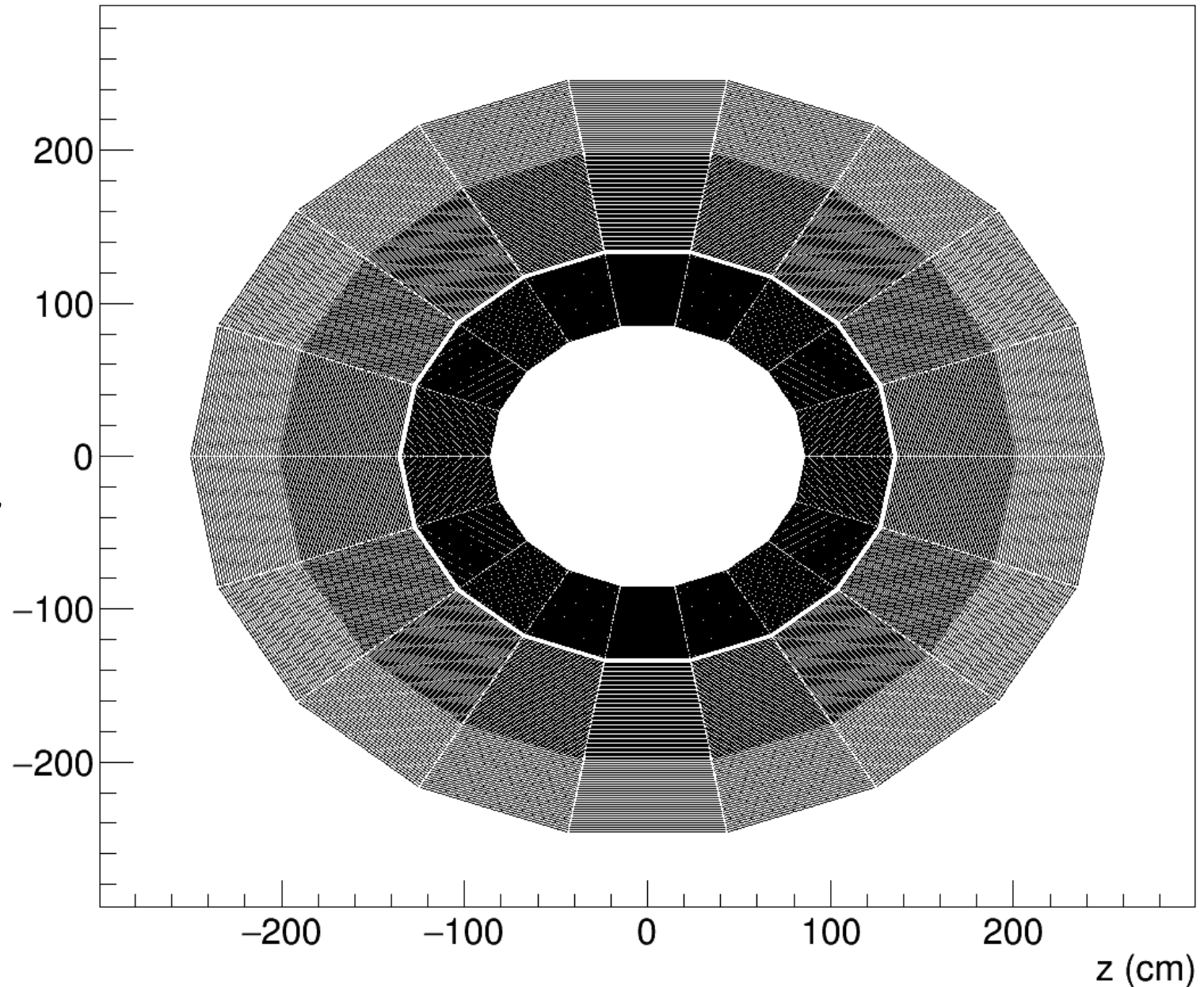
$292140 \times 2 =$
584280 channels y (cm)
(c.f. 557568 in
ALICE, or about 4.6
extra per pad row)

18 sectors

63 pad rows on
the inner chambers,

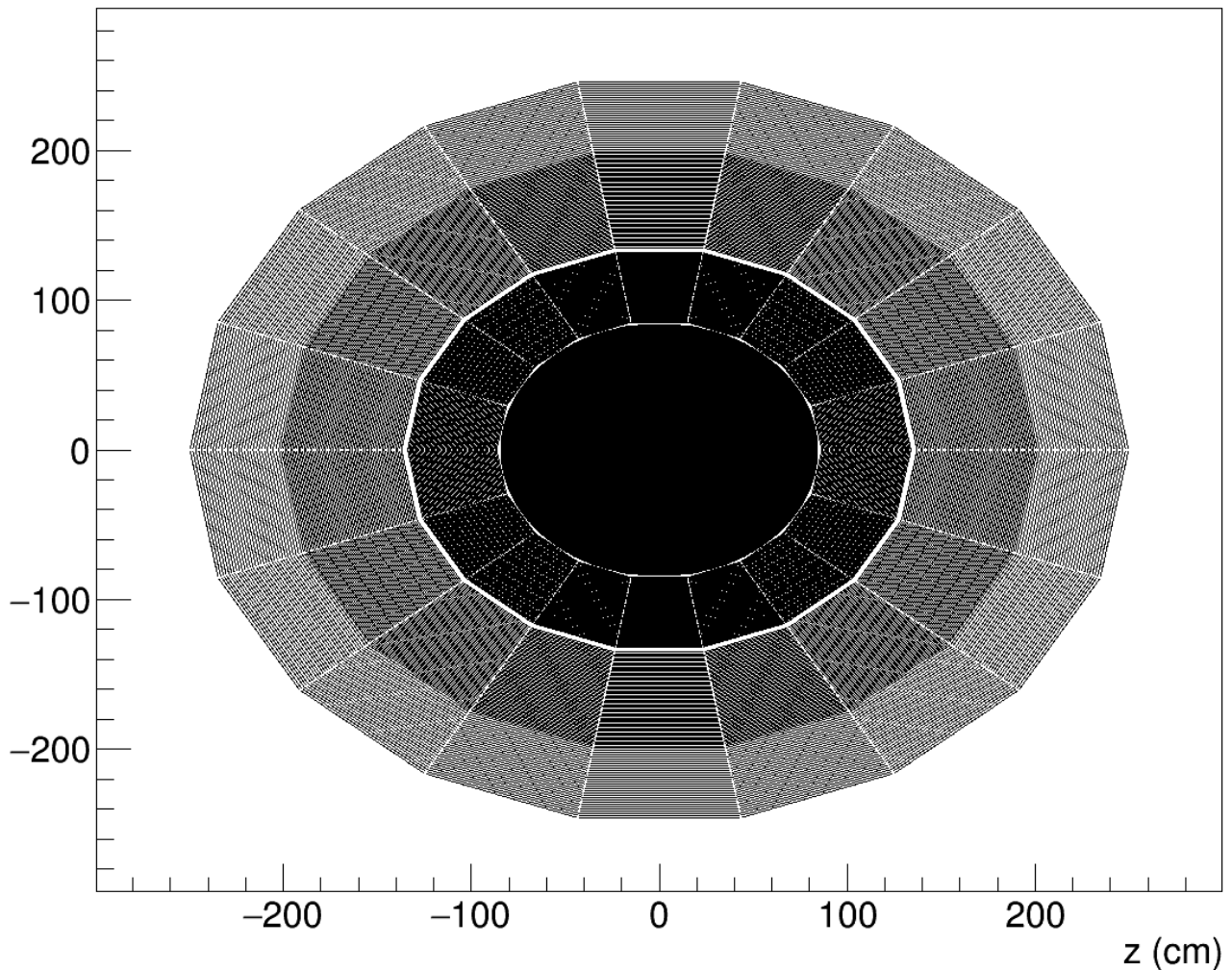
64 pad rows in
the inner outer
chambers

32 pad rows in the
outer outer
chambers



Filling the Hole

- Rectangular array of pixels in a disk
- Pixel size: 6mm x 6mm
c.f. 4 mm x 7.5mm for inner pad rows.
- Central disk has 62060 pads per detector side.
- Total channels per side is now 354200. About 18% of channels are in the disk.



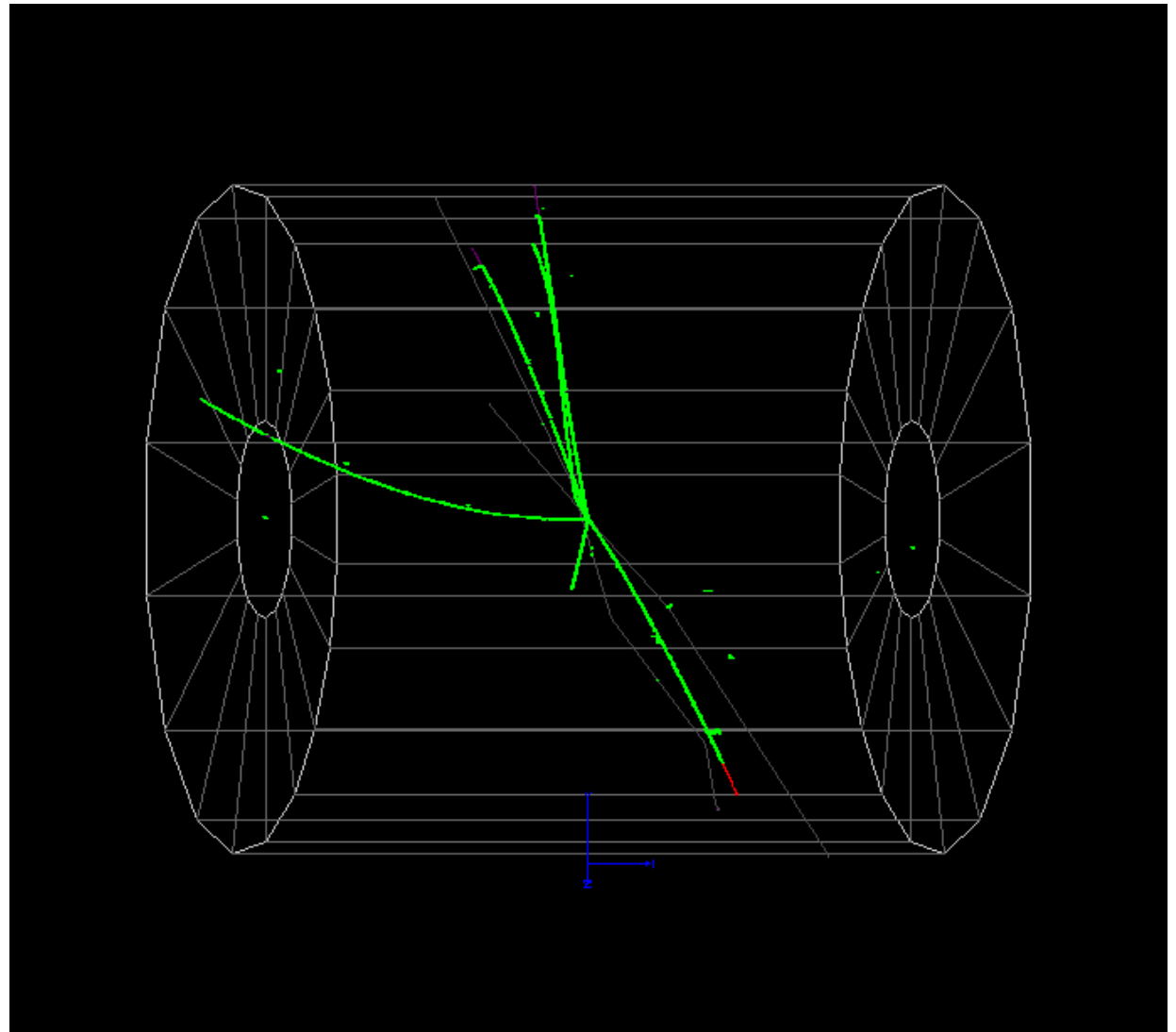
Raw Digits in Green on the 3D event display

Uses the channel geometry, raw::RawDigit simulation, and a drift model.

I had to cheat the event time however.

A $10\ \mu\text{s}$ beam spill window width causes visible displacements in raw digits.

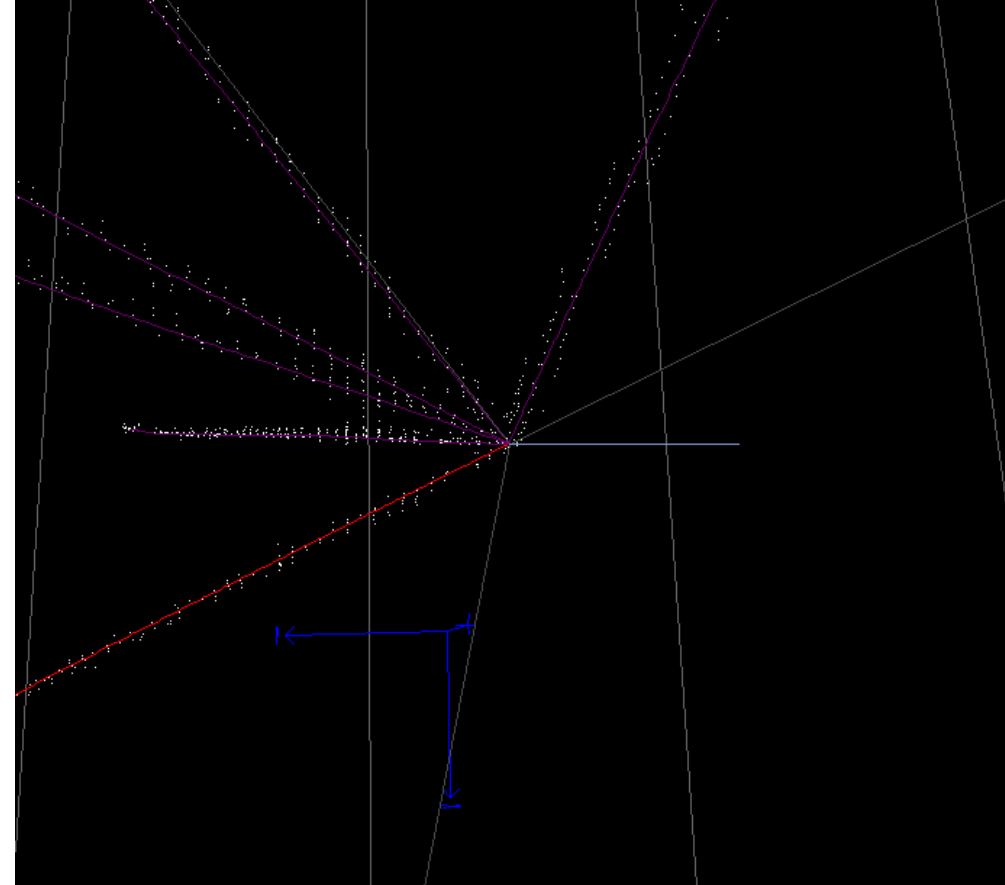
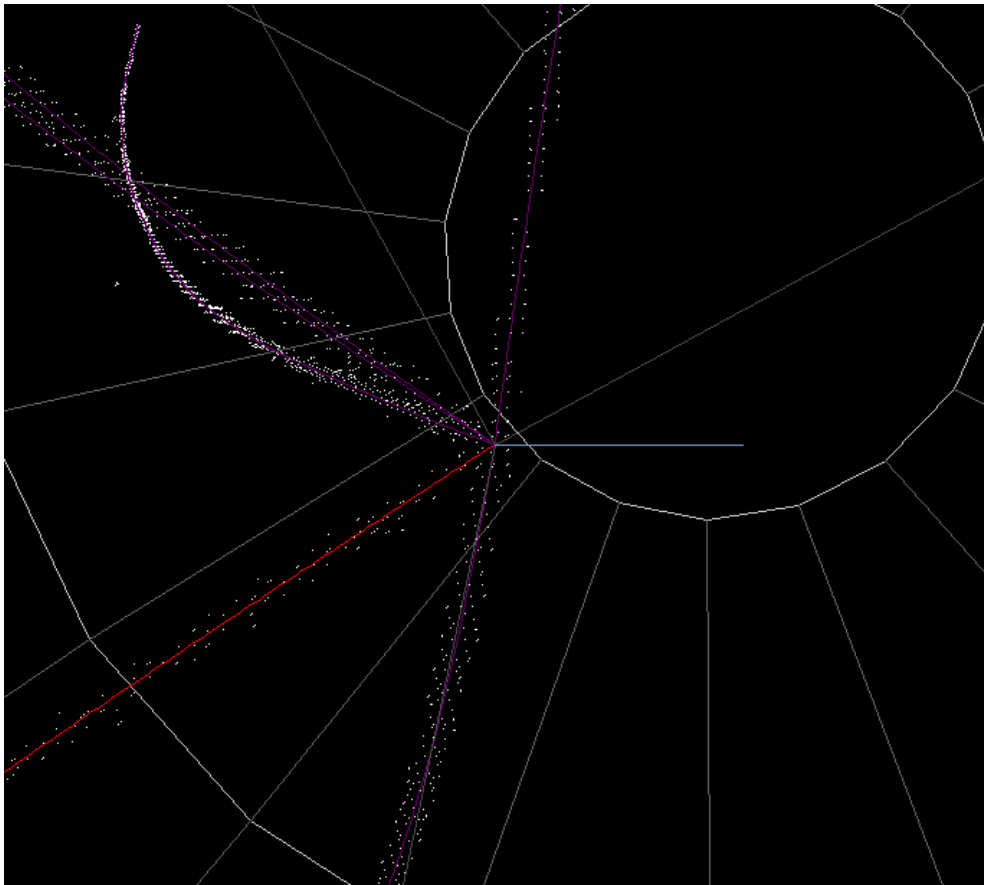
Visible when overlaying MC Truth, but also visible for cathode-crossing tracks



Hit Finder

- Simple initial hit finder -- Just use the Zero-Suppressed blocks of Raw Digits to make hits.
- Threshold applied when making zero-suppressed blocks. 5 ADC counts in simulation. No noise, so no fake hits.
- 5 extra ticks before and 5 after the above-threshold region to get sub-threshold tails.
- Nearby hits are now clustered as ALICE does to improve resolution and simplify pattern recognition and fitting.
- Hit data product used to store hit cluster locations. Charge centroids are now the hit locations.
- How far away a hit can be to be added to a cluster is an adjustable parameter
- Clustering resolution needs to be tested.

Unclustered Hits GENIE veCC Event in the Middle of the Detector



Hit-finder is a simple threshold algorithm
Hits are located at the pad (y,z) locations, and arrival time gives x.
Grid pattern noticeable on tracks. Distribution in x is much narrower

Initial Pattern Recognition

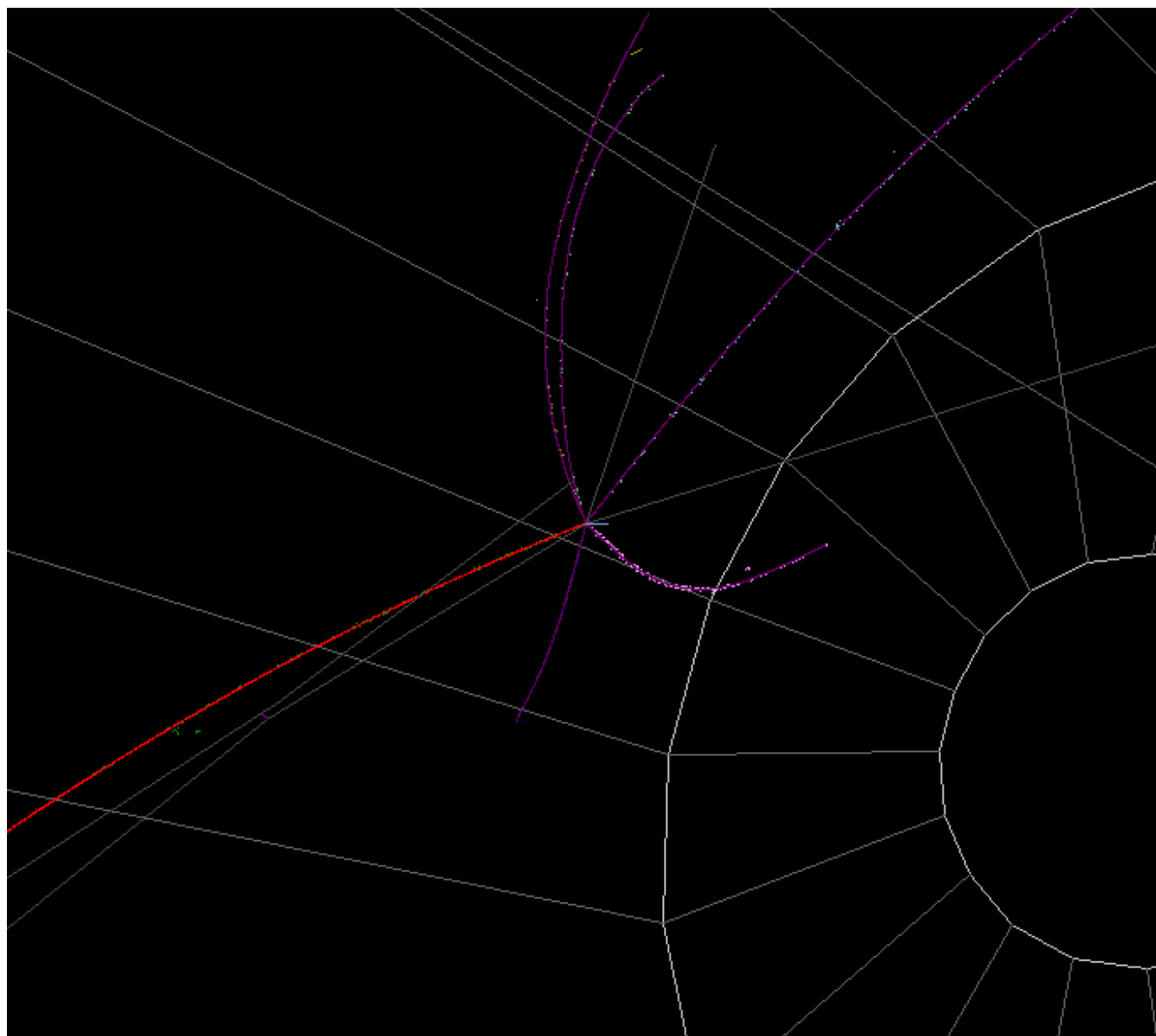
- Hits are sorted in x (E and B point along x)
- Hits are clustered by looping in the $+x$ direction, grouping hits that are nearby in y and z with existing groups. Start a new group if new hit is far away from existing ones at that x .
- Do the same iterating in the $-x$ direction.
- Very forgiving for kinks. Greedily attaches tracks on other side of a vertex.
- Look at the $+x$ and $-x$ clusters and split them into disjoint subsets. Hits in a cluster must be clustered with other hits in the same cluster in both the $+x$ and $-x$ sorts.
- This final step breaks the ambiguous pieces of tracks apart from the long, unambiguous pieces.
- Still follows kinks if they are followed in both $+x$ and $-x$ directions.

Pattern Recognition on First Event

Lines: MC Truth
Red: electron
Purple: Protons

Hits on tracks
shown in
colors (many
are obscured
by the track
lines)

Min # hits=40
Short proton has
dark blue hits
(hard to see on
display)



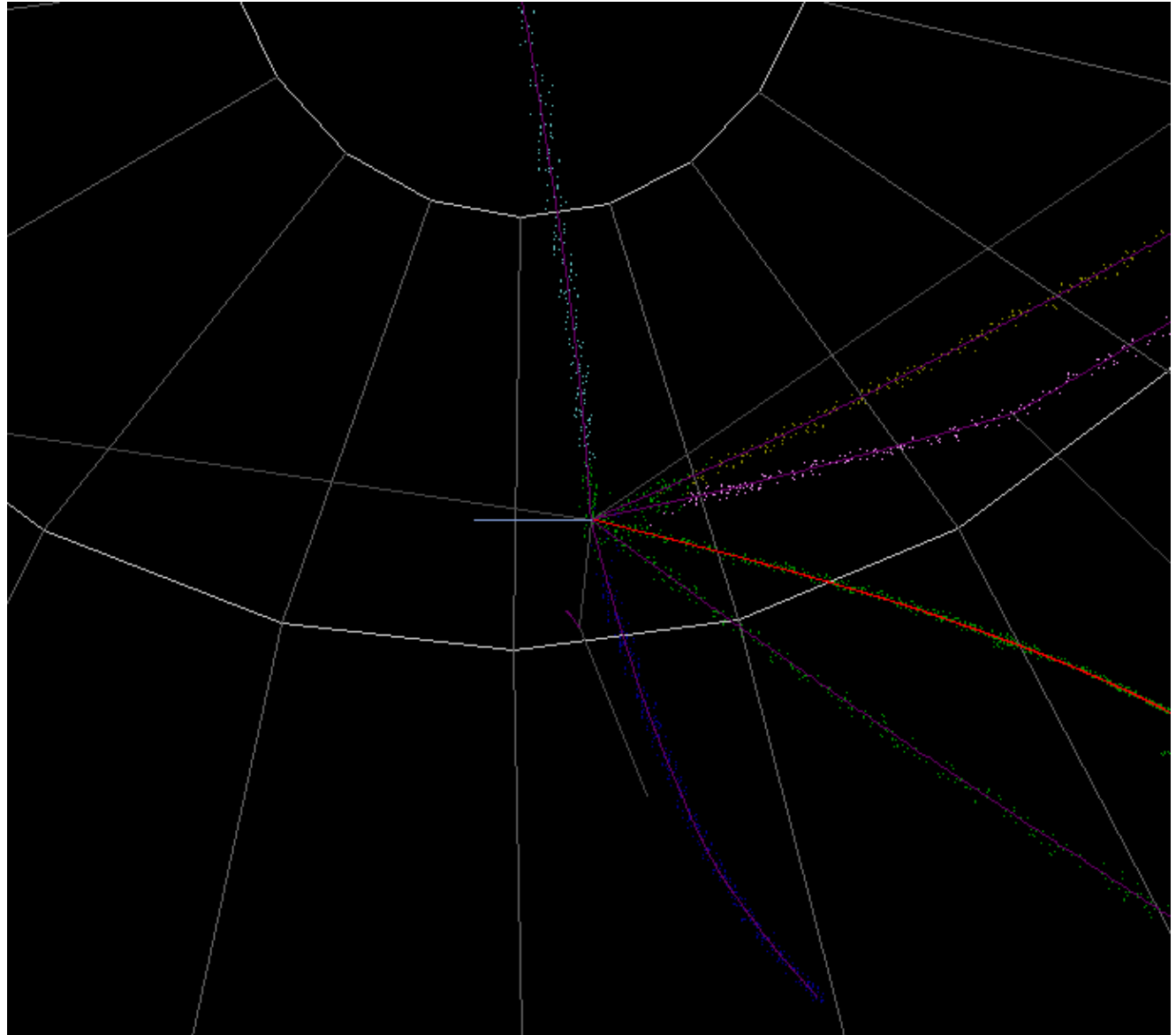
Still Not Perfect Near Vertex

Zoomed further in.

The green cluster overstates its welcome. It extends into the blue, purple, and yellow tracks, and continues along red and purple.

Pattern recognition is a little too forgiving when tracks kink

Also anisotropic – hits are sorted in x and stepped in just one direction



Tracking Strategy

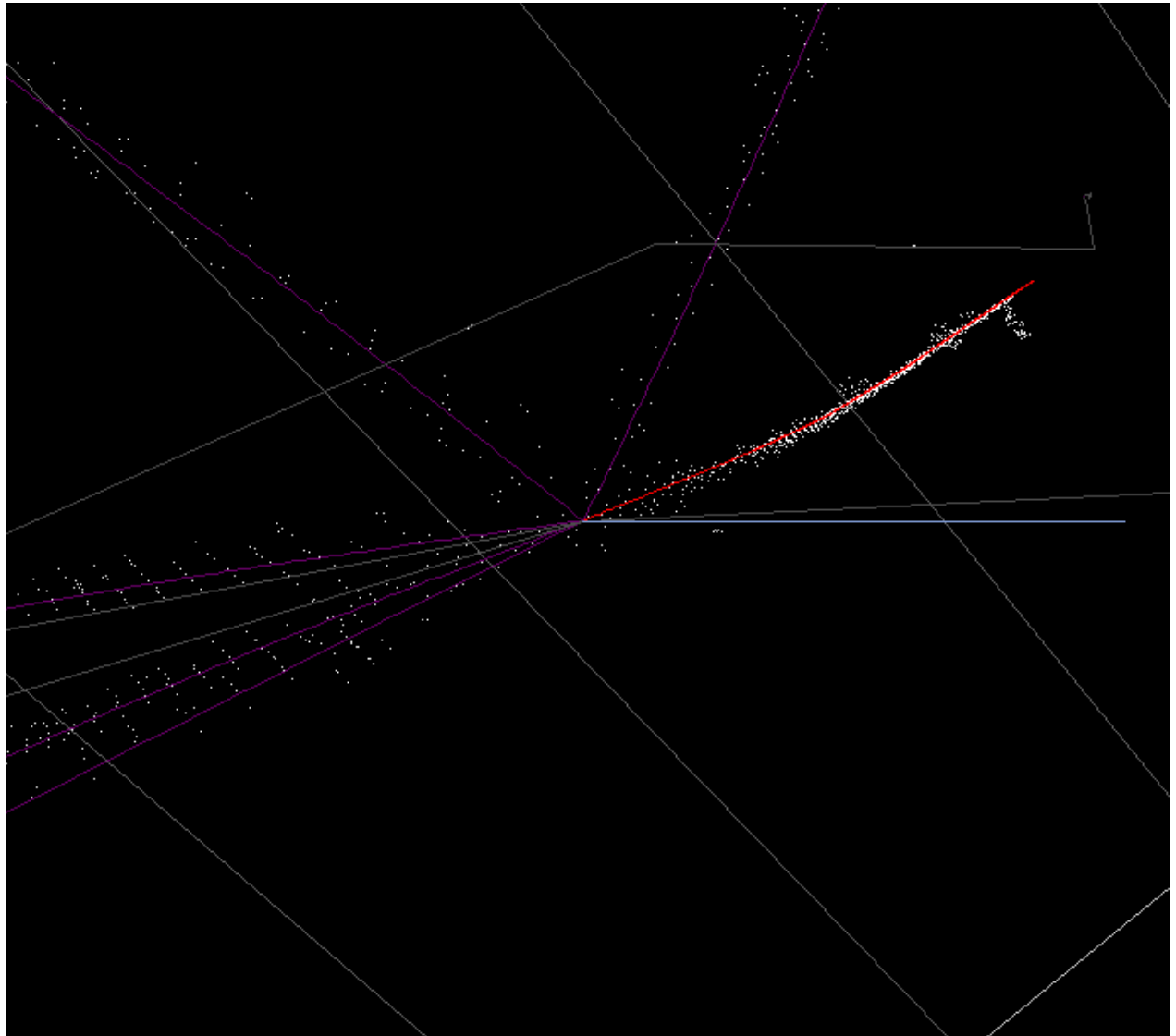
- Run hit finder and clustering
- Run Initial Pattern Recognition
- Fit tracks with a high cut on n_{hits} – find long tracks first.
 - Two directions per track.
 - Initial guess of track parameters to seed fit
 - Kalman or simple Helix fit. Run twice, dropping hits with large contributions to track chisquared.
- Fit vertices with long tracks found and fit above
 - Initial vertex finder and fitter written. Linear extrapolation of tracks from nearest point. Need to iterate with helix parameterization.
- Reassign hits near vertex: identify short tracks
- Fit short tracks near vertex

Kalman Filter Track Fitter

- Initial code written – in `garsoft/reco/tracker1_module.cc`
- Fits each track in both directions
- Kalman Filter algorithm starts with track parameters at one end of track, at a specific x , (y , z , curvature, ϕ , slope), and a guess at the covariance matrix
- Filter steps along the track and updates the track parameters and covariance matrix at each point.
- Adjustable parameters:
 - Hit uncertainties
 - Initial track parameters and uncertainties
 - Tolerance for dropping hits

Delta Ray

- Unclustered hits for this display
- Kalman fit can slavishly follow the track or it can keep its ideas of the track parameters more rigid

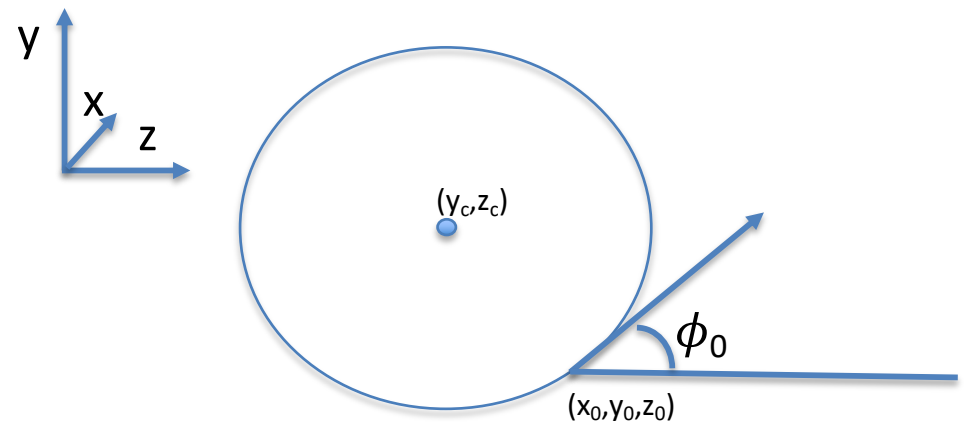


Working on Helix Fit

- "simpler" than the Kalman fit, but possibly less stable.
- Uses MINUIT (for now) to minimize sum-squared distances from each hit to the helix.
- Need to start the track parameters off with good guess.
- Track parameters I use (units are cm)
 - (y_0, z_0, c, ϕ_0, s) , quoted at x_0
 - x points along the drift field and the B field
 - y is the vertical axis
 - z is x cross y
 - c is the curvature in cm^{-1} (n.b. can be positive or negative). Natural parameter rather than radius as high-momentum tracks have small c and big r . Hard for a fitter to flip the sign of a big number rather than a number near zero.
 - ϕ_0 parameterizes the direction at (x_0, y_0, z_0)
 - s is the slope (n.b. can be positive or negative). $d\rho/dx$, where $d\rho = \sqrt{(dy)^2 + (dz)^2}$

Helix Track Parameterization

- Derived parameters:
 - Radius: $r=1/c$ (in cm, positive or negative).
 - Circle Center (in cm) (y_c, z_c)
 - $y_c = y_0 + r \cos\phi_0$
 - $z_c = z_0 - r \sin\phi_0$
- Locus of points on helix. $\phi=0$ at (x_0, y_0, z_0)
 - $x = x_0 + (r/s)\phi$
 - $y = y_c - r \cos(\phi+\phi_0)$
 - $z = z_c + r \sin(\phi+\phi_0)$



Encoded in TrackPar.cxx, .h

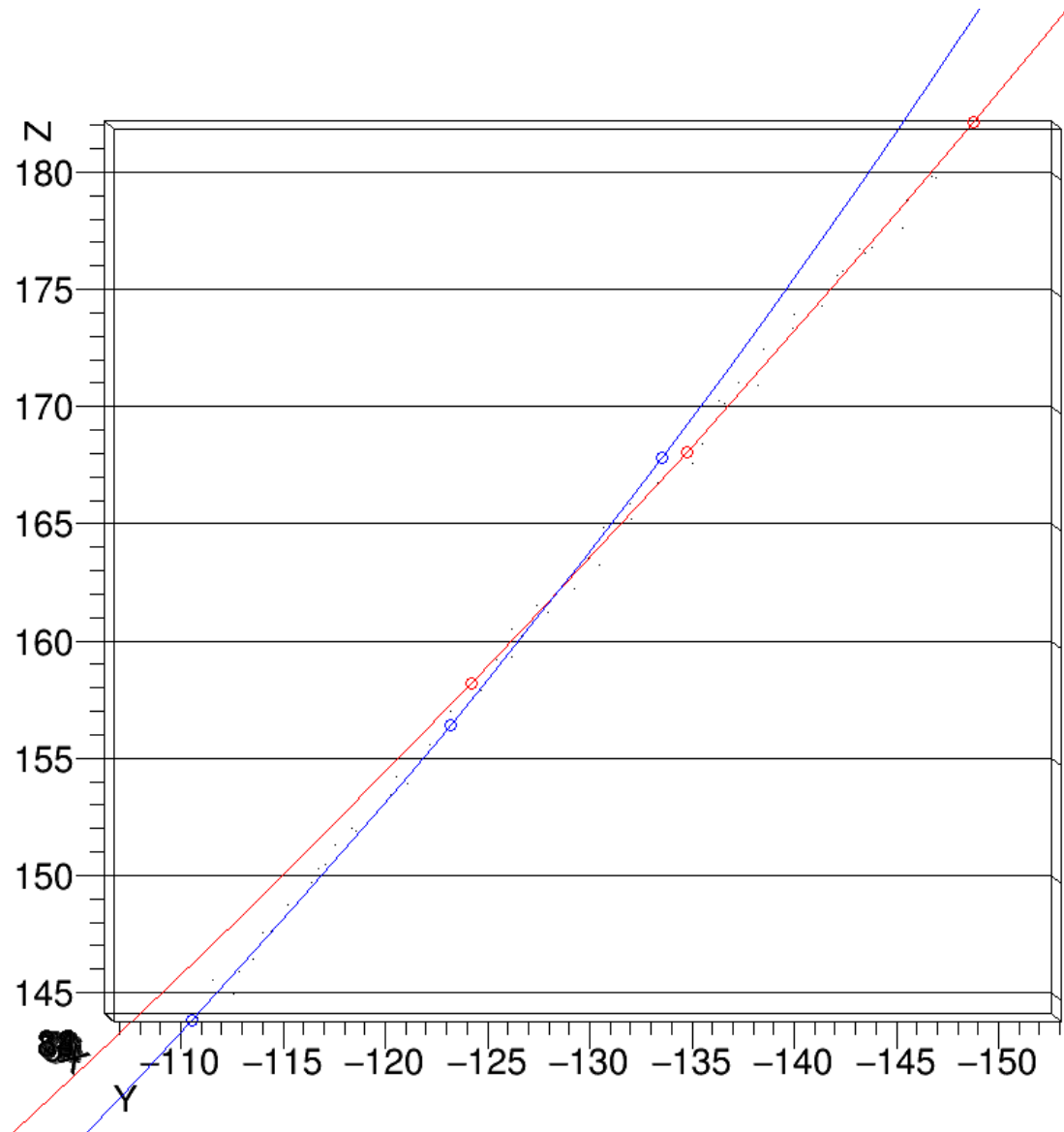
Performance of Initial Track Parameter Guess

(Y,Z) view.
Projection is
a circle.

Min # hits = 40,
so look at
hits 0, 19, and 39

Do it twice, starting at
either end of the track.

n.b. hit 0 may be more likely
to be a mistake than others.

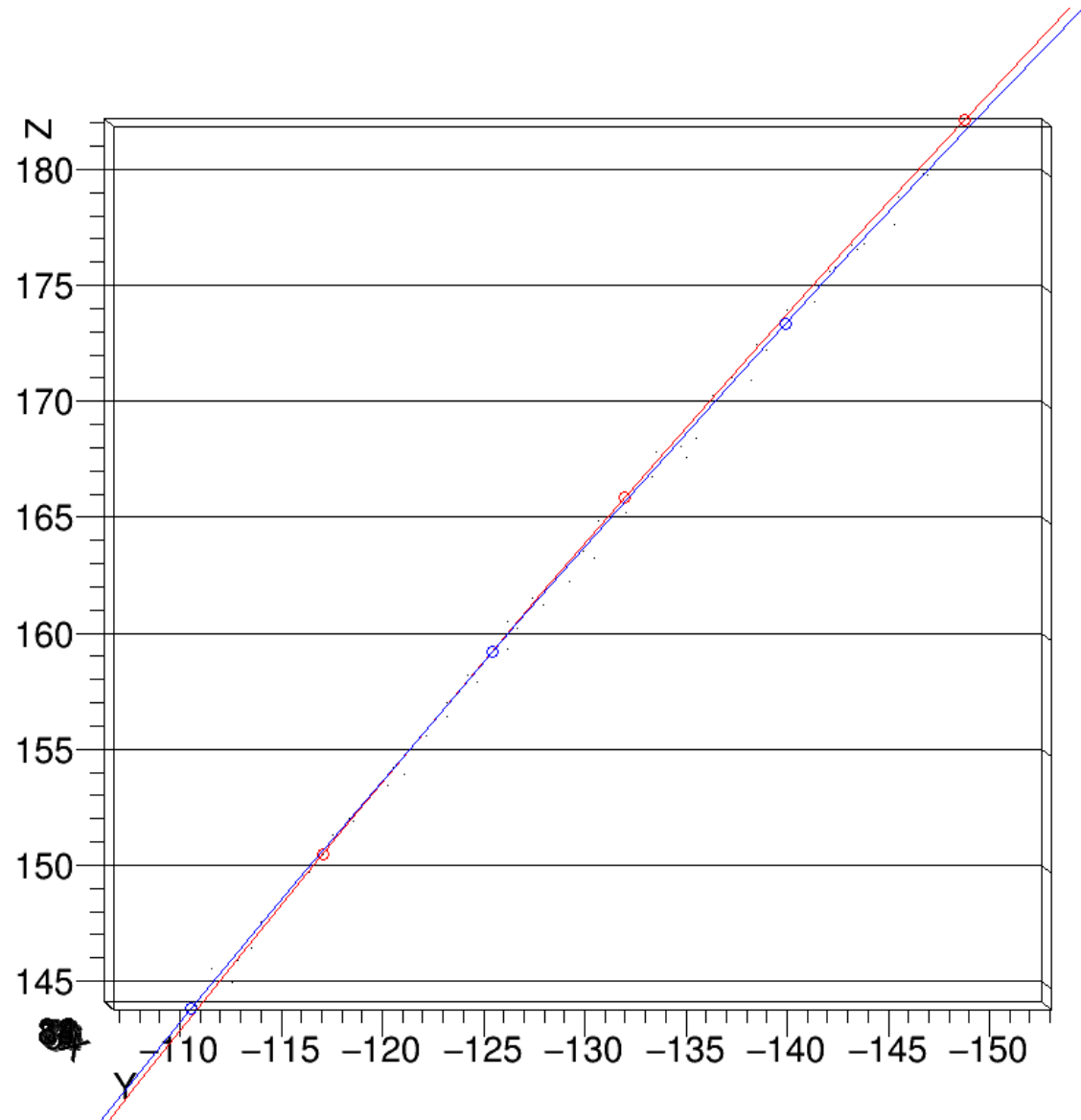


Same track, different three hits

Hits 0, 24, and 49

(Y,Z) view again.

Looks good!

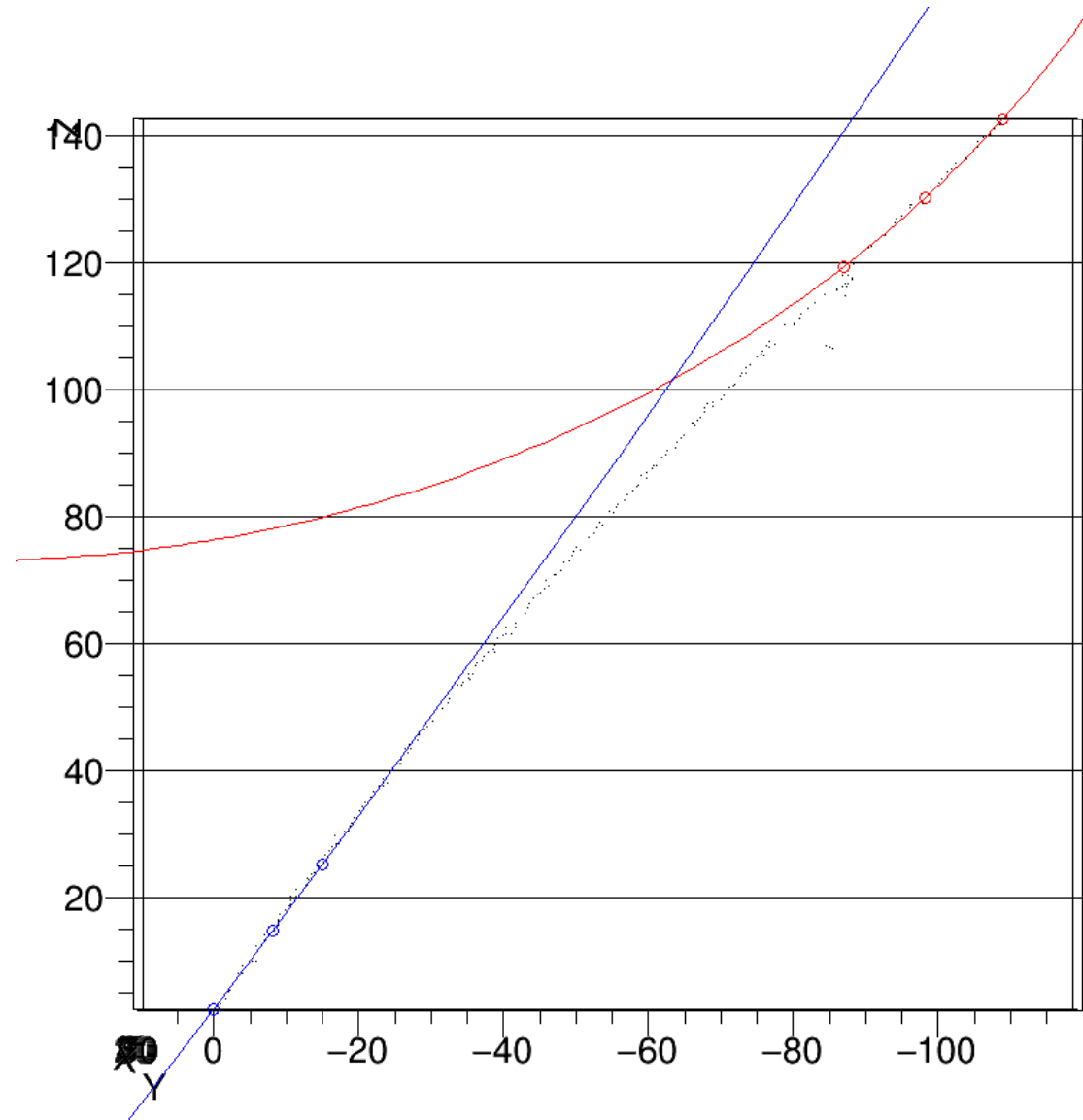


Here's one that got away:

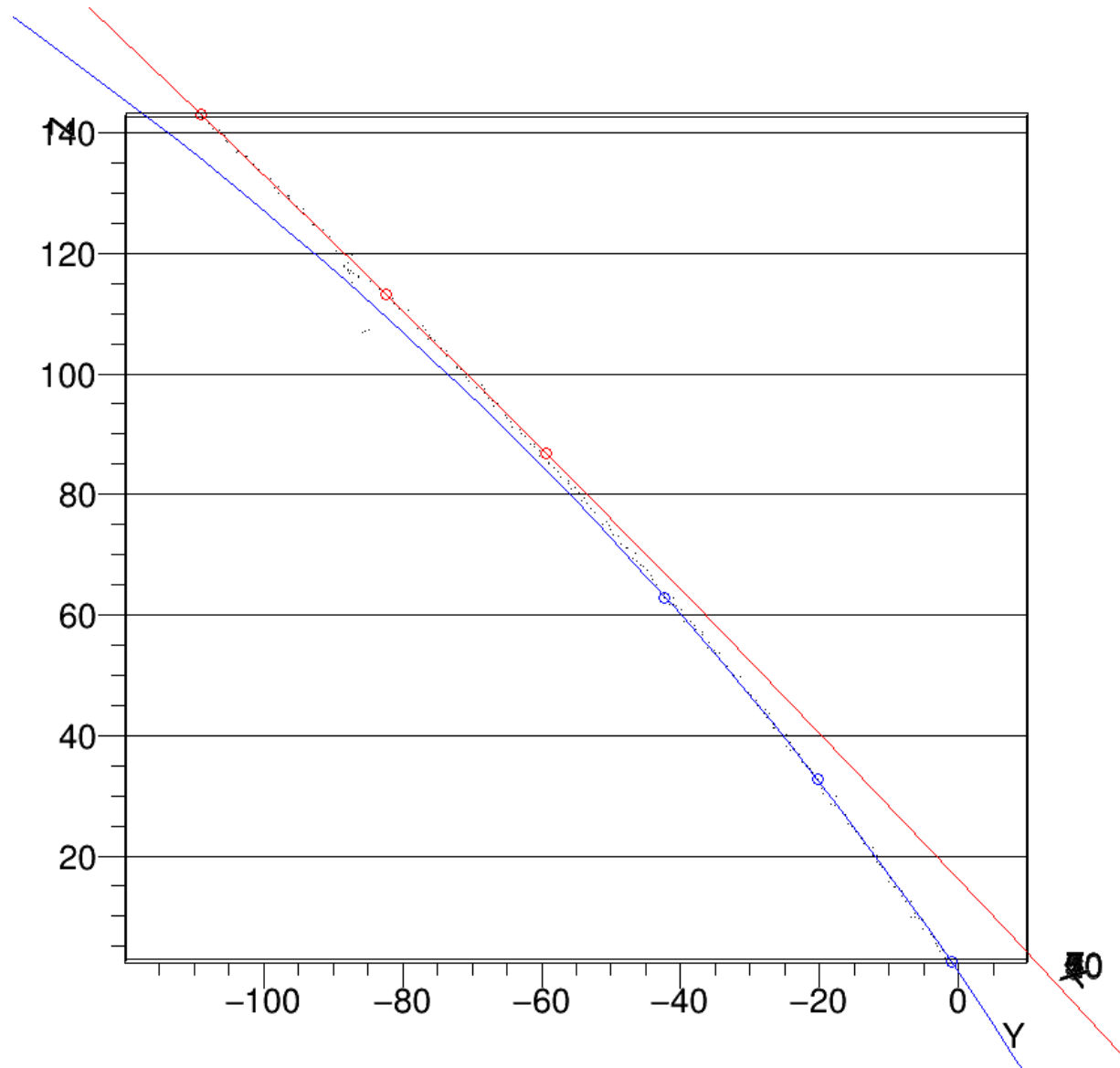
A longer track with a little scattering and a few extra hits.

Hits 0, 13, 37

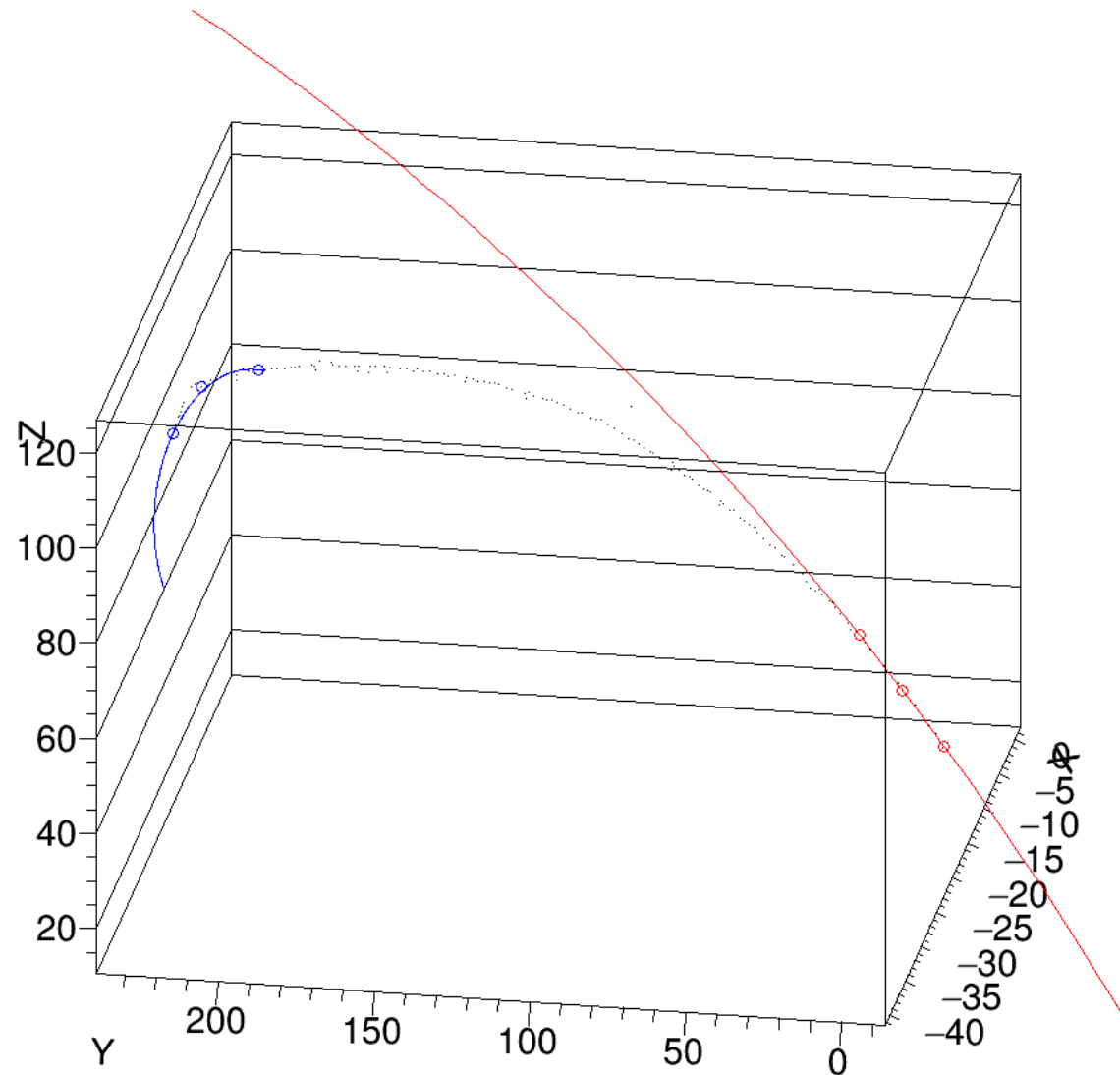
(Y,Z) view



Much Better with 100 hits



A track that scatters on the end

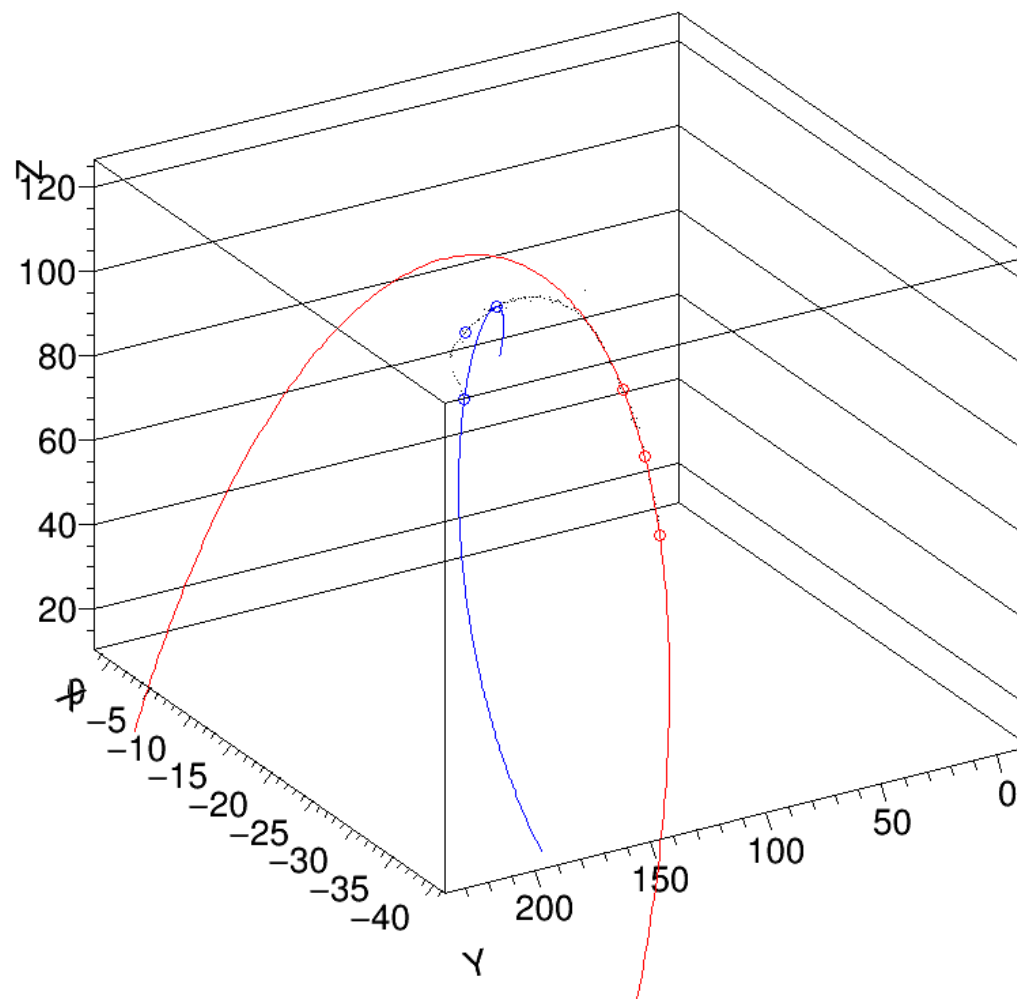


Same track, different view

Track is not
very helical
on the end

So the helical parameterization
only goes through the first
and third points.

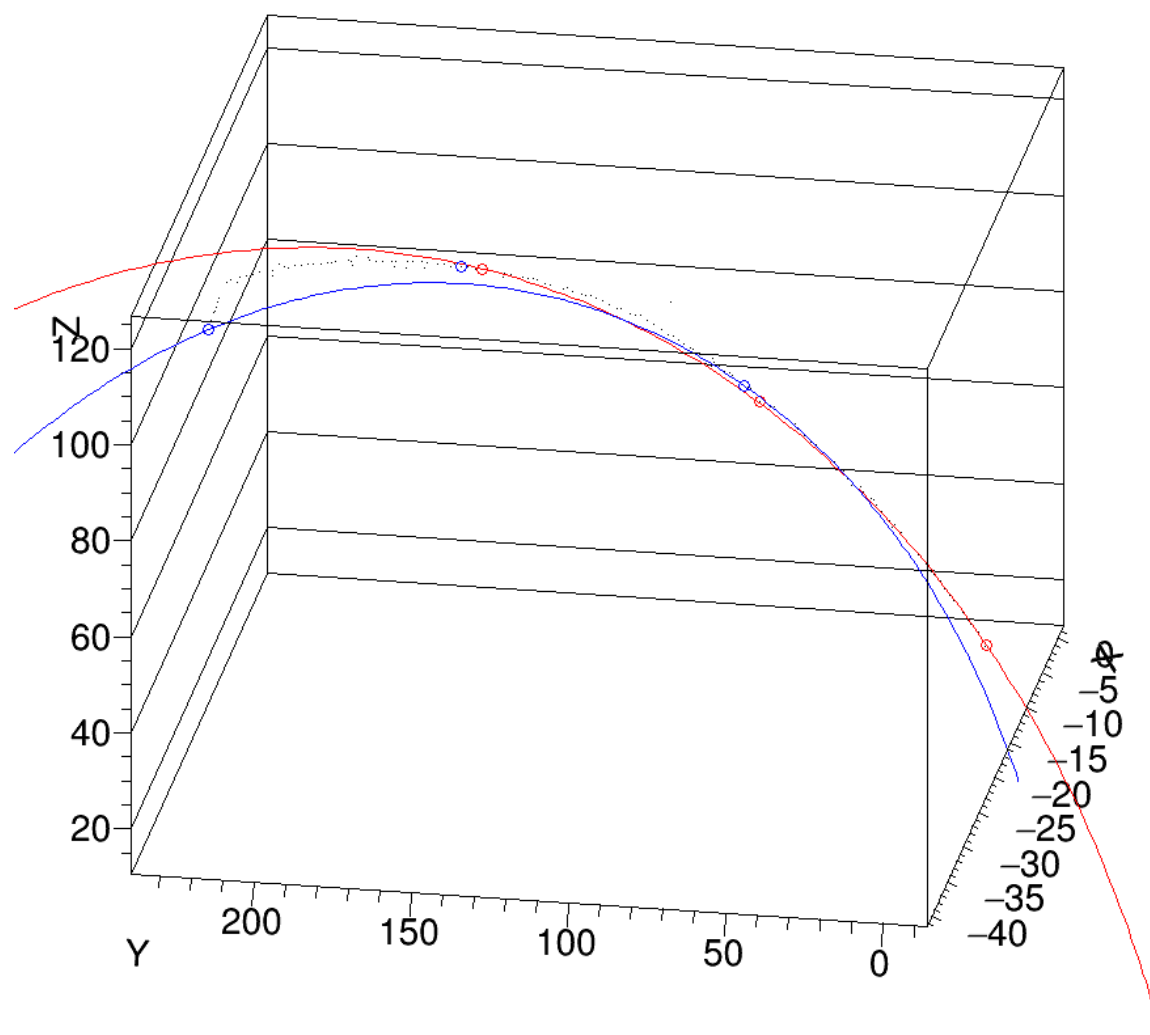
In the YZ view, the circles
go through all three points.



Same Track, Longer Baseline

Close! But we don't expect perfection if we include bits of the track that aren't very helical.

Primary vertex for this event is at (0,0,0), so it's not a case of following a track across the primary onto another track



















Single Helix Fit

- Since a perfect helix doesn't have any energy loss, the track parameters (curvature, slope, center) are the same at the beginning and the end.
- No need to make a distinction here for either end. Might as well use the entire track (or chop off the ends just to be safe). Track parameter estimates look better with longer baselines.
- To do: select the three points for initial track parameters and test the helix fit.
- Differences with ALICE:
 - We're interested in tracking near a cluttered vertex
 - Scattering and energy loss are more important at 10 bar
 - We are interested in much lower-momentum tracks

Building with Jenkins

Build script for garsoft currently in larutils, along with the dunetpc build script and all the larsoft build scripts. And MicroBooNE's, LArIAT's...

The build matrix:

Configuration Matrix		OSX-10.12	OSX-10.13	SLF6	SLF7
debug	e15				
	c2				
prof	e15				
	c2				

The c2 builds are much "fluffier" than the e15 builds. Larger libraries.
Still good to compile with two compilers. Catch bugs that way.

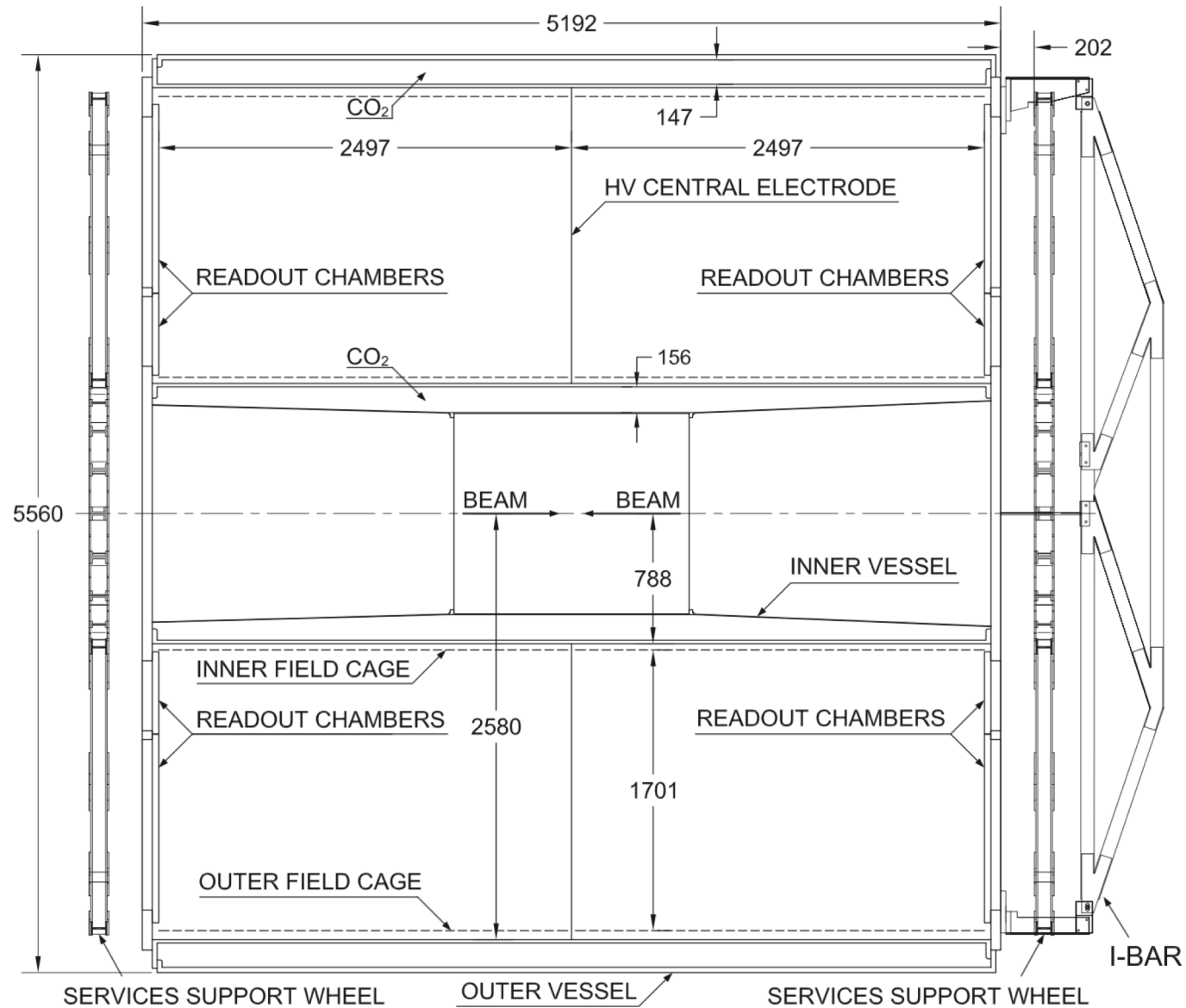
- To set up the pre-built release
`source /cvmfs/dune.opensciencegrid.org/products/dune/setup_dune.sh`
`setup garsoft v01_01_01 -q e15:prof`
- To find what versions are available: `ups list -aK+ garsoft`
- Setting up garsoft *and* larsoft (e.g. dunetpc) in a session is not recommended! Likely to involve version clashes of dependent products like art. The setup command will likely complain.

Getting Ready to be Part of a Bigger ND Complex

- Geometry from Mike Kordosky places the origin somewhere in ArgonCube
- GArSoft shouldn't bake in an origin in the GArTPC
- Origin now determined from the geometry read in from the GDML file
- Channel locations, event display, etc. now shifted.
- Tested with displaced origin.
- Axis definitions are starting to ossify. X is along drift, Y is up, Z is $X \times Y$.

Extras

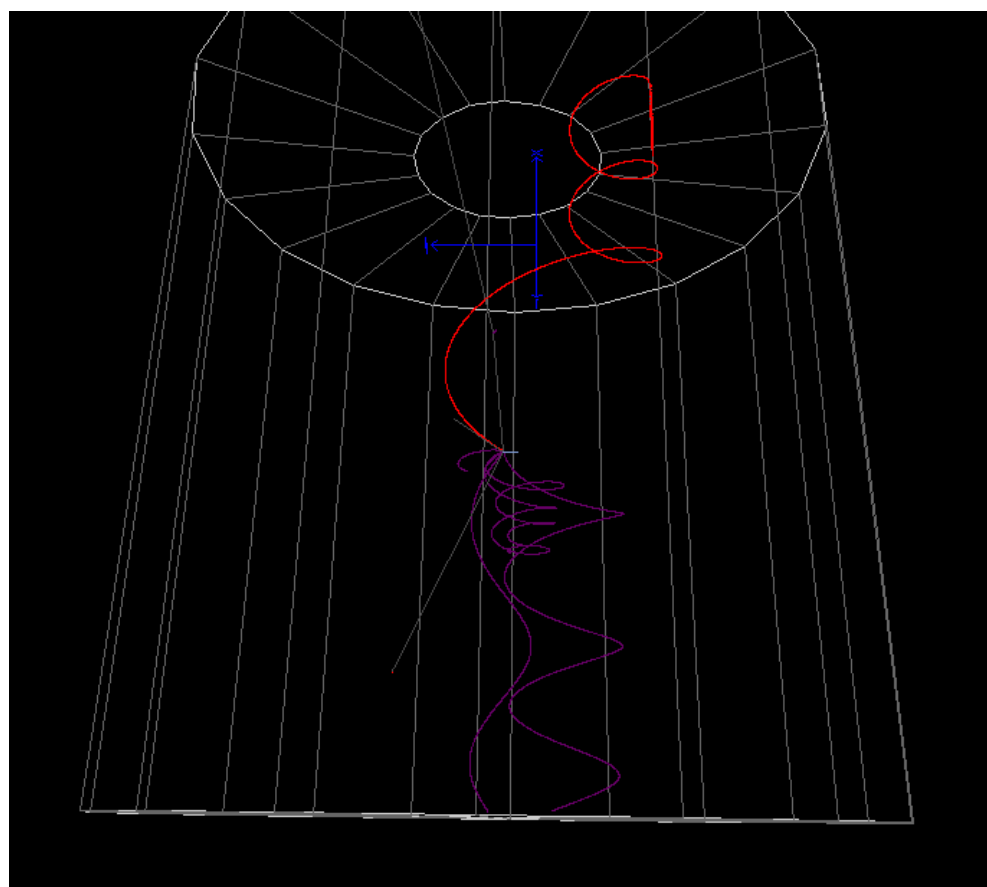
Side View of the ALICE TPC



Another ν_e CC Event

Gray neutrons
more visible here

One of the neutrons scatters
in the GAr.



Adding Raw Digits in Green

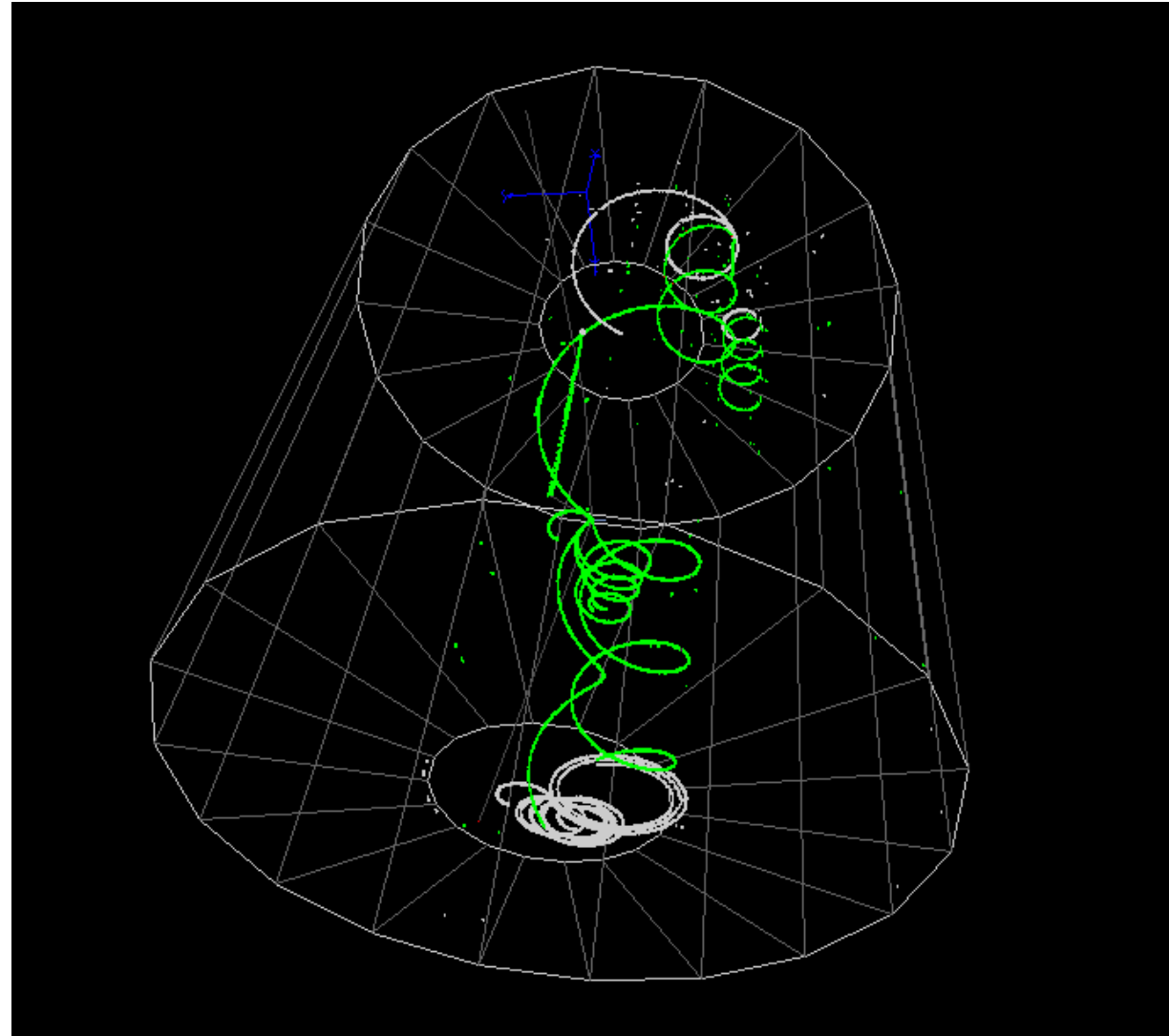
And also Gray for
the 2D projection
on the readout planes

Uses the channel geometry,
raw::RawDigit simulation,
and a drift model.

I had to cheat the event
time however.

A $10\ \mu\text{s}$ beam spill window
width causes visible
displacements
in raw digits.

Visible when overlaying MC Truth,
but also visible for cathode-crossing tracks



Software Improvements

- GArSoft came with a `raw::RawDigit` data product similar to the LArSoft one
- I added zero suppression, a feature used by ALICE. I (mostly) copied the LArSoft ZS algorithm (J. Insler)
- Adjusted the electrons to ADC conversion factor (what is the gas gain?) and the drift velocity. Still unknown. $0.2 \text{ cm}/\mu\text{s}$ assumed for now.
- For now, the noise is turned off, so the gas gain just needs to be big enough for hits to pass the zero-suppression threshold.
- Added a magnetic field of 0.4 Tesla (interface already in nutools, but uniform magnetic field only)
- Upgraded to art v2_22_02 and nutools v2_23_02, so we can shoot monoenergetic neutrinos at the detector.
- Tested GENIE interface
- Stepsize limit of 2mm as a fcl parameter (already in GArSoft)
- Resurrected the 3D event display and added an ALICE wire frame
- Fix bugs

Estimate Initial Track Parameters

- Three points define a circle (2D).
 - Stole ALICE routine to compute center and curvature
 - Coded one up from scratch to check 1: sign of curvature, and make sure center is on the correct side.
 - Need ϕ_0 and s once we have c .
 - $\phi_0 = \text{ATan2}(z_0 - z_c, y_c - y_0)$. If $c < 0$, add π to ϕ_0 . This is gotten from the angle from the center of the circle to (x_0, y_0, z_0)
- Slope gotten from a point on the track (x_2, y_2, z_2) .
Use $x_2 - x_0 = (r/s)\phi_2$
so
$$s = (r/(x_2 - x_0)) * [\text{ATan2}(z_2 - z_c, y_c - y_2) - \phi_0]$$

Term in brackets is $\Delta\phi$. If it's bigger than π , subtract 2π . Less than $-\pi$, add 2π , assuming we're following the track for less than one turn. Low- p_T curlers can alias and look like higher- p tracks.
- Three points *overdefine* a line! So the slope is different if we use any two of the three points used to define the circle. We know the helix axis is along x .