SVT Hit-on-Track Finding Efficiency*** for 2021 data

Matt Graham
Recon/Analysis Workshop
May 2, 2022

Datasets, detector and code

- Data: run 14166 from evio
  - This is a low luminosity run from 2021
- Hps-java: iss895 … merged HEAD ***into this*** ~mid March
- Detector: HPS_Run2021Pass2FEE
- Steering: org.hps.steering.analysis.PhysicsRun2019SVTHitEffKalman.lcsim
  - I use Kalman tracking for this
- Analysis Driver: org.hps.recon.tracking.kalman.SvtHitEfficiencyKalman.java
  - This is just the 2019 reconstruction code, but just doing Kalman tracking and including this analysis driver

Additions/hacks to Kalman Code

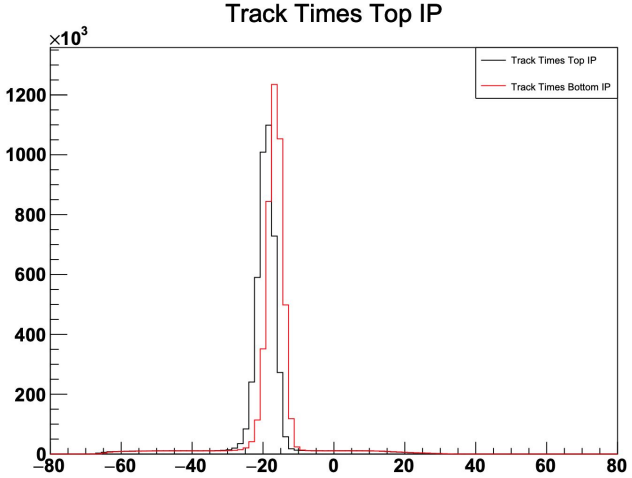
“Did we put a hit on the track if we think a track went through the active part of the sensor”

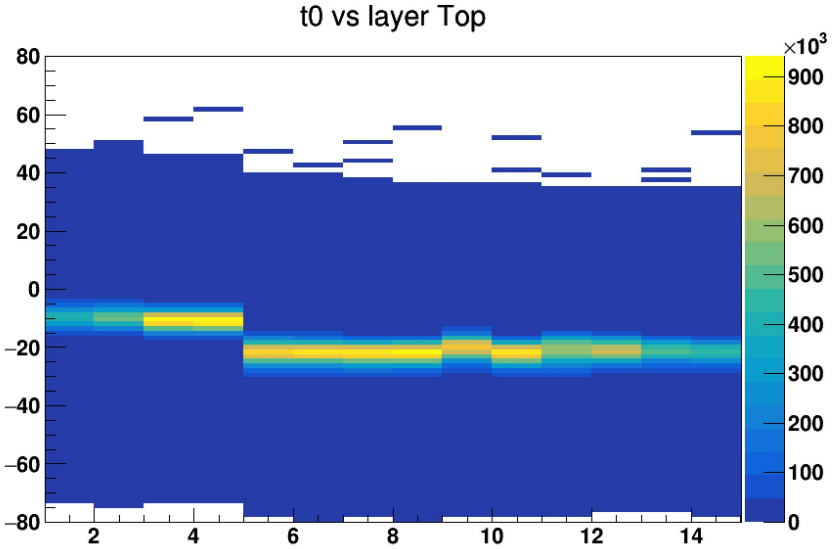
- I asked Robert to make me code to return the unbiased sensor intersection
  - KalTrack.java already has code for unbiased residuals, this is similar but it returns intersection even if there is not a hit-on-track for a layer
- A couple other changes to KalTrack & KalmanPatRecDriver
  - Made a generic class (TrackIntersectData) and saved intersections for all layers (14) for every track; also relations between these and the tracks
  - KalTrack trims the layers (MeasurementSites) not included before & after track; I hacked this so they didn’t get dropped…have to come up with a permanent solution
- When the MeasurementSite does not have a “smoothed” trajectory, the site gets dropped…does this ever happen for track trajectories through the sensor?

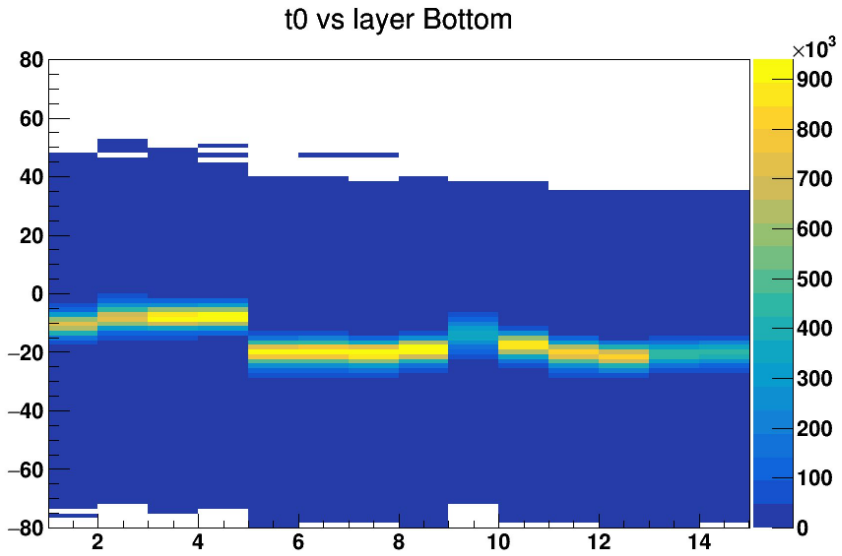
Hit & Track times

Track time distribution is very sharp (as are hit time distributions). Makes sense, this is a low lumi run.

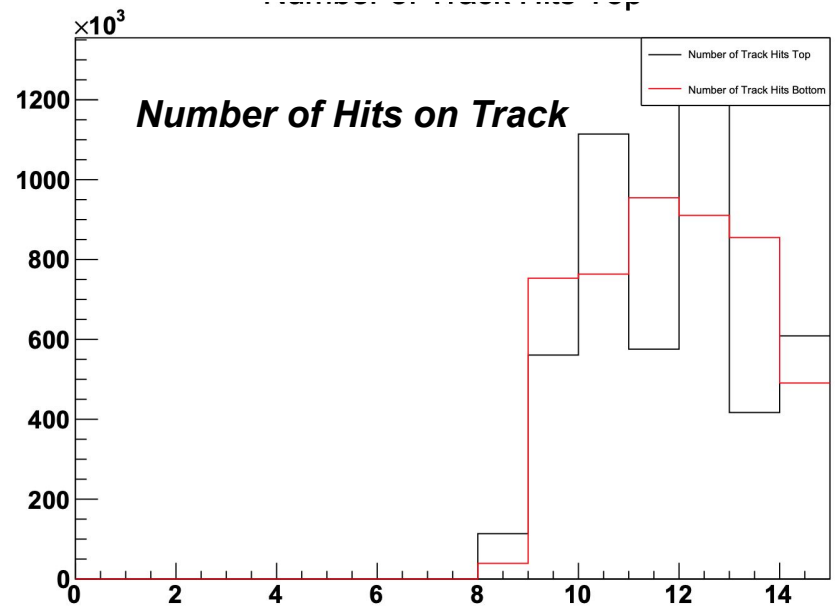
Known issues to fix up: thin sensor hit timing has big offset wrt thick sensors; top-bottom timing offset; per-sensor timing calibration

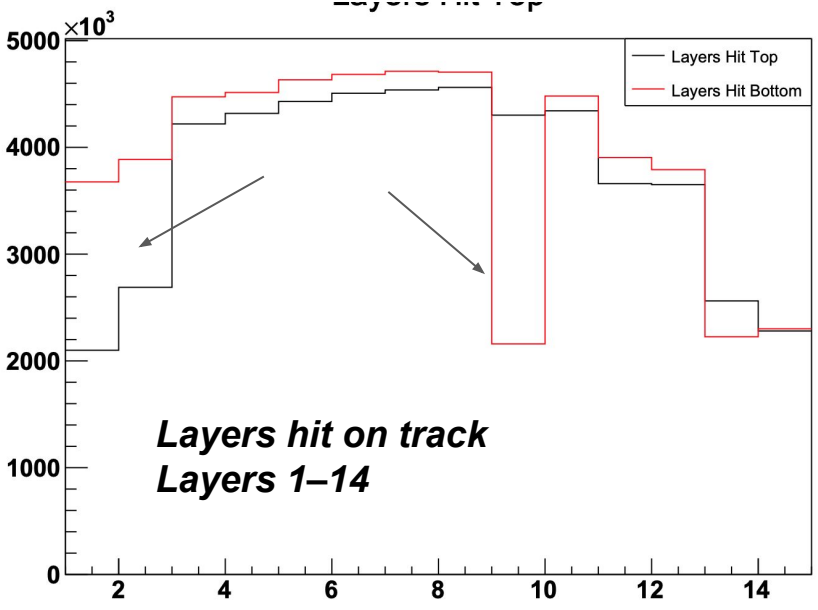






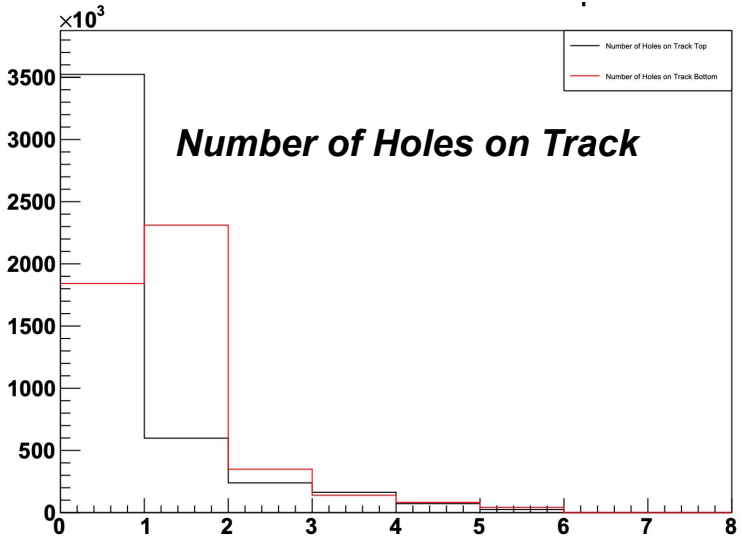
Track-layer composition

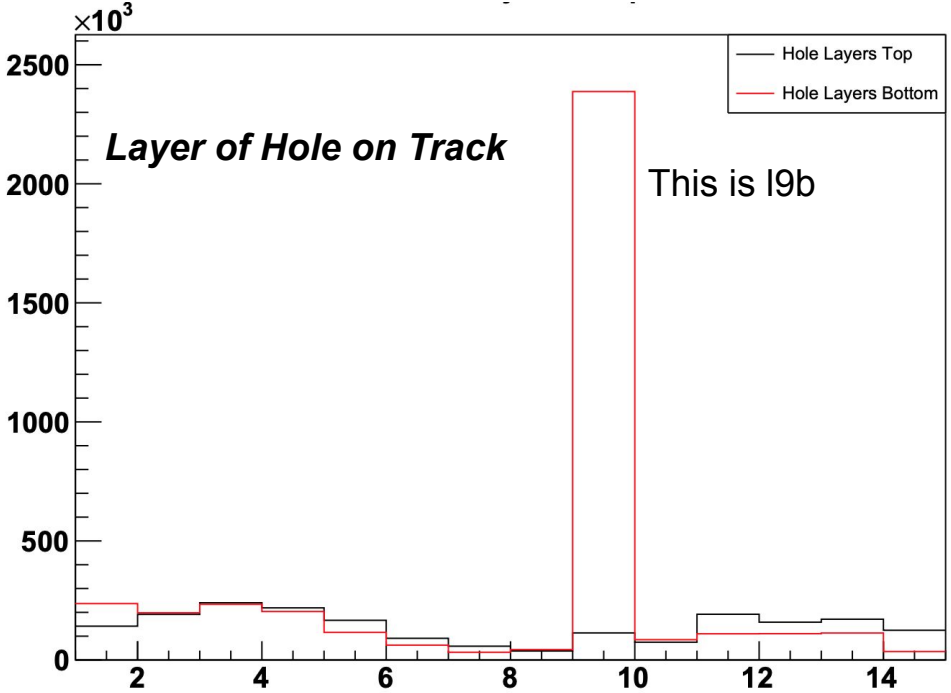




Number of Top & Bottom tracks are ~ same…no scaling on these plots
Note: I put a nHits>8 cut after reconstruction; there were a (very) few tracks with 6, 7 hits

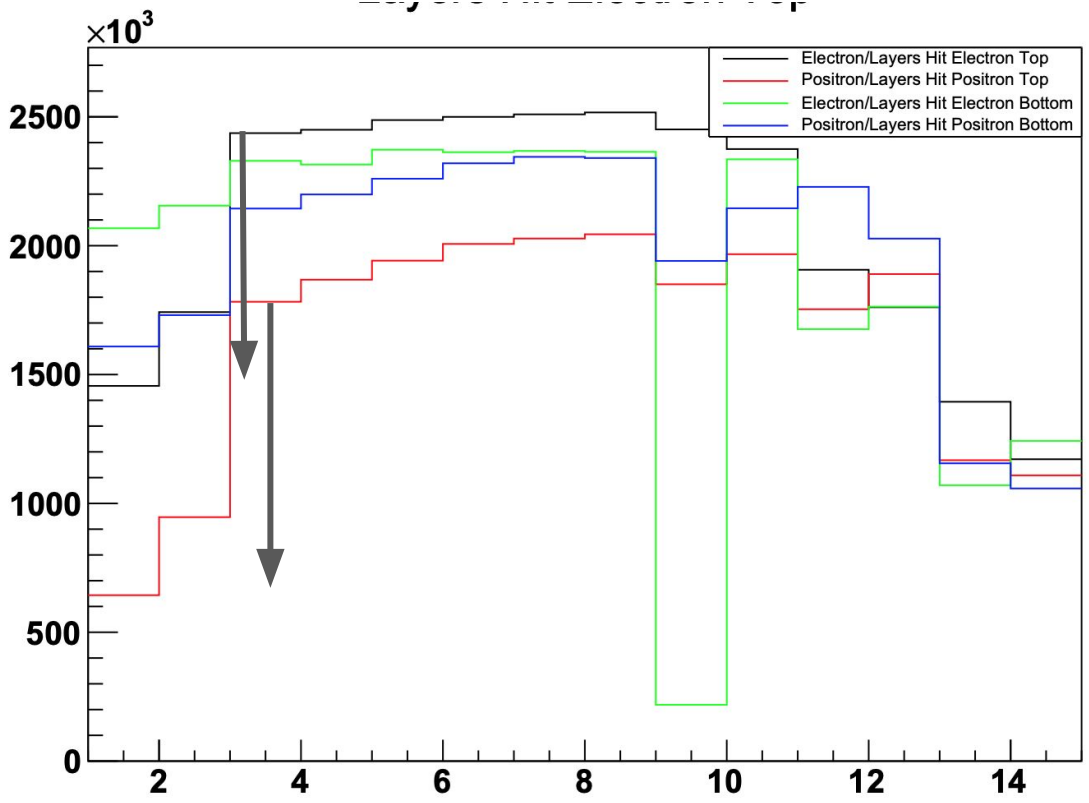
Track-holes!





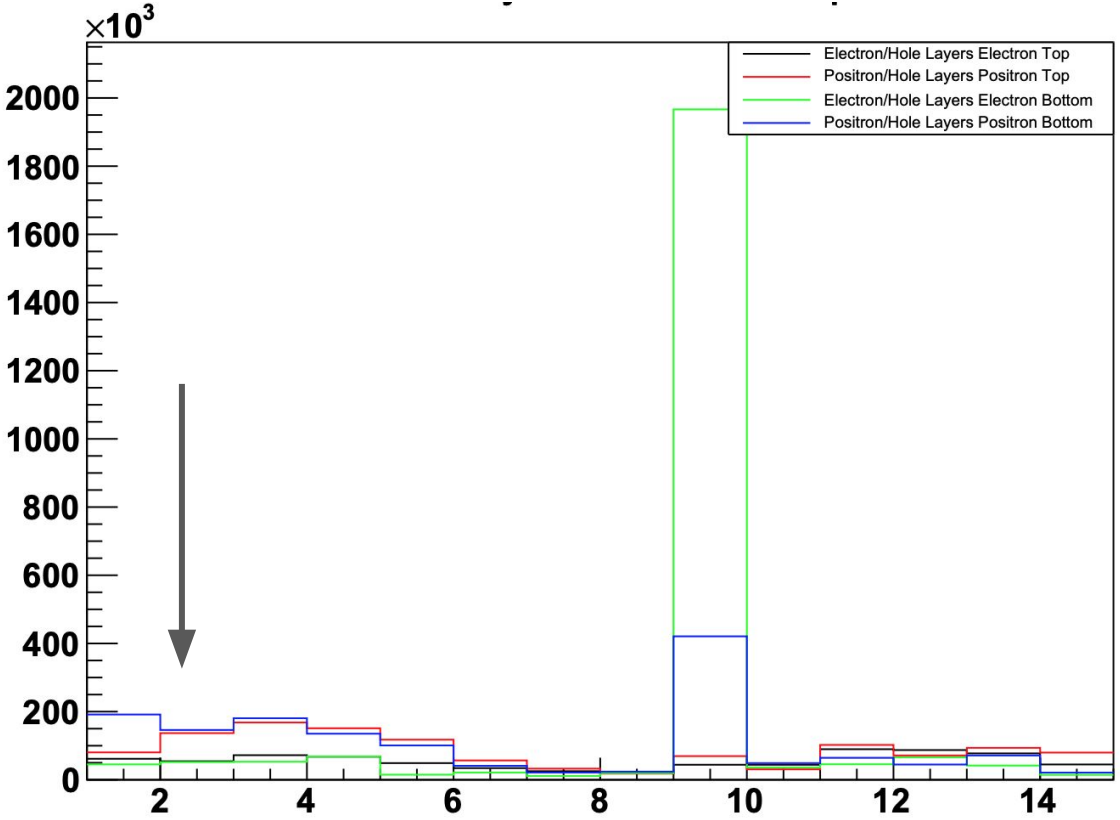
A hole is basically a missing hit…tracks should have gone through sensor but didn't get included on track

Charge/Half separated layers hit



Fewer I1, I2 hits for all charge/half combo but it definitely bigger in top; seems ~same size for positrons & electrons…

Charge/Half separated holes



If these were tracks hitting the l1 or l2 sensors but not getting assigned hits there, they would show up at holes…but I don't see them

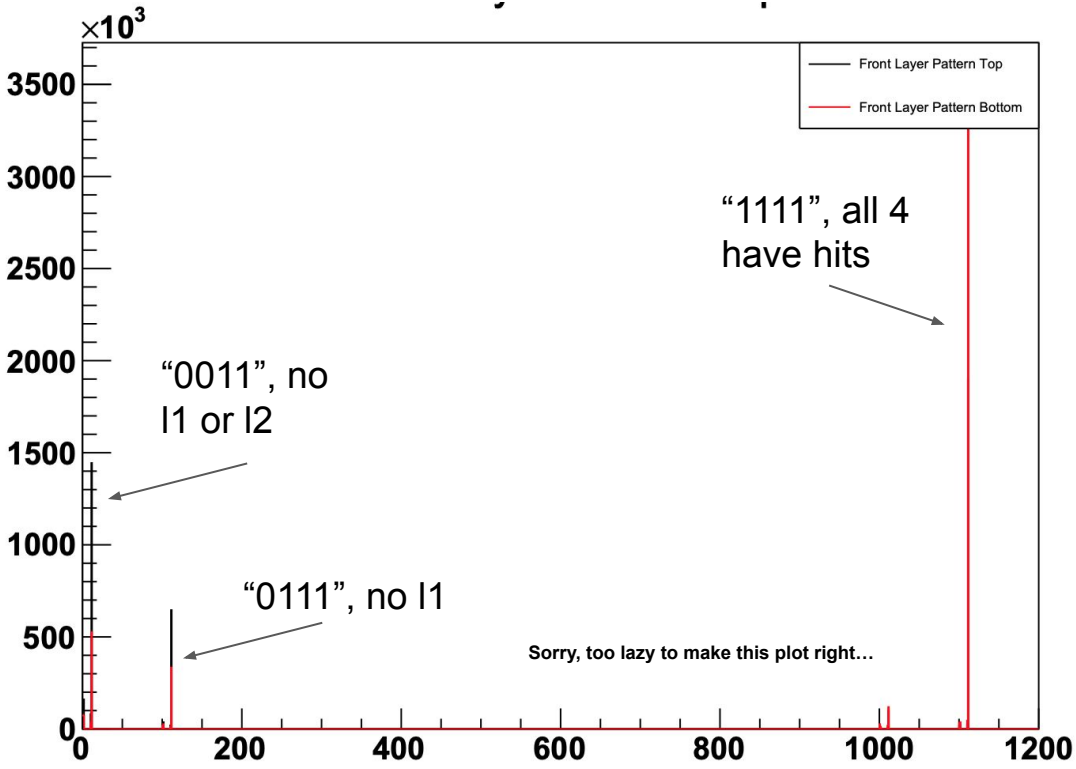
These tracks must have just missed the l1,l2 sensor in top

Patterns of layers 1-4 (thin) hits on tracks

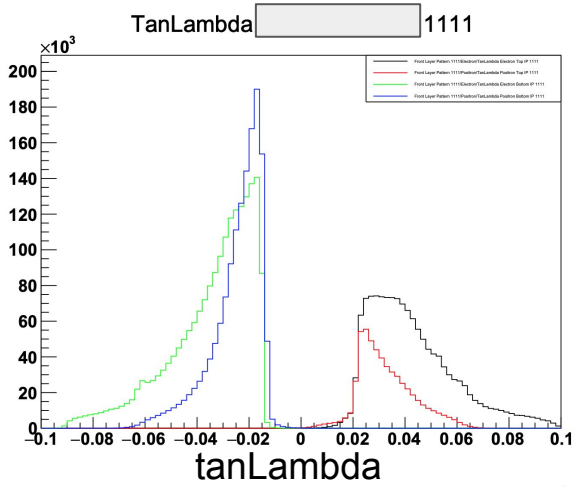
With an eye toward vertexing analysis, took a look at the hit content in the first 4 layers…"coded" it in 4 bits with layer 1 MSB, layer 4 LSB.

Bottom tracks seem to do better at getting I1 & I2 hits (as also seen on slide 5).

Next slides, I'll look at the three dominant patterns….

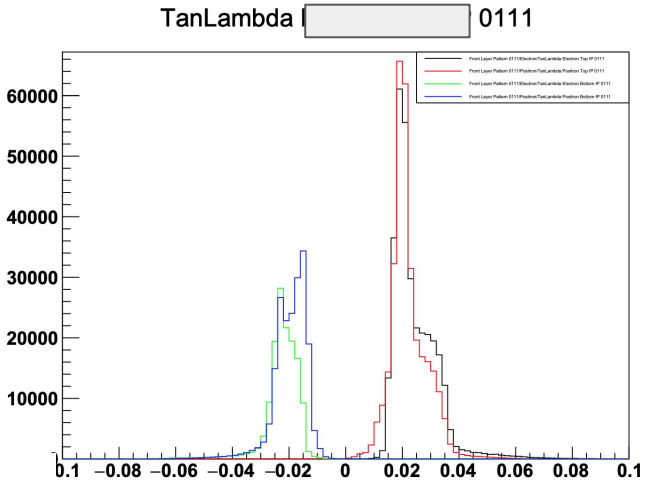


“1111”, “0111”, “0011” : tanLambda (at 0,0,0)

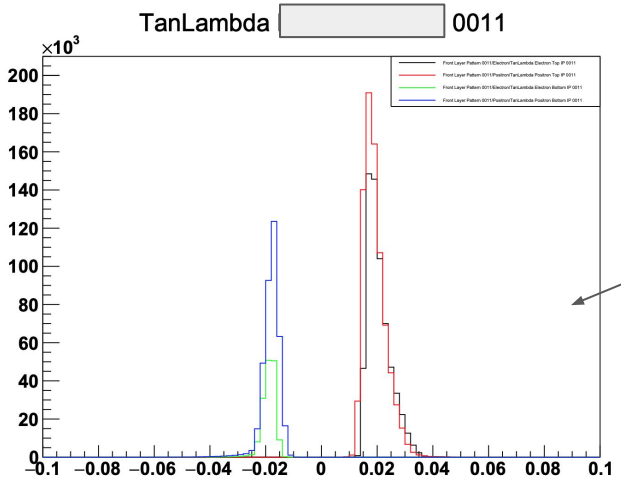


WEIRD!

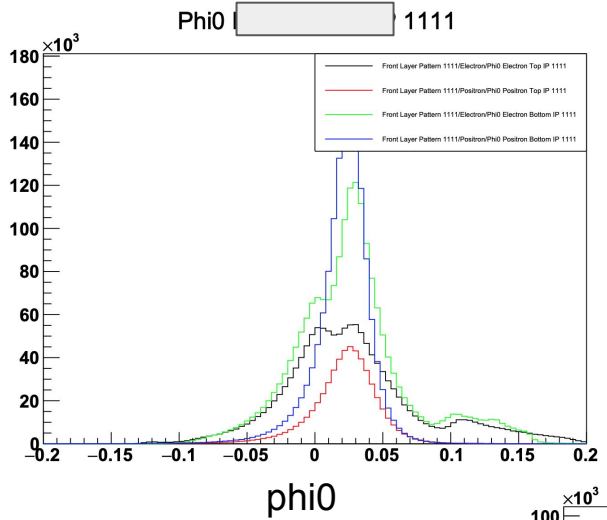
Black: Top Electron
Red: Top Positron
Green: Bot Electron
Blue: Bot Positron



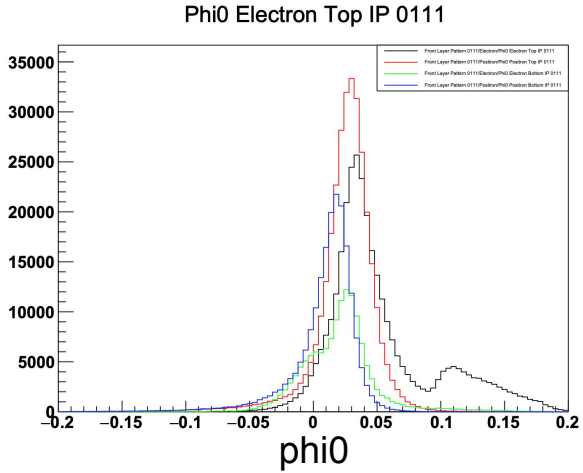
Mostly lower angles

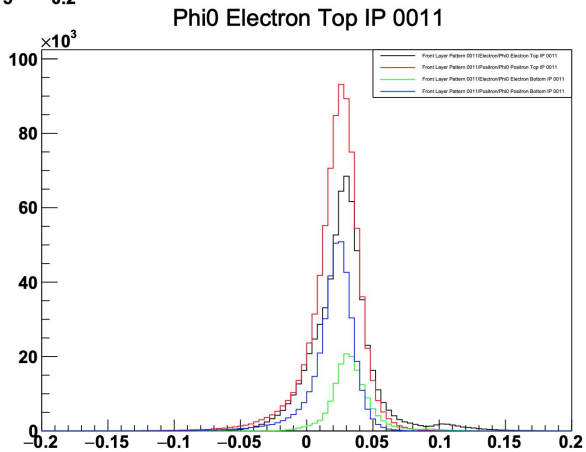


“1111”, “0111”, “0011” : phi0 (at 0,0,0)



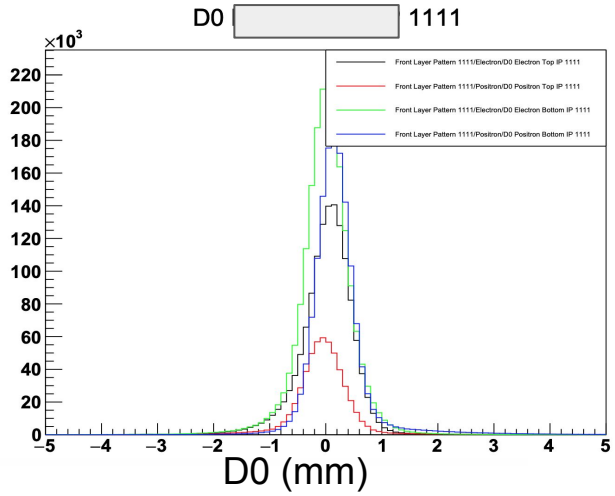
Black: Top Electron
Red: Top Positron
Green: Bot Electron
Blue: Bot Positron



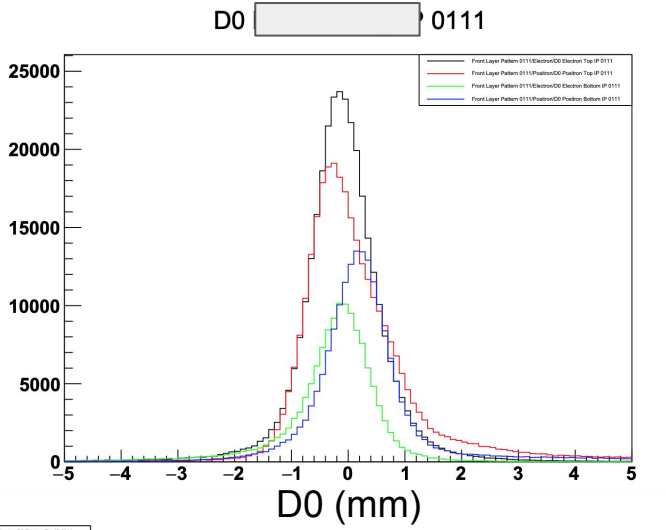


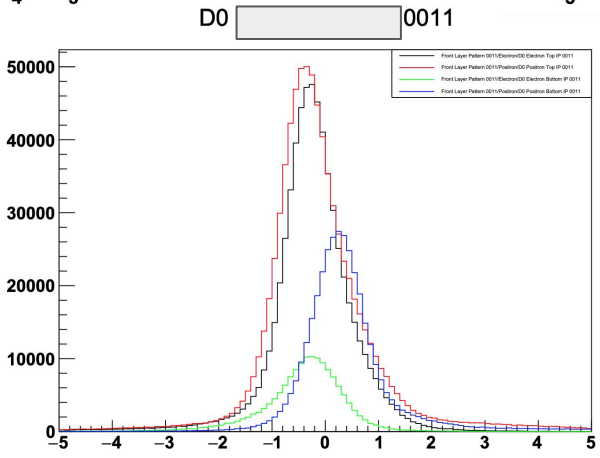
Very WEIRD!

“1111”, “0111”, “0011” : d0 (at 0,0,0)

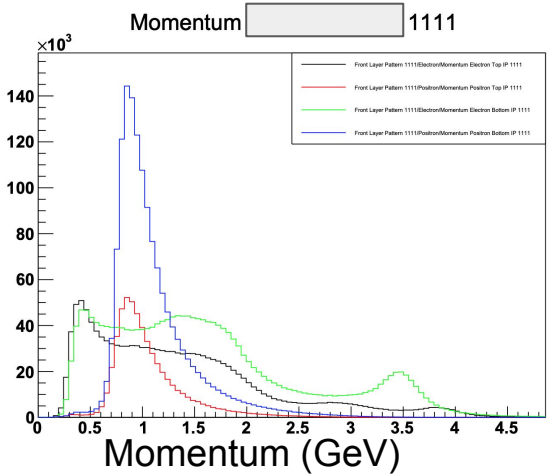


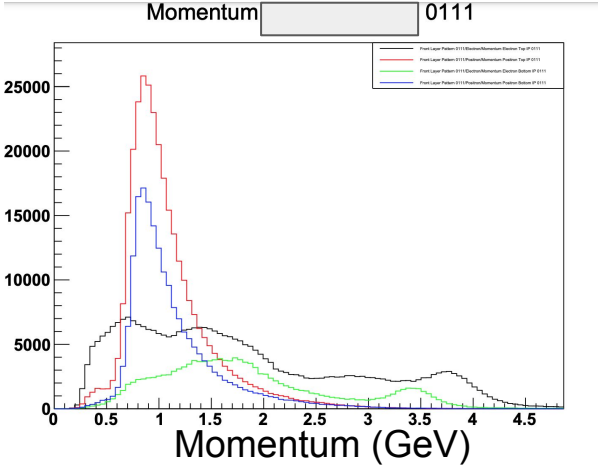
Black: Top Electron
Red: Top Positron
Green: Bot Electron
Blue: Bot Positron

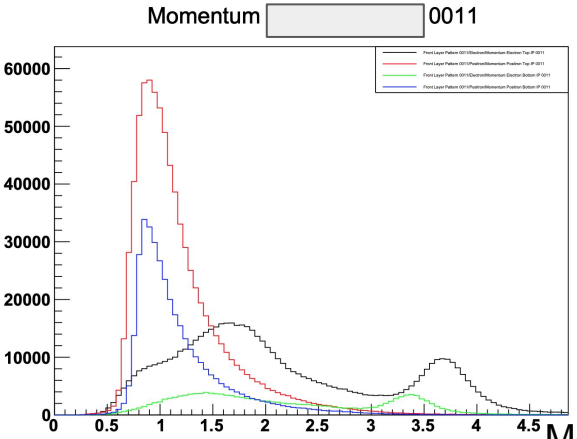




“1111”, “0111”, “0011” : Momentum







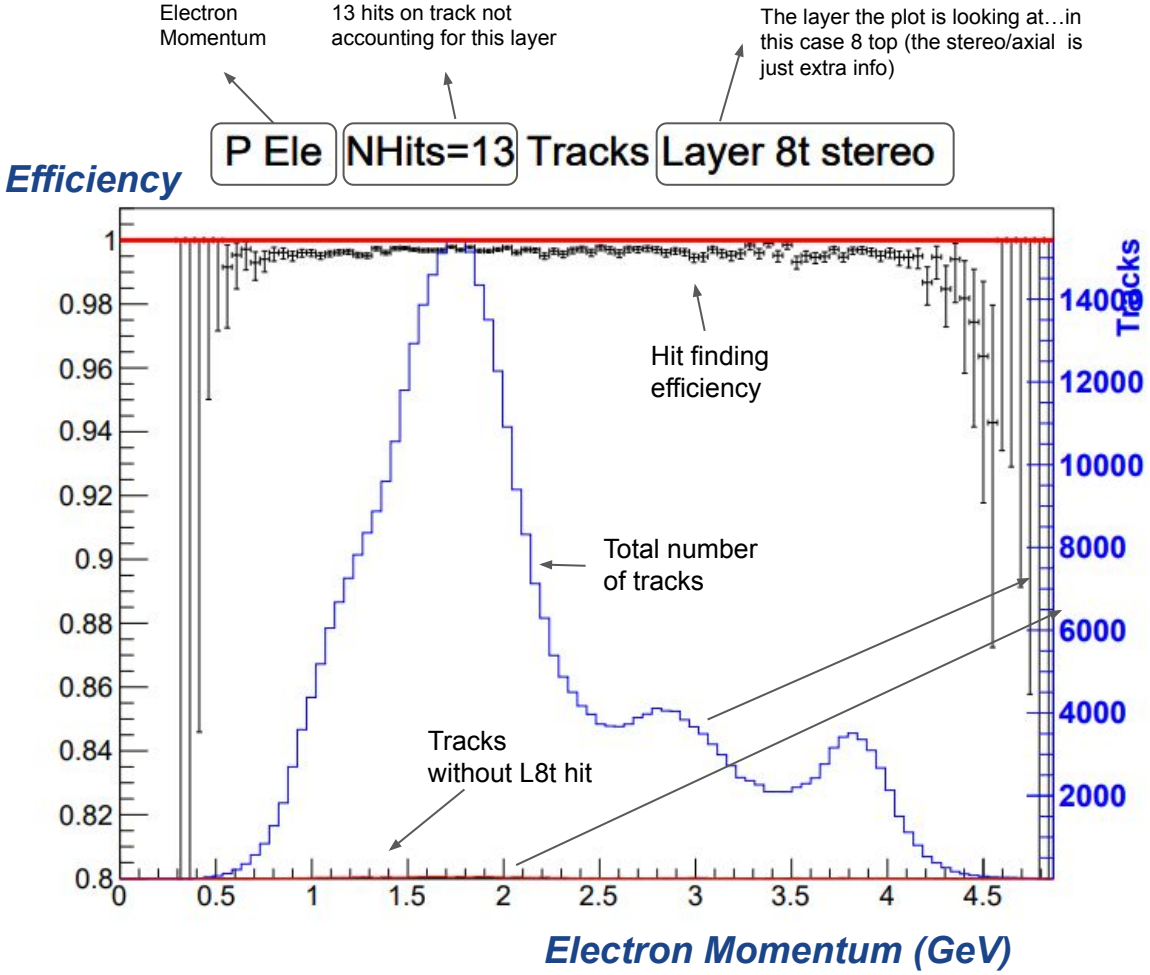
Defining “efficiency”

$$\frac{N(tracks - with - hit)}{N(tracks - in - acceptance)}$$

- Track: at least 8 hits (though I think this is required before a last pass to get rid of outlier hits)
  - Only a very loose cut on hit times in current kalman code
  - …and that’s it, I think…probably a very loose cut on chi2 in kalman code ?
- Denominator: tracks have to have trajectory that goes through active silicon
  - For these MeasurementSites with no smoothing result, mark at as outside of sensor (i.e. don’t count it)
  - Most of these sites without smoothing are in the back layers…likely low energy stuff?
    - I should look into these more closely
- Numerator: the track has a hit at that layer
  - Much more straightforward than with SeedTracker, where I had to do tracking pass without the layer of interest

Analysis Driver

- Run this in hps-java from the evio
- SvtHitEfficiencyKalman.java currently does this:
  - Gets tracks and intersections from the event
  - For each layer (1-14), checks if u-intercept is valid (not -999 and gives a good channel number for that sensor)
  - Checks if layer has a hit on track
  - Makes lots of plots
- Types of things I plot:
  - For each layer, # Tracks & # Tracks with hit on layer vs: unbiased “channel number” (calculated from intersection), track momentum
  - Also, those same plots for each number of hits on track (not including layer) 8-13 hits
  - Also, looked at layers hit and “holes” for each track
    - Hole is defined as when a track trajectory went through sensor but didn’t have a hit on track for that layer

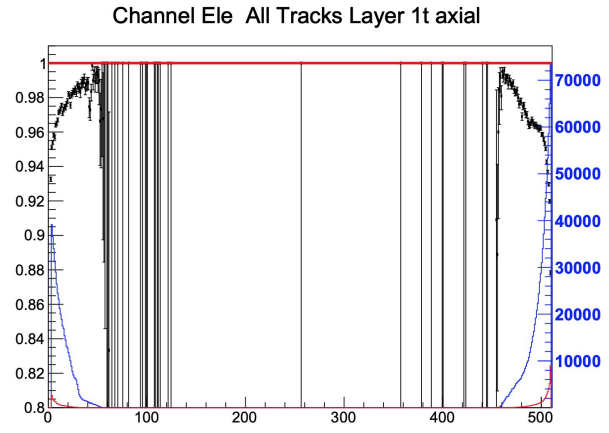


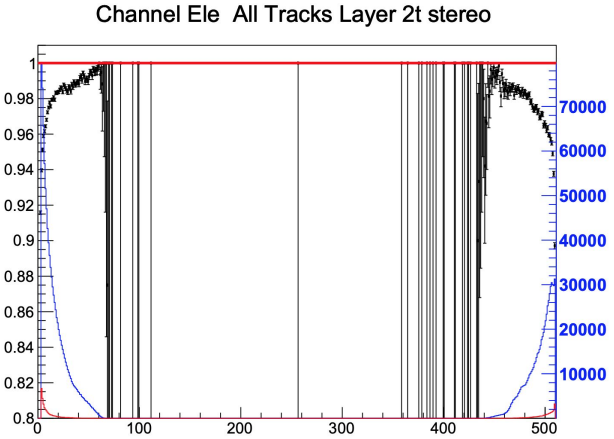
This is an example of the plots I make…not a great one (efficiency is too good) but here we are.

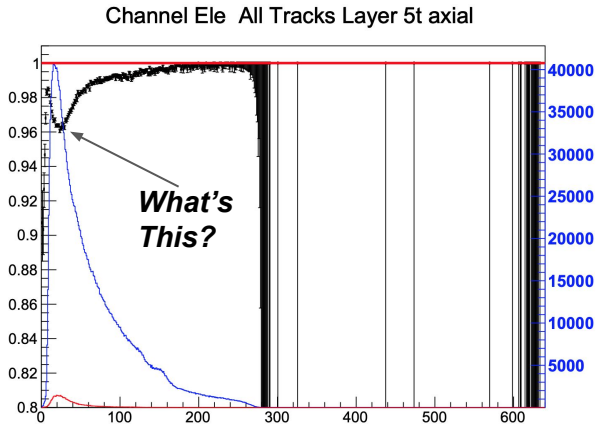
The efficiency (black) refers to left axis while the number of tracks (total and without a hit) is on the right.

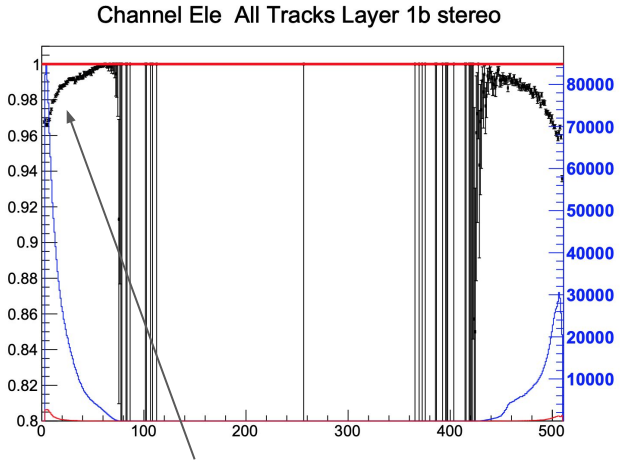
Note that the efficiency axis is zero suppressed!!!

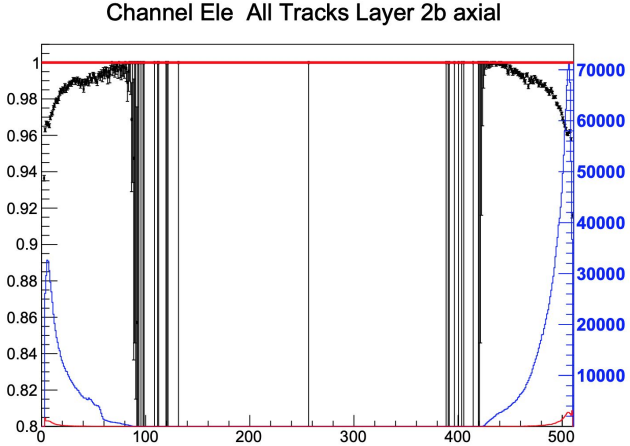
I scaled the # of tracks plots so that the max of the total tracks distribution to be 1 on the efficiency axis…

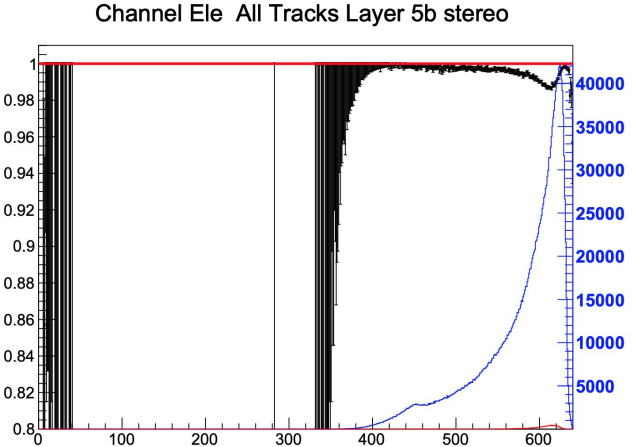










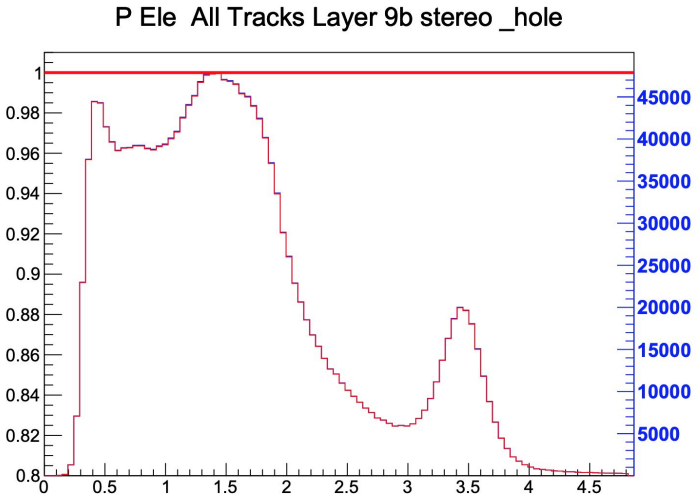


At least some of this edge drop is due to tracks missing the sensor

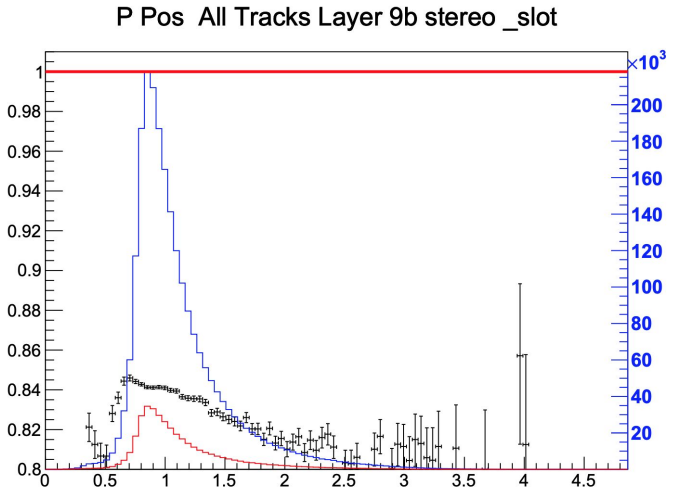
Hit-on-track efficiency

- https://www.slac.stanford.edu/~mgraham/TrkEfficiency2019/HitEfficiency/
  - Lots of plots in here…hit efficiencies for all layers, separated by positron/electron, vs channel number and momentum, and number of hits on track
  - Also non-efficiency track stuff like shown earlier in talk
- Overall, I think efficiencies look reasonable with a few caveats
  - See next slide for l9 bottom
  - This method can't claim that the ***right*** hit got put on the track…just that if found a hit to put on. Mis-hits are in there…
  -

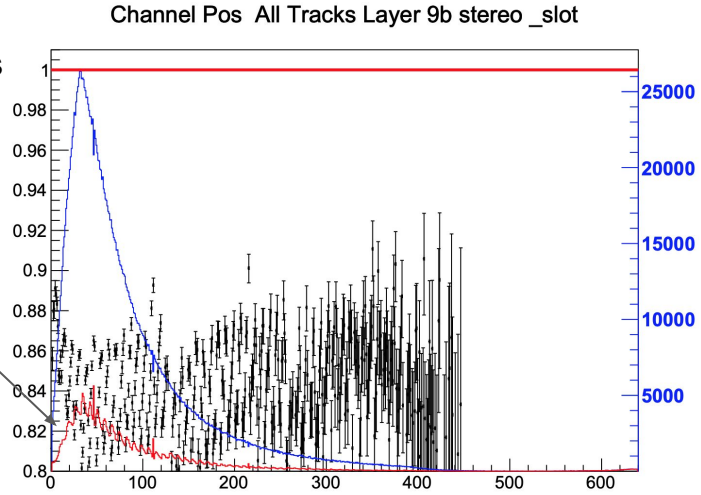
Layer 9 Bottom



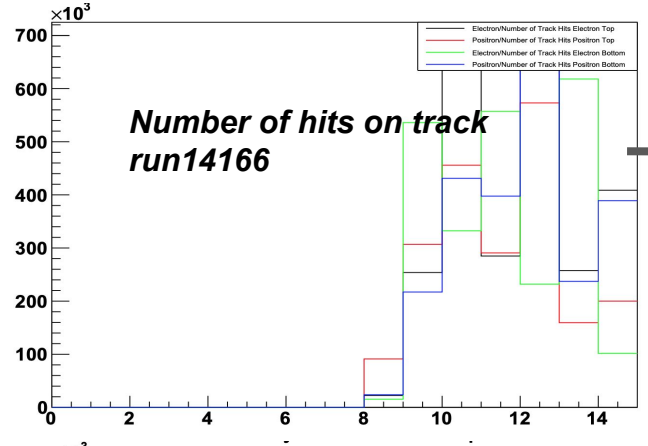
The hole sensor is just dead…weirdly there are a very few “hits” but I think they are probably from the slot side from tracks that were very close to edge between hole/slot

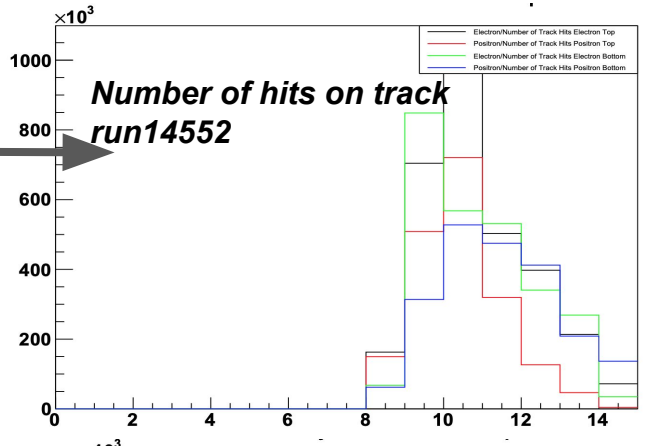


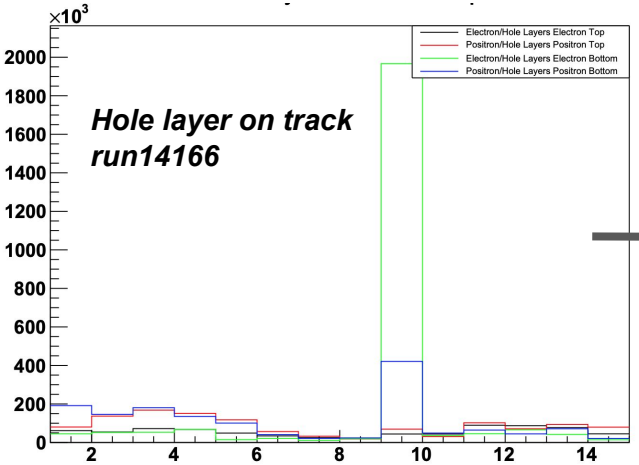
The slot side was very noisy…see this funny channel pattern for misses

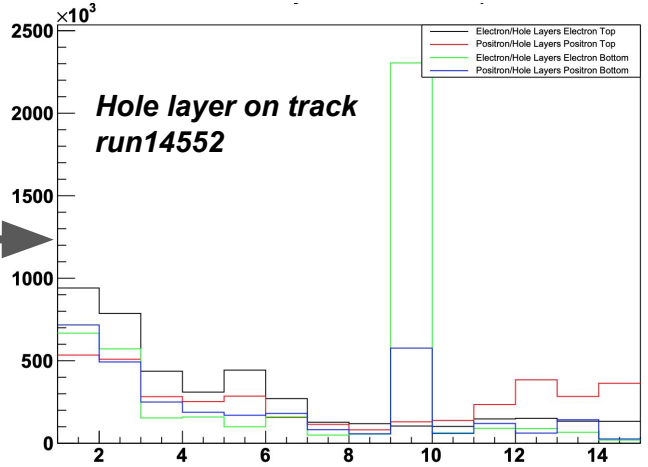


Looking at run14552: later run, nominal lumi

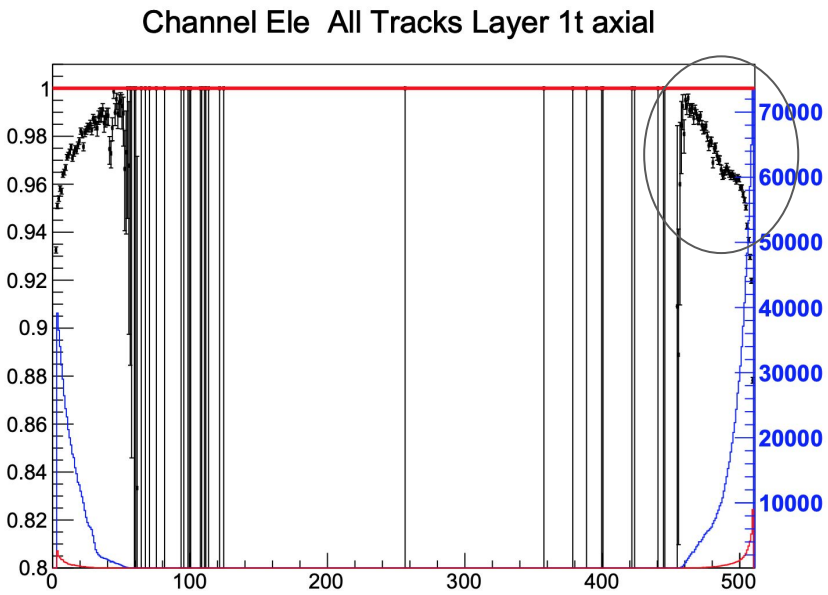




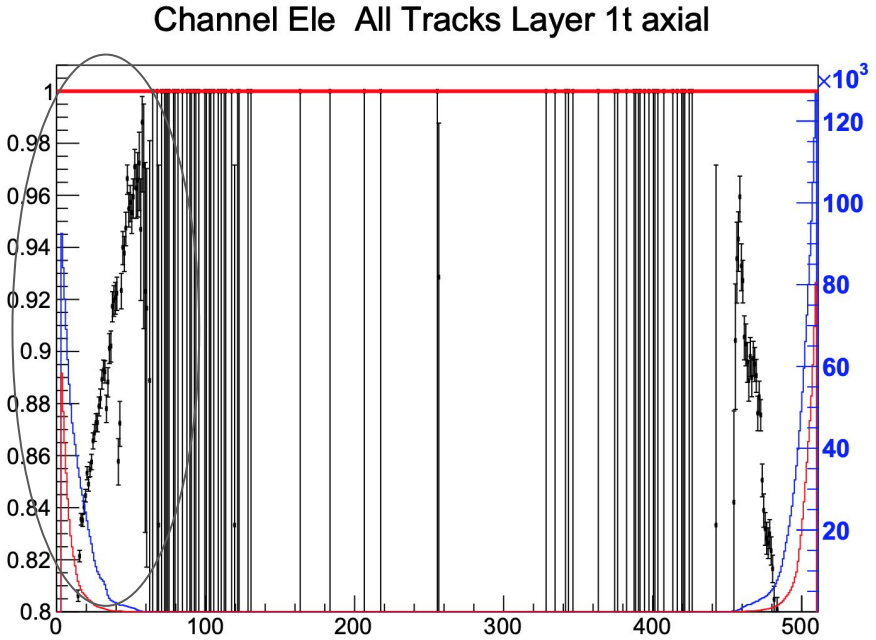




Efficiency vs Channel run 14552



Run 14166

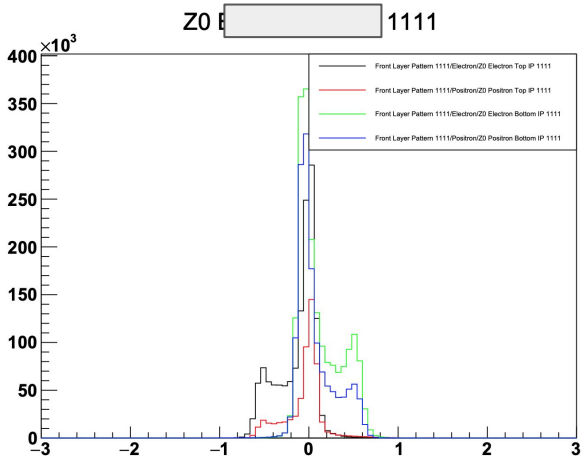


Run 14552

Run 14552 show much more hit inefficiency compared to the low lumi (and earlier) run 14166….

Take-home messages

- We have code that computes the hit-on-track efficiency for each layer with Kalman tracks…
- The low-rate run shows pretty good efficiencies/layer (~98%)...later, nominal rate runs show much lower efficiencies
  - This is not track-finding efficiencies (but is related)
- Understanding the efficiency on the inner layers is very important for vertexing analyses (including SIMPs, iDM etc)
- …


Z0 E 1111
×10^3
400
350
300
250
200
150
100
50
0
−3
−2
−1
0
1
2
3
Front Layer Pattern 1111/Electron/Z0 Electron Top IP 1111
Front Layer Pattern 1111/Positron/Z0 Positron Top IP 1111
Front Layer Pattern 1111/Electron/Z0 Electron Bottom IP 1111
Front Layer Pattern 1111/Positron/Z0 Positron Bottom IP 1111