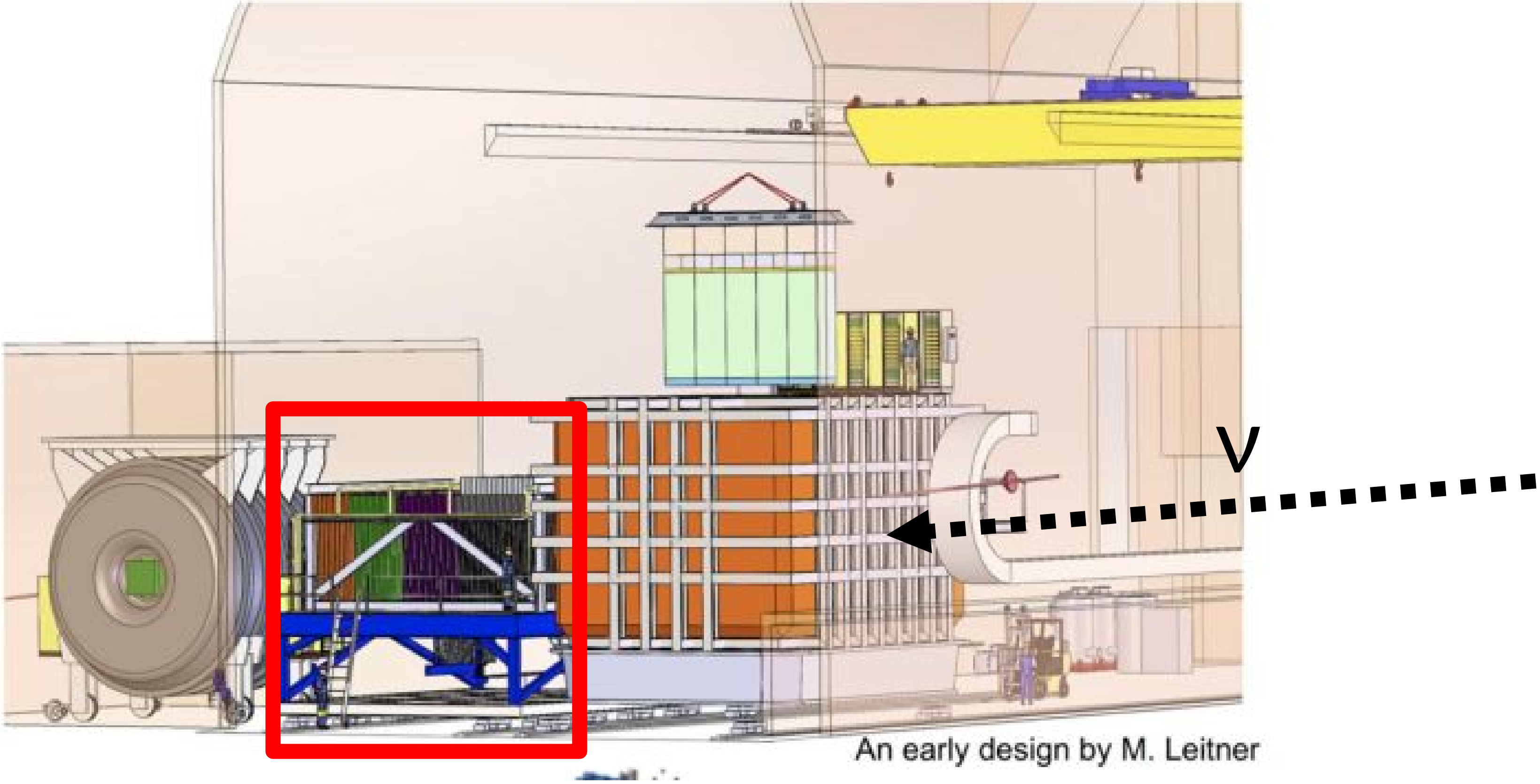
Reconstruction in the DUNE Temporary Muon Spectrometer ND

Clarence Wret

Rochester meeting
14 Dec 2020

Introduction

- Wait, what detector?!

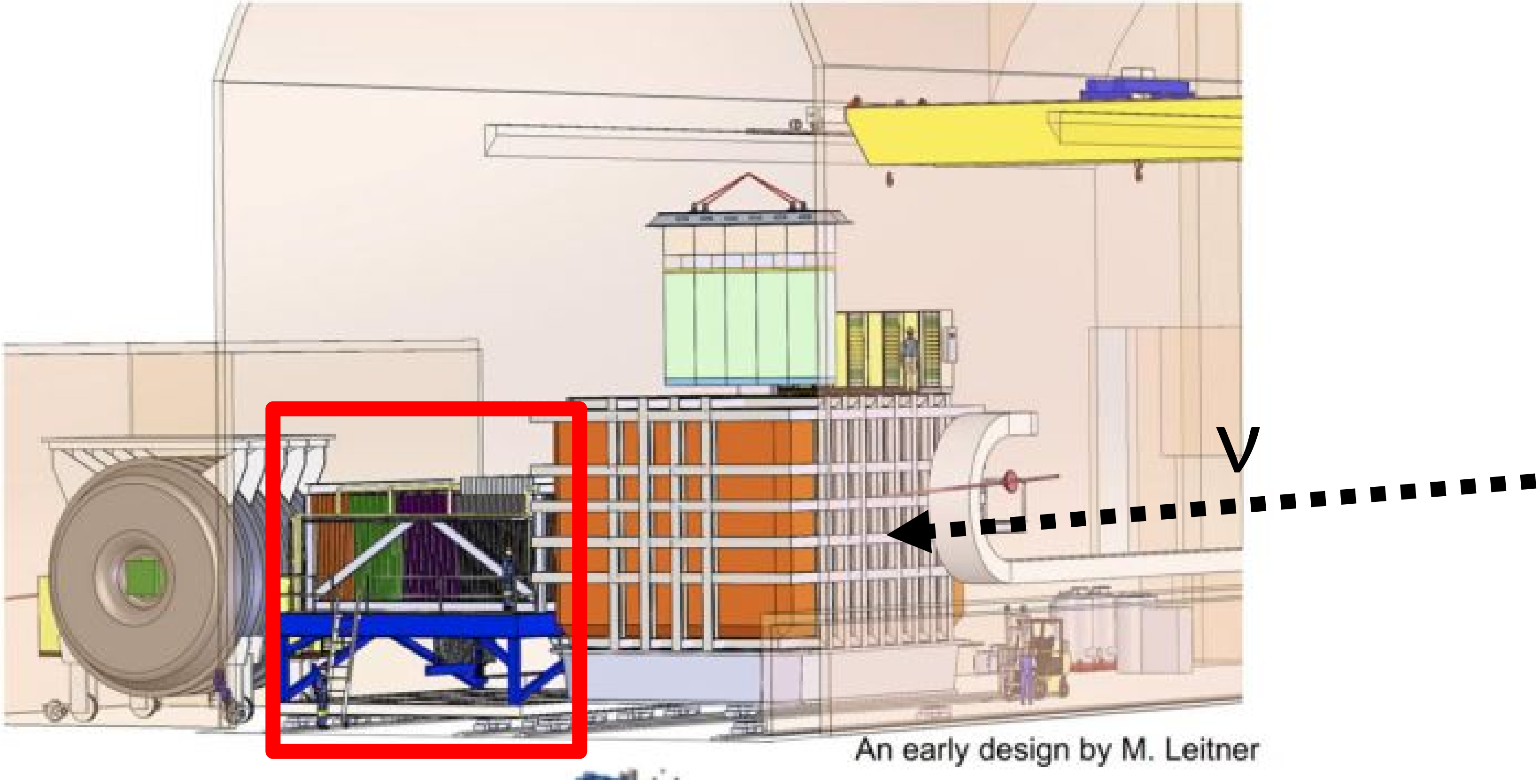


An early design by M. Leitner

The DUNE ND with the lovely GAr violently removed and in it's place behold MINOS 2.0

Introduction

- Wait, what detector?!



An early design by M. Leitner

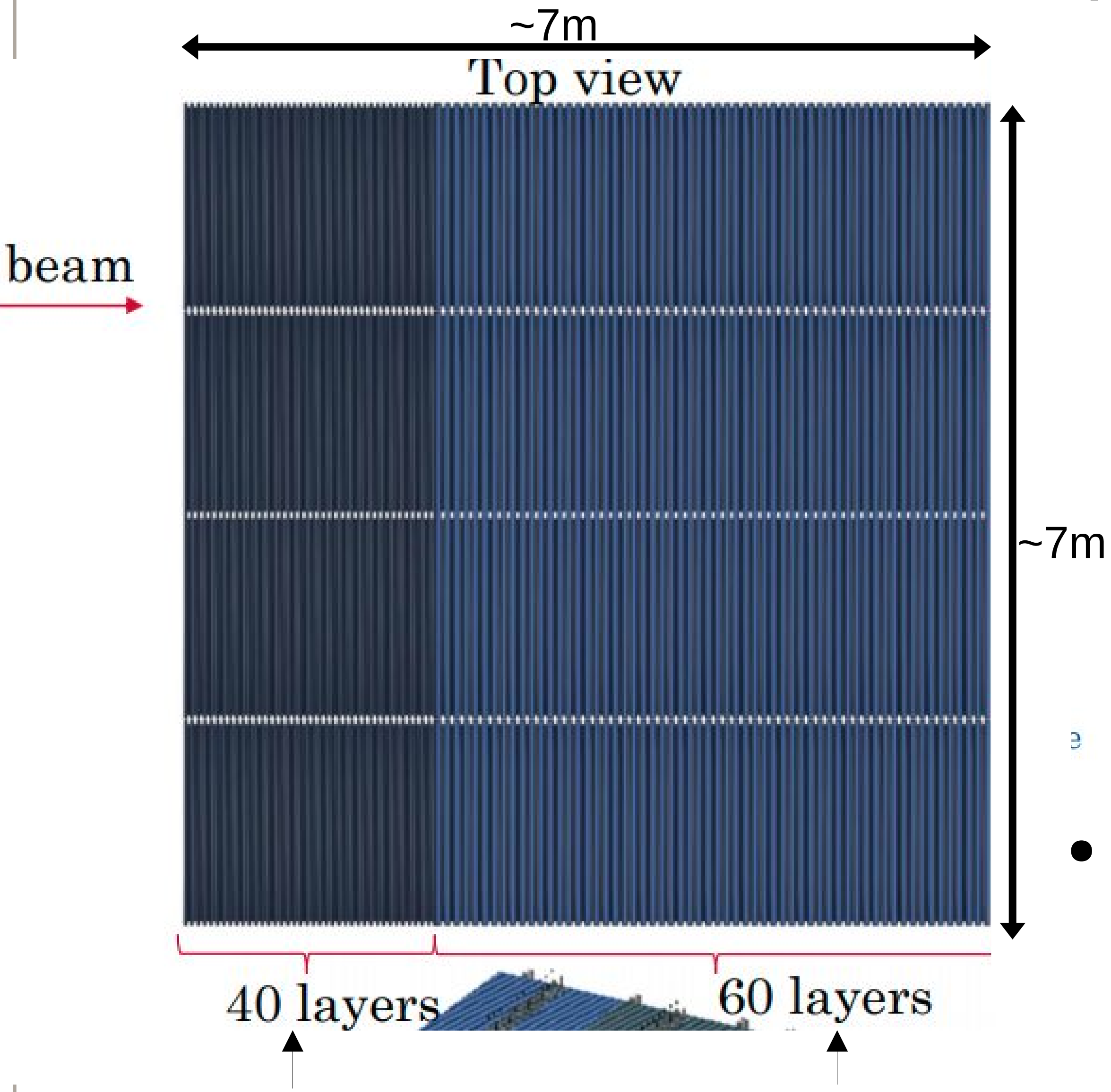
Sufficient to do DUNE CPV studies for the first *N* years

N~3?

Will then likely need GArTPC

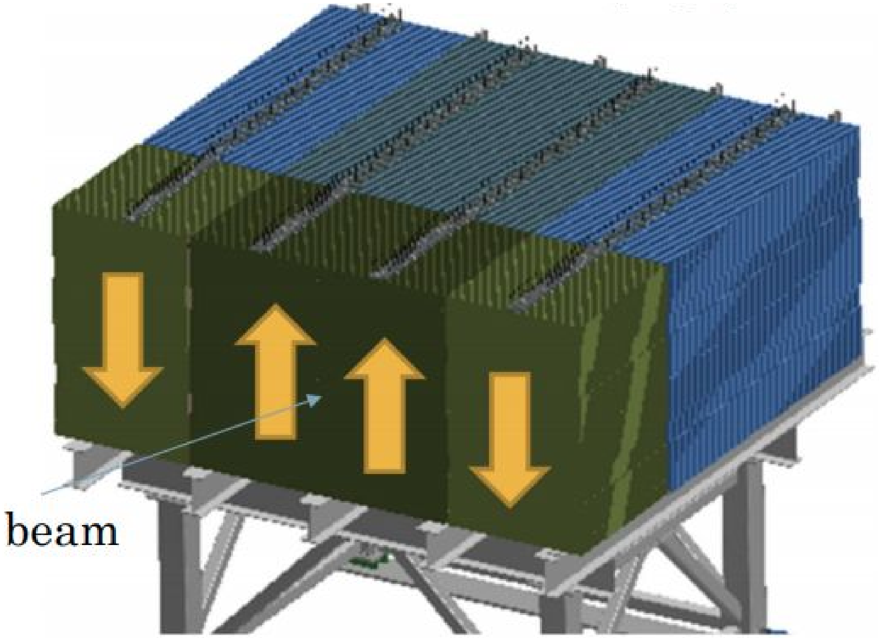
Temporary Muon Spectrometer

- Plastic scintillator/steel sampling calorimeter



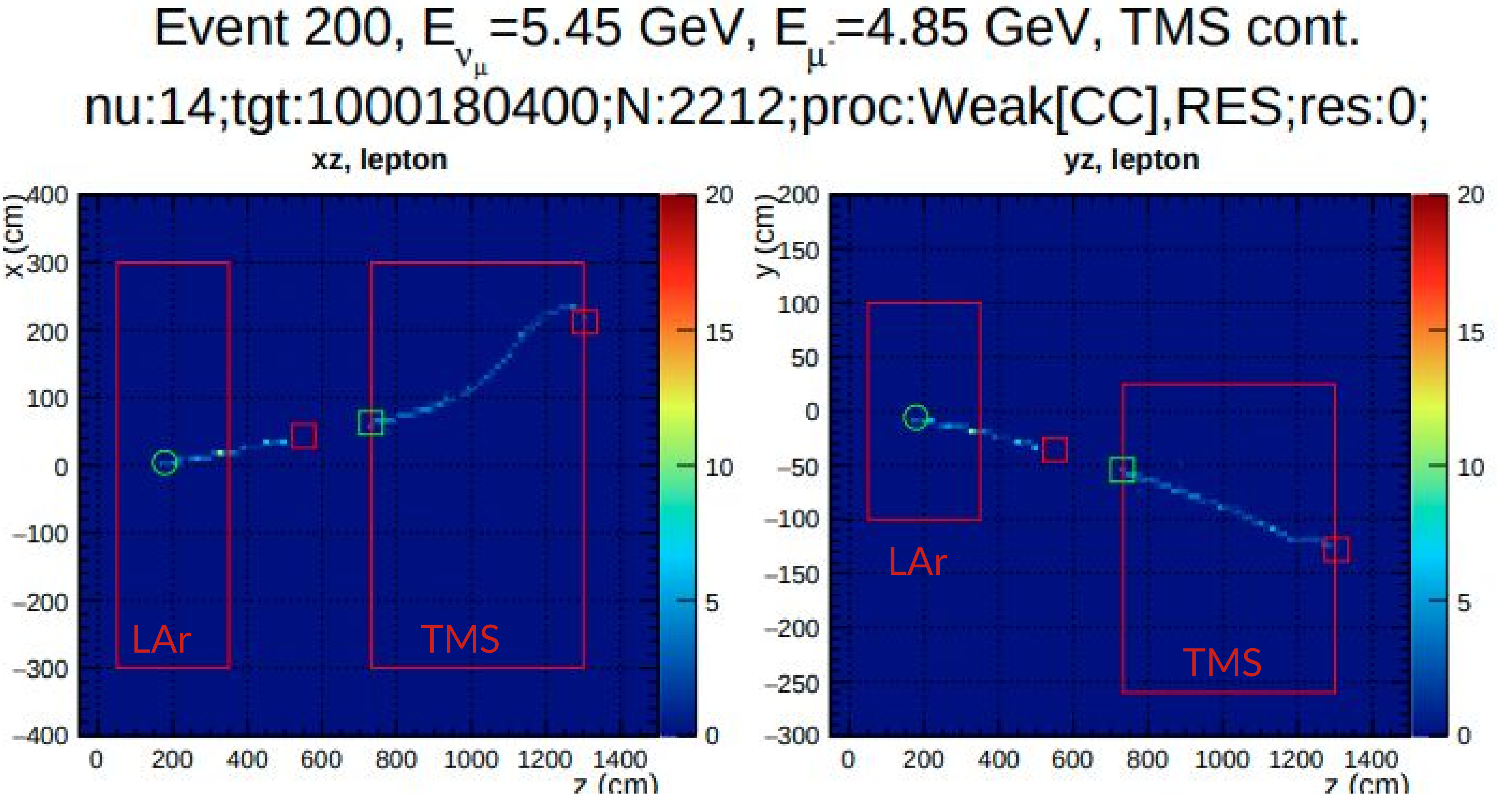
1cm scint, 1.5cm steel

1cm scint, 4cm steel



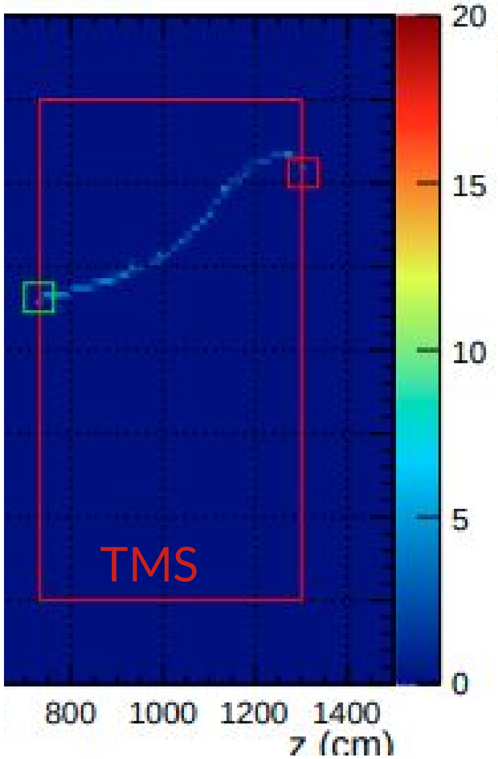
- Steel magnetised at 0.9-1.2T

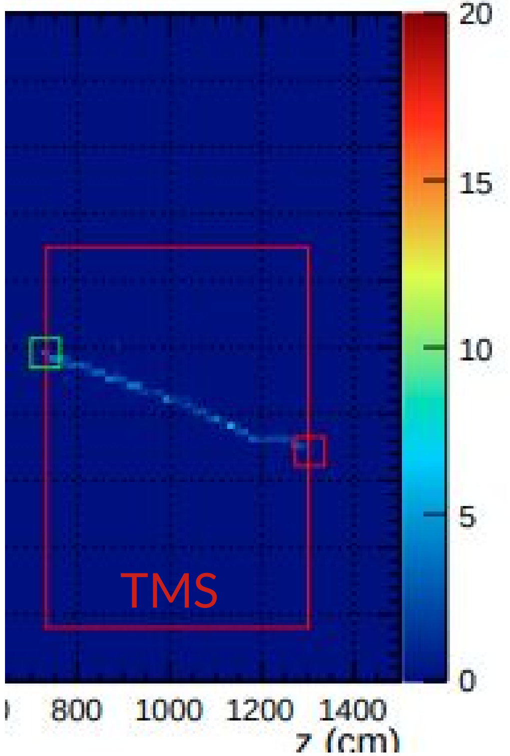
Event displays



- Want stand-alone reconstruction in the TMS
  - Communicates with LAr after LAr and TMS reconstruction is complete
- Maybe one day joint reconstruction, but not today

Event displays





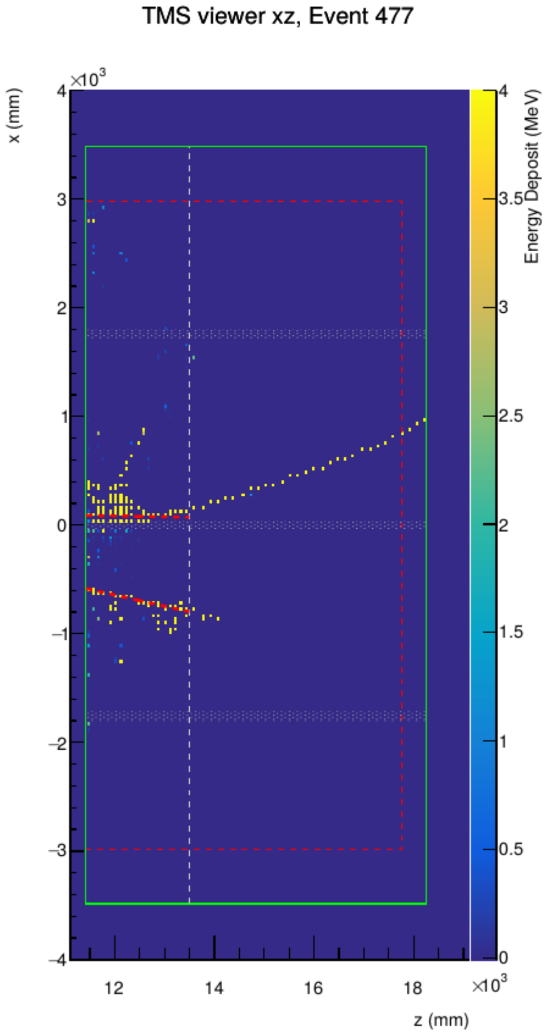
- Want stand-alone reconstruction in the TMS
    - Communicates with LAr after LAr and TMS reconstruction is complete
- Maybe one day joint reconstruction, but not today

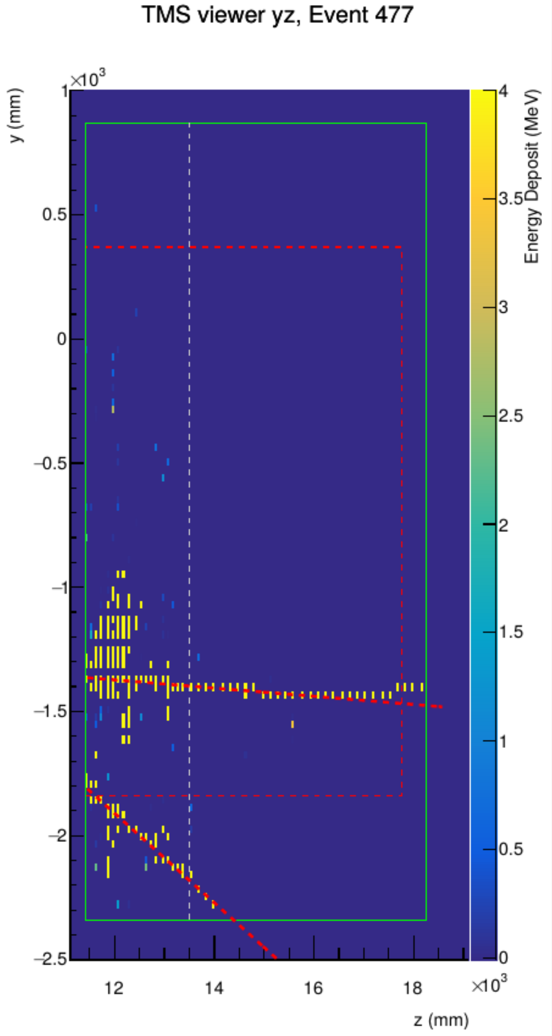
General idea

- Wrote up data products, now let's use them!
- Simplistic track finding to pick up muon track
  - Only trying to reconstruct tracks going through TMS for now
  - Prefer a greedy algorithm since Kalman filtering will be applied at later stage: let Kalman decide on good/bad hits
  - Can run iteratively in case of multiple tracks
- Feed track candidates into a Kalman filter
  - Deduce track momentum from energy loss, multiple scattering and bending in magnetic field
  - Kalman filter discards hits above some man-made χ^2 metric
  - Run iteratively; back-filtering and smoothing
- No PID plans yet, could **maybe** use deposited energy

Track finding, Part I

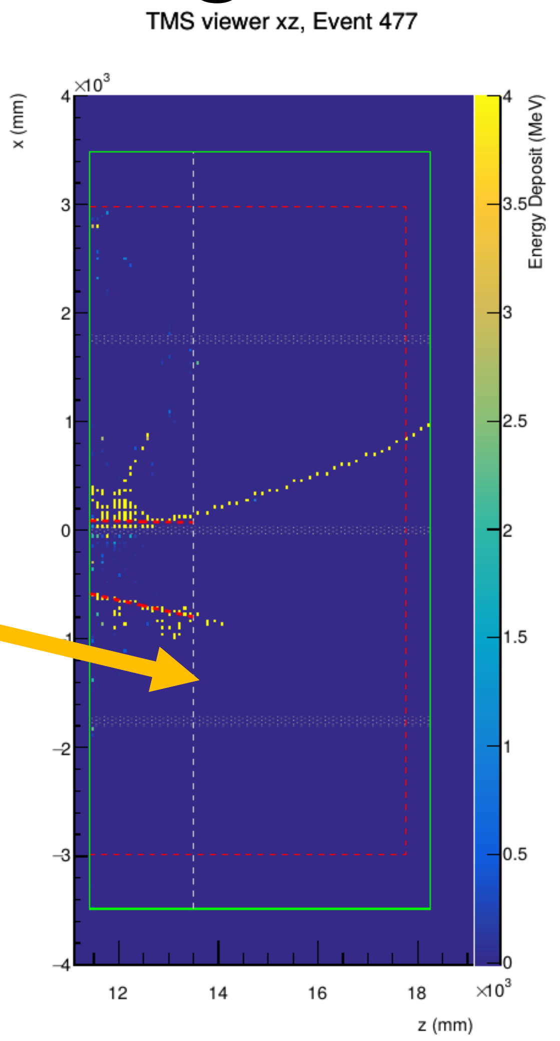
- Hough transform of linear type in both views
  - yz view likely won’t be realised ($$$, SAD!)

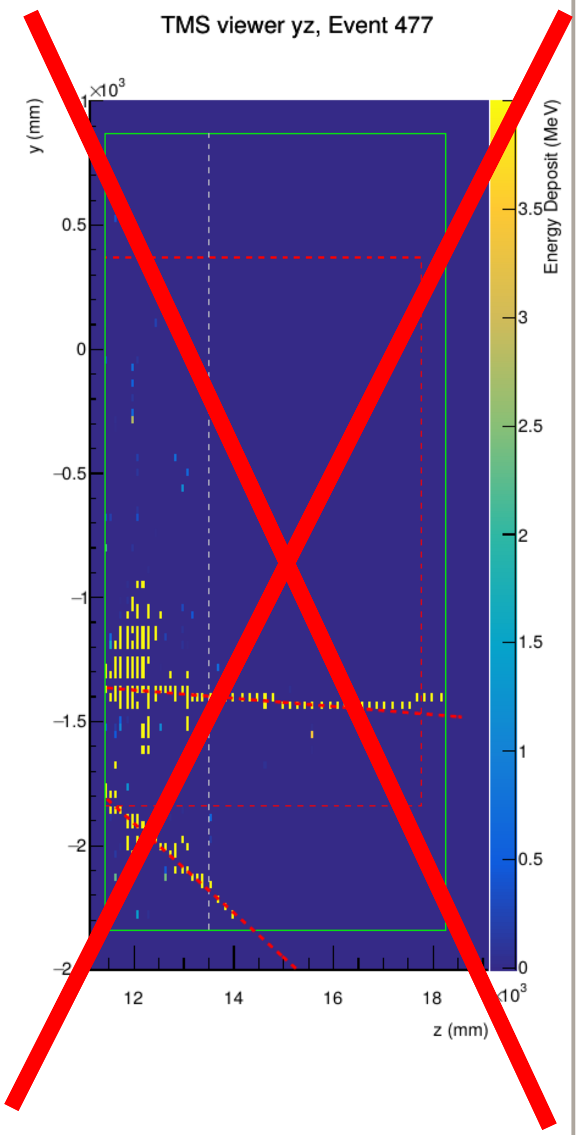




Track finding, Part I

- Hough transform of linear type in both views
  - yz view likely won’t be realised ($$$, SAD!)
  - Hough transform in xz not awesome because of bending: transform only up until transition layer
  - Merge neighbouring hits

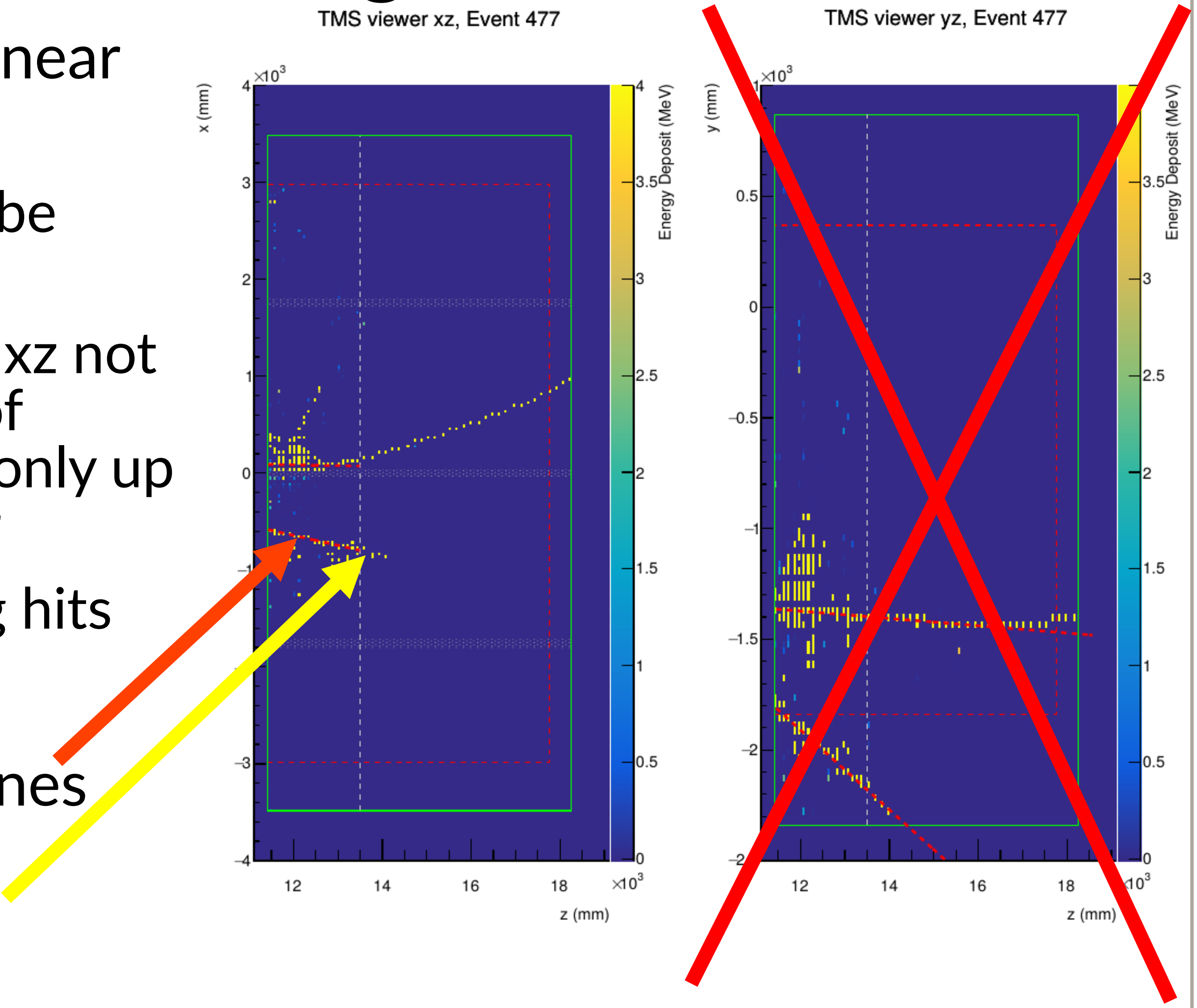




Track finding, Part I

- Hough transform of linear type in both views
  - yz view likely won’t be realised ($$$, SAD!)
  - Hough transform in xz not awesome because of bending: transform only up until transition layer
  - Merge neighbouring hits

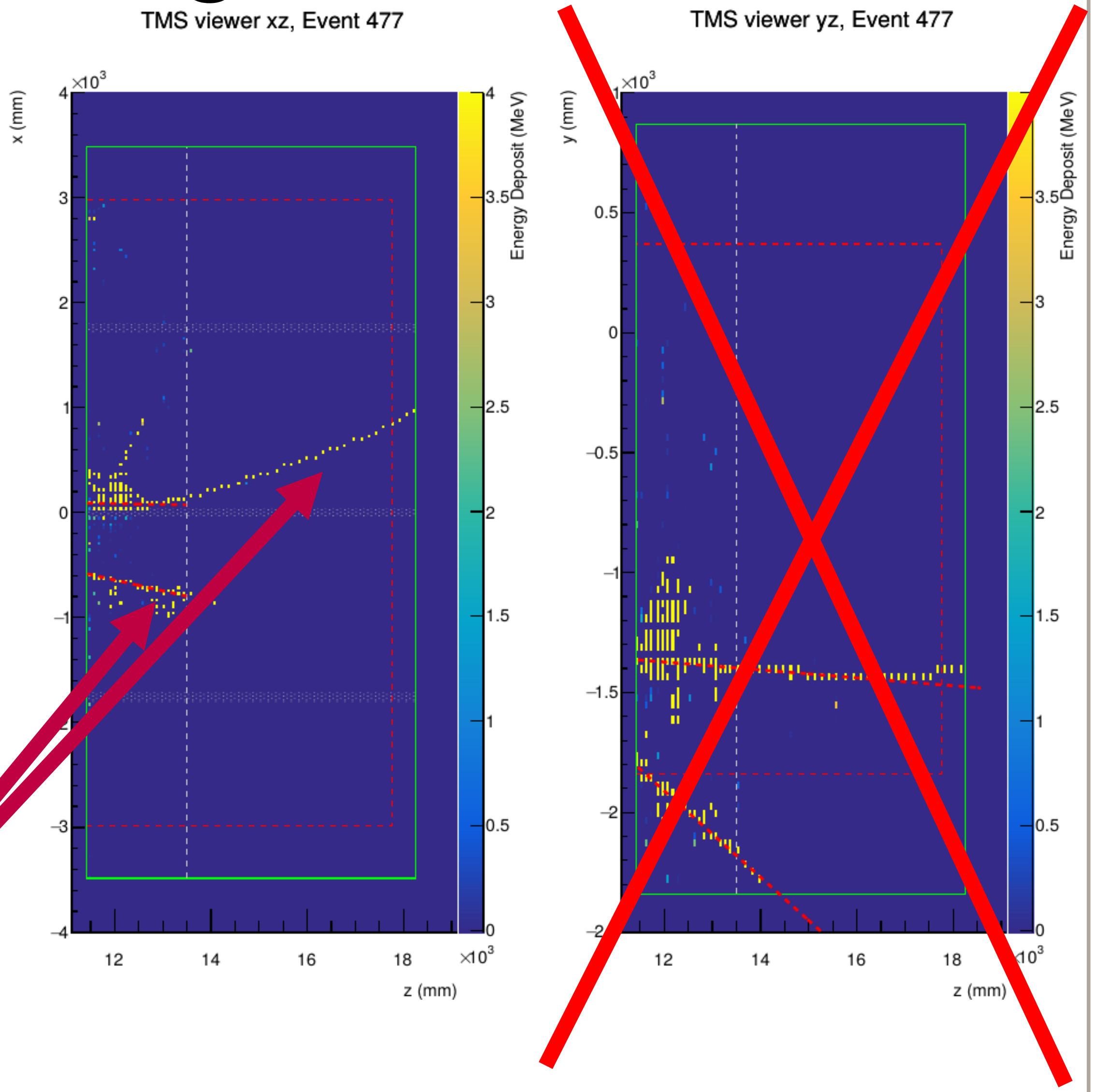
- Red dashed: Hough lines
- Yellow hits: Found by Hough+merging



Track finding, Part I

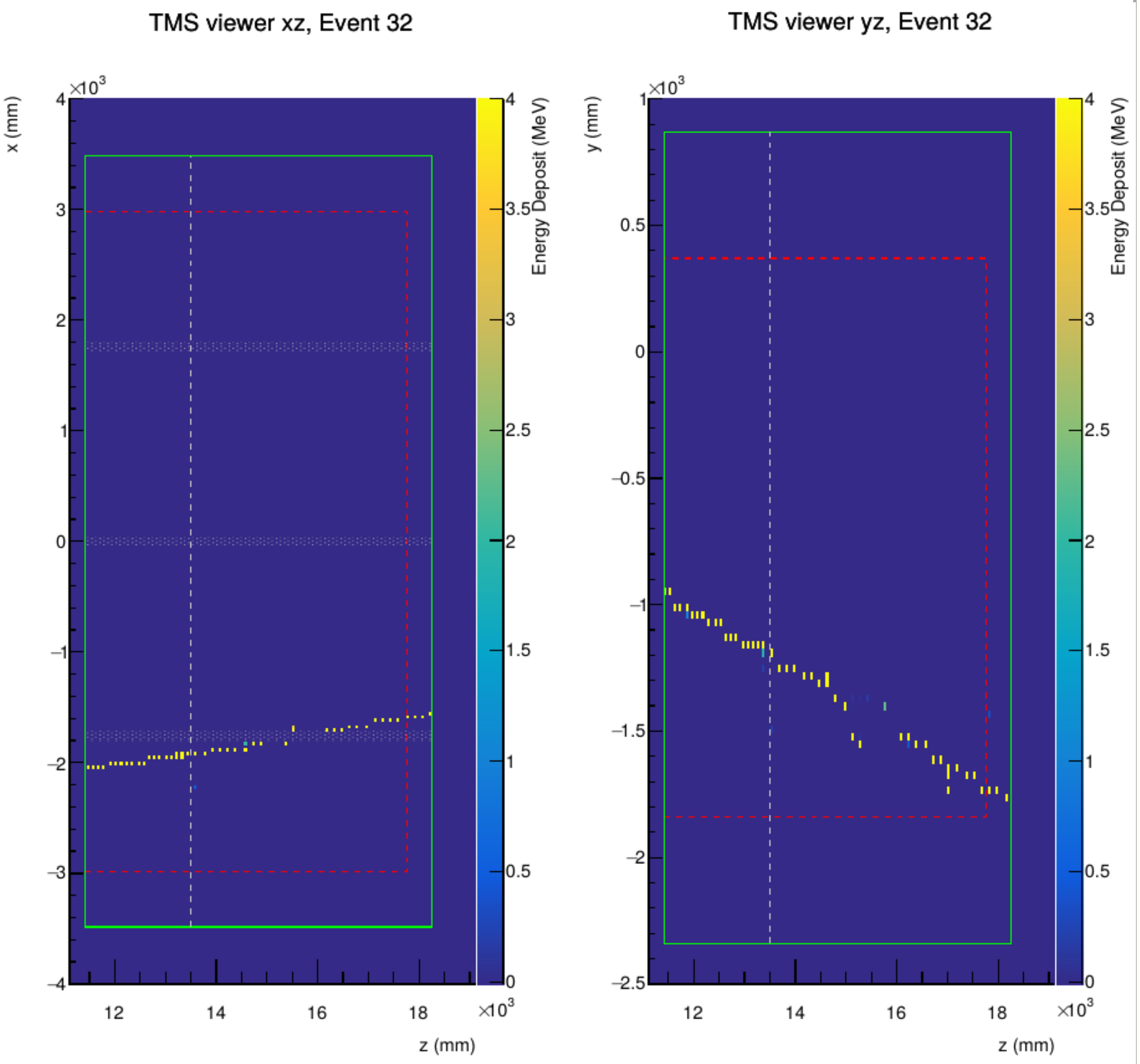
- Hough transform of linear type in both views
  - yz view likely won’t be realised ($$$, SAD!)
  - Hough transform in xz not awesome because of bending: transform only up until transition layer
  - Merge neighbouring hits

- Hough transform until N% of hits are included
  - Get multiple tracks
  - Creates ghosts, but I’m ok with that: Kalman filter will remove them



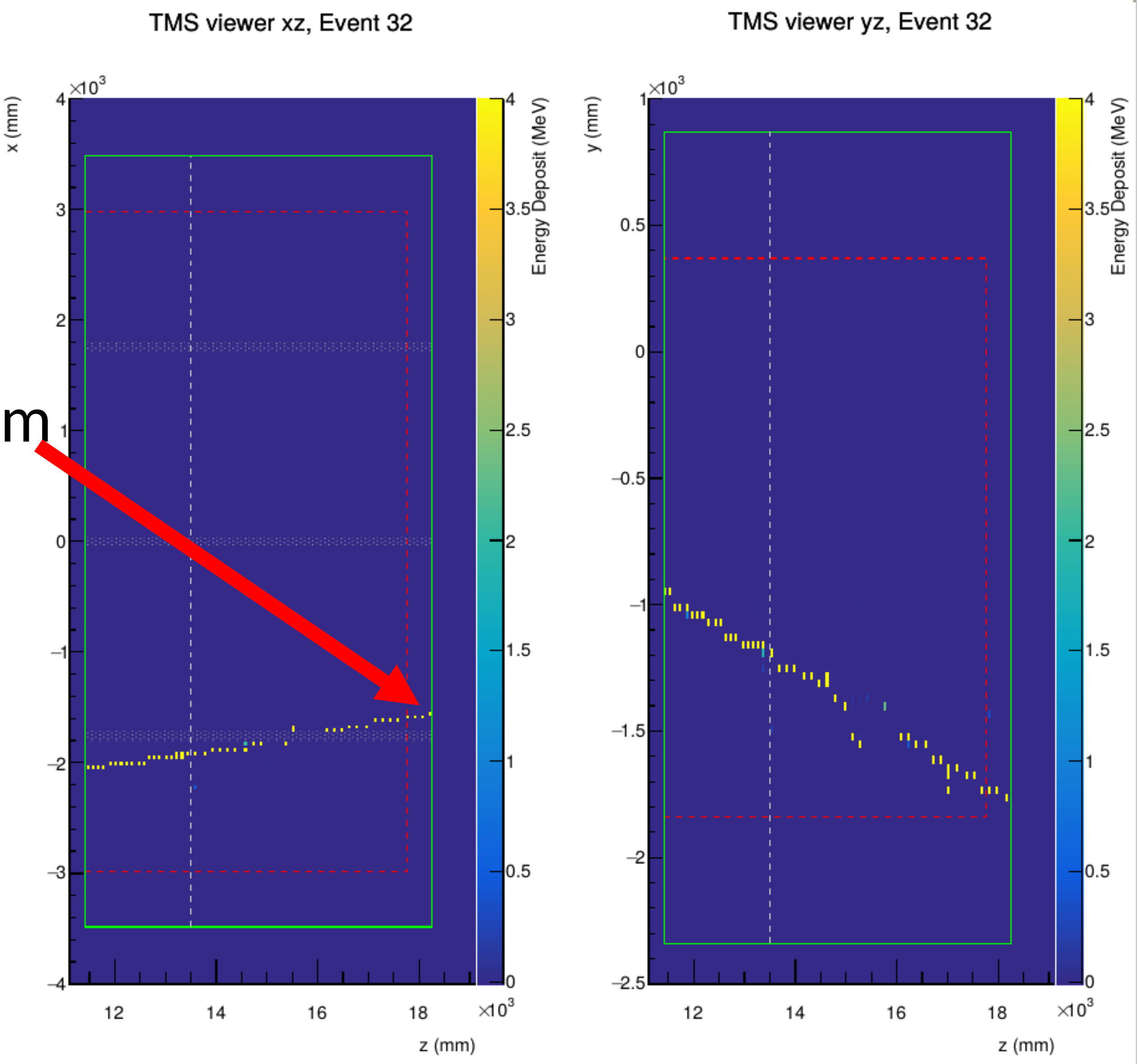
Track finding, Part II

- Wanted an alternative smarter algorithm for comparison
  - A* graph traverse algorithm



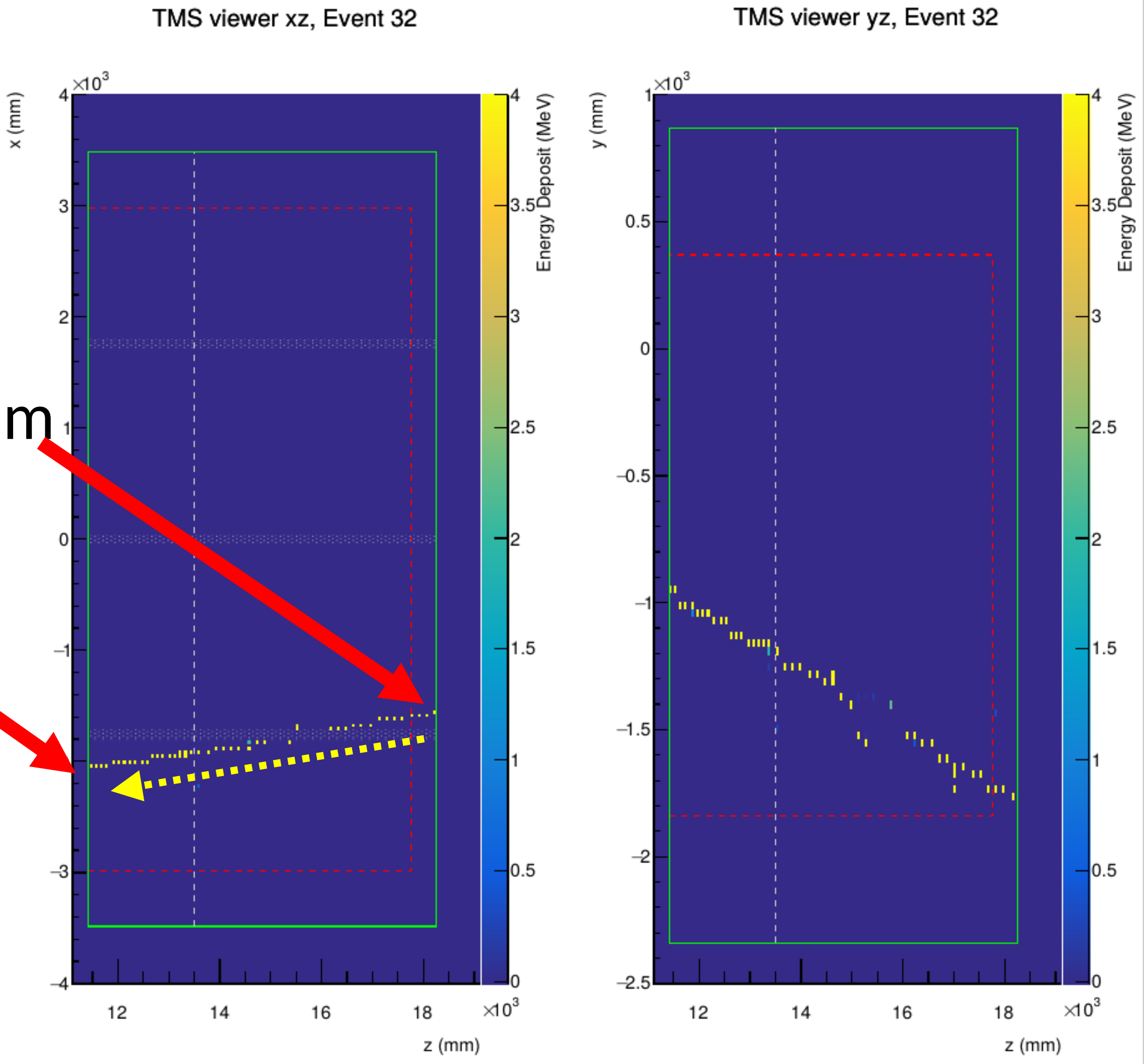
Track finding, Part II

- Wanted an alternative smarter algorithm for comparison
  - A* graph traverse algorithm
- Start at most downstream hit, find optimal path to connect to most upstream hit



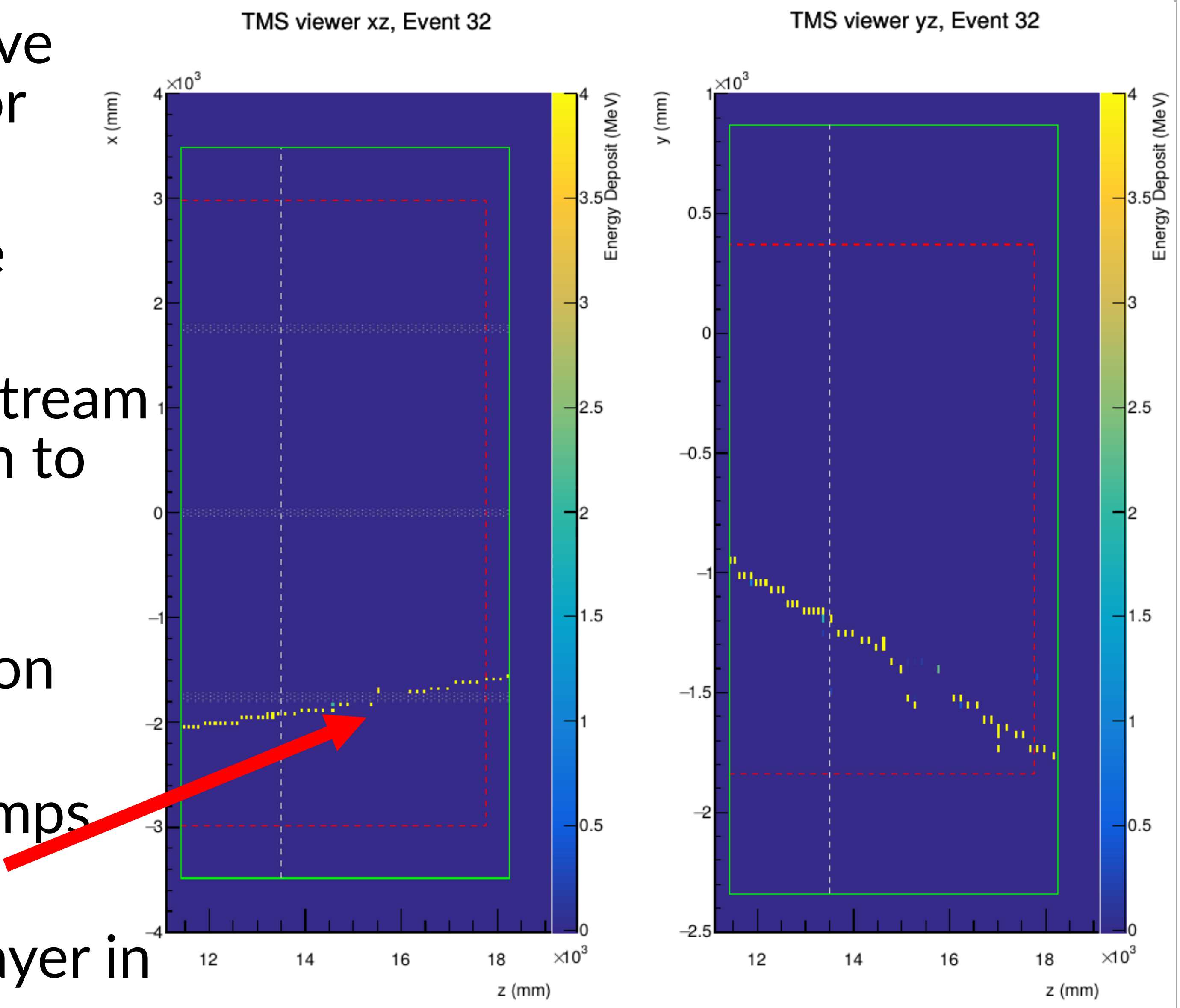
Track finding, Part II

- Wanted an alternative smarter algorithm for comparison
  - A* graph traverse algorithm
- Start at most downstream hit, find optimal path to connect to most upstream hit



Track finding, Part II

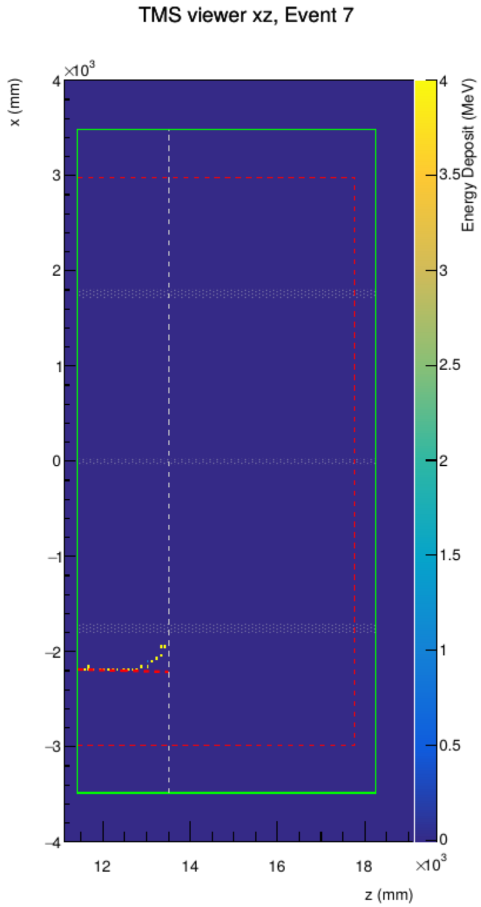
- Wanted an alternative smarter algorithm for comparison
  - A* graph traverse algorithm
- Start at most downstream hit, find optimal path to connect to most upstream hit
  - Use a cost function connecting hits
  - Allowing large jumps in gap regions
  - Allow to miss 1 layer in x, no misses in z (unless near gap region)

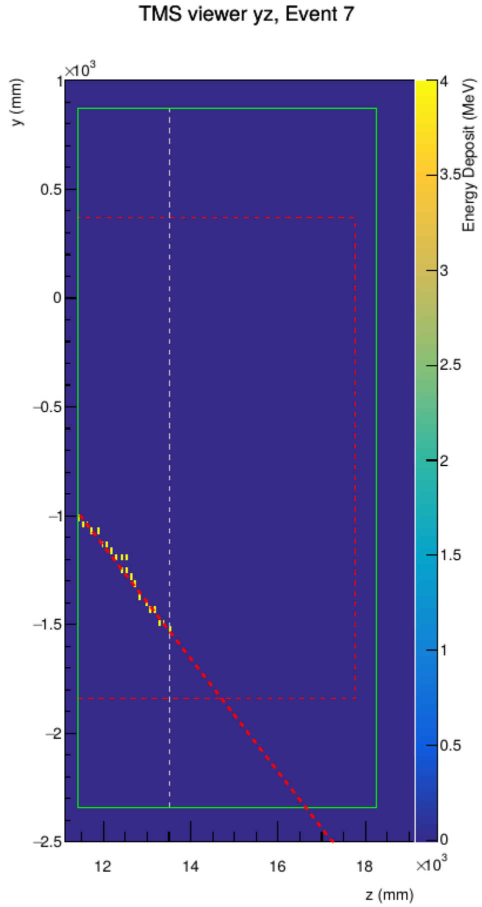


Track fitting

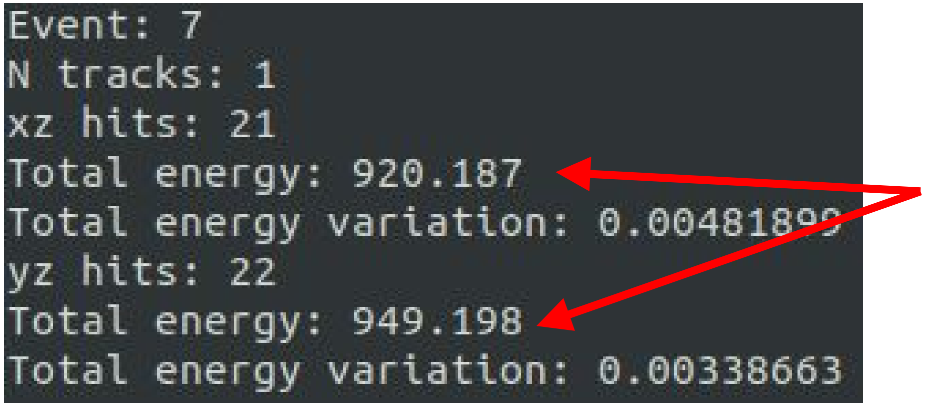
- Kalman filter is basically an optimal solution finder for a process with Gaussian noise
- Correlate hits across the detector, predict the next (upstream or downstream) hit, and iterate
  - Back propagate and smooth once other noise hits are discarded
- Could technically run Geant4 to do all of the physics
  - But very slow, and accuracy is not necessary
- Decided to implement my own energy loss, multiple scattering and magnetic field functions
  - … essentially a mini-generator, surprisingly rewarding
  - Most of my DUNE time spent on this

So what’s the results?





Single iteration Kalman fitter (with **questionable** covariance!)

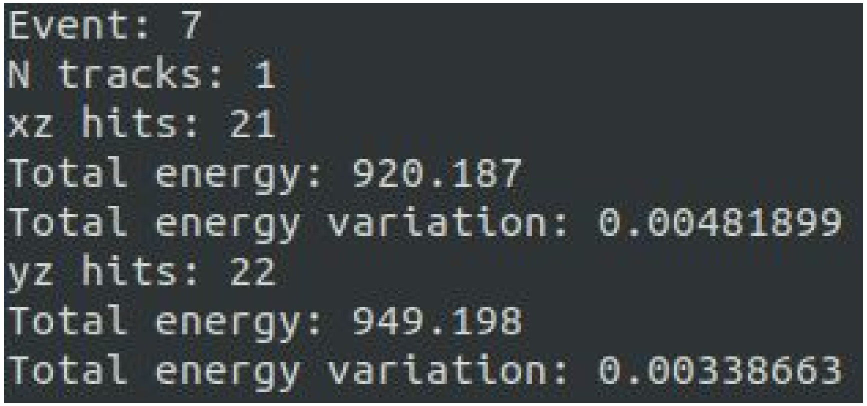


```
Event: 7
N tracks: 1
xz hits: 21
Total energy: 920.187
Total energy variation: 0.00481899
yz hits: 22
Total energy: 949.198
Total energy variation: 0.00338663
```

Similar-ish energy estimates and uncertainties in both views

So what’s the results?

Single iteration Kalman fitter (with **questionable** covariance!)



```
Event: 7
N tracks: 1
xz hits: 21
Total energy: 920.187
Total energy variation: 0.00481899
yz hits: 22
Total energy: 949.198
Total energy variation: 0.00338663
```

- Finish up matrix multiplications for more Kalman iterations
  - This week
- Total energy variation is too small? Maybe unit conversion error or something else silly
- Uncertainty from multiple scattering looks spot on (not shown)
- Magnetic field integrator not working
  - Run without for preliminary results
  - Shouldn’t be a problem because Kalman filter has few hits to choose from (most of the time a single muon track): should just mean a bad χ^2
  - Revise my integrator and profit
- Then finally evaluate the goodness of the estimators
  - Data products will require some loving

Thanks