pi0 Shower Reconstruction: Cluster Merging

Mike Wallbank

Thanks as always to Tingjun, Robert & Dorota

19/8/2015

Recap

- Have the pi0 clustering in 2D; now trying to use these to make showers in 3D.

- Robert & Dorota use the 2D clusters to match between the views and find 3D tracks, with associated hits, which can be used to make showers.

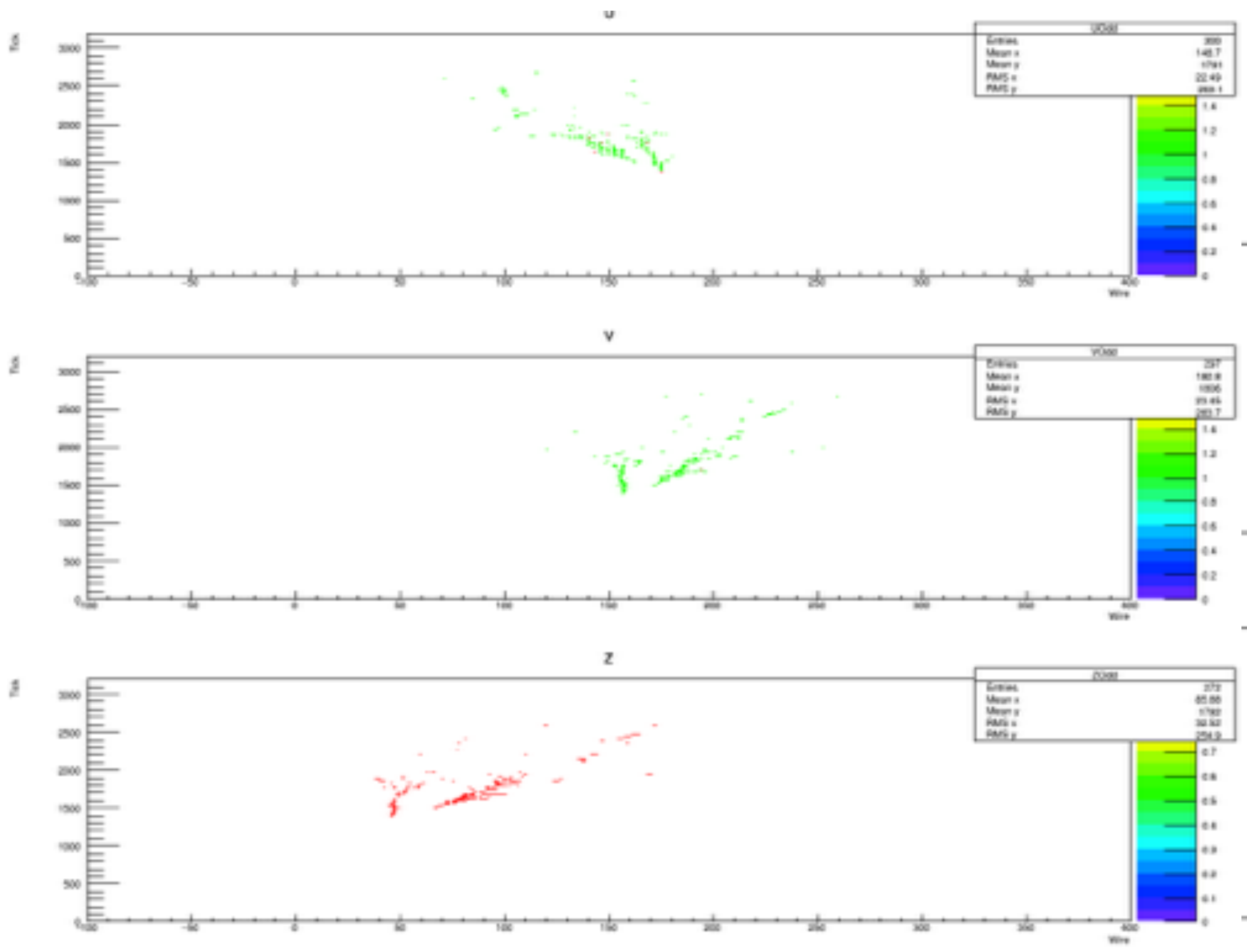
- Last week, showed my attempts at merging the fragmented tracks in 3D to make showers. Proved very difficult!

- Have this week gone back to 2D in the hope the cluster merging will be more straight forward!

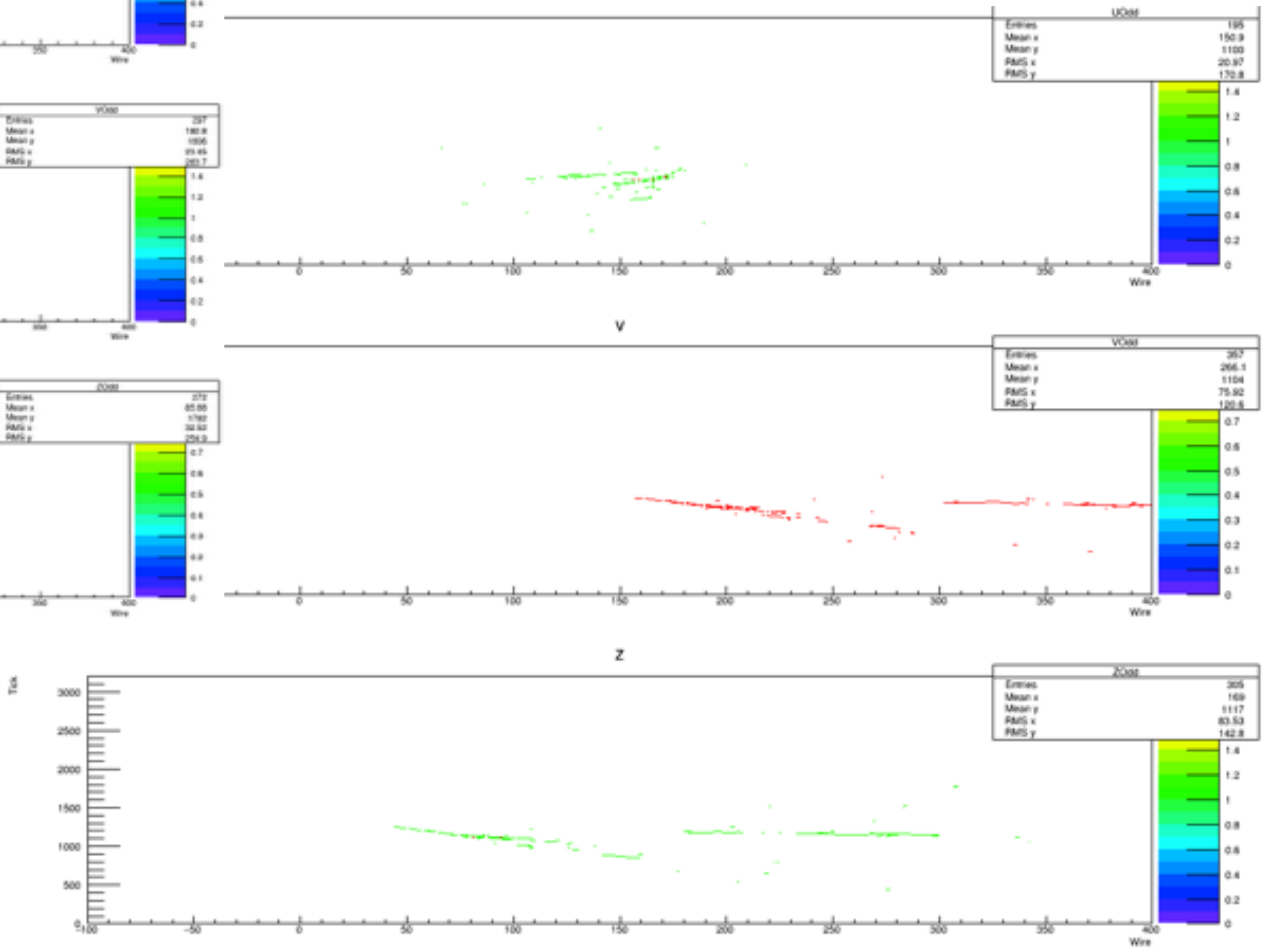
Global TPC Reconstruction

- Mentioned this last week.
- Rather than reconstructing in each TPC separately, which causes issues with showers being broken by the TPC boundaries, it makes sense conceptually to form a 'global' TPC and perform the reconstruction in this.
- A 'global wire' coordinate is found and the wire/tick space redefined for the entire cryostat.
  - Right now just looking at the large drift volume (odd numbered TPCs).
  - Reconstruction in this way has not been attempted before but has proved successful!

Example ‘Global TPC’



Colours mean nothing!



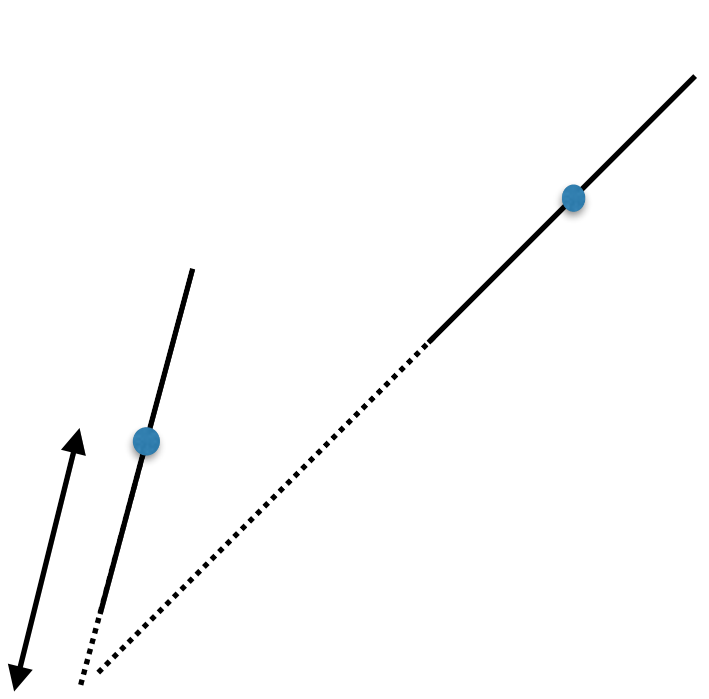
Can see showers that would have been split into multiple TPC and require recombination:

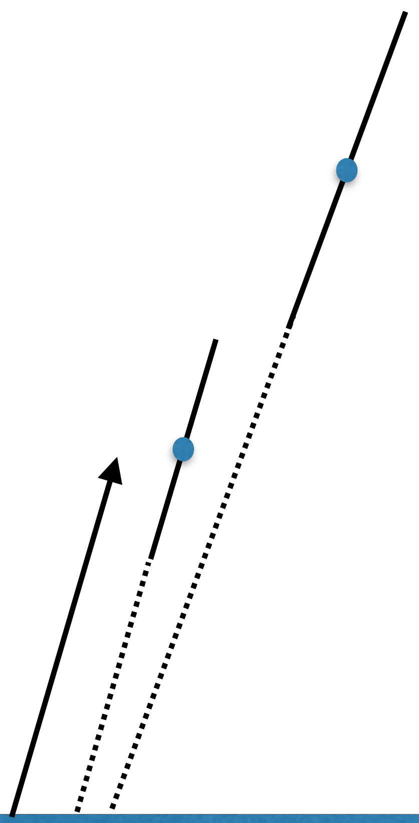
Improved 2D Merging

- With the reconstruction performed in the 'global TPC' space, the clustering merging can also follow from this.
- Same problem we had in the split TPCs; long showers are fragmented and reconstructed as separate clusters.
- I rewrote the algorithm I wrote for the separate TPC recon and have been trying to improve it…
- Found some discriminating variables to separate the incorrectly/correctly merged clusters (last time just used 'goodness of PCA').

Angle & Crossing Distance

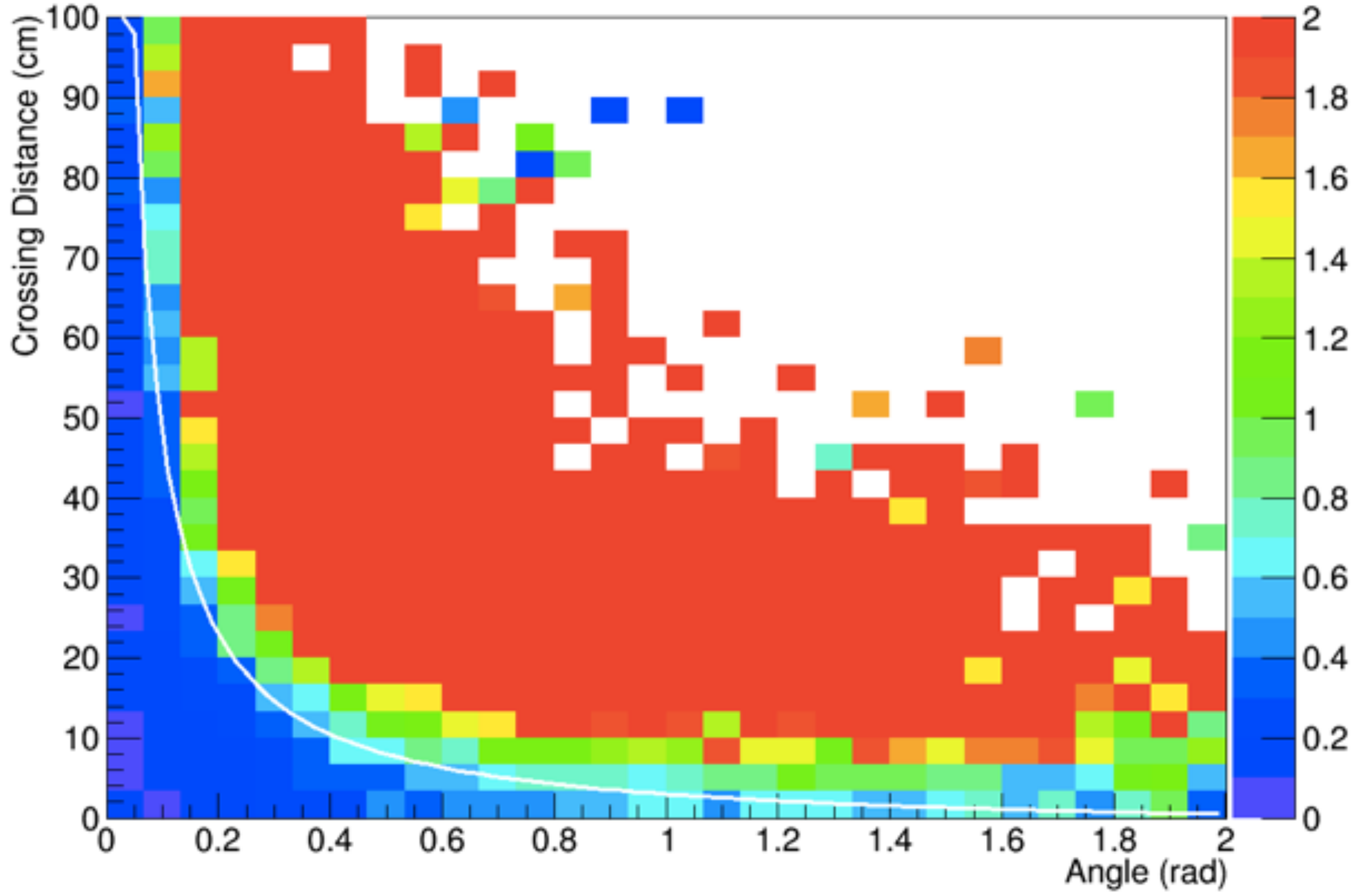
- Using a PCA, a charge weighted centre and many vector projections, a direction vector and start and end points of the cluster are roughly defined.
- These are used to find various quantities, including angle and ‘crossing distance’:





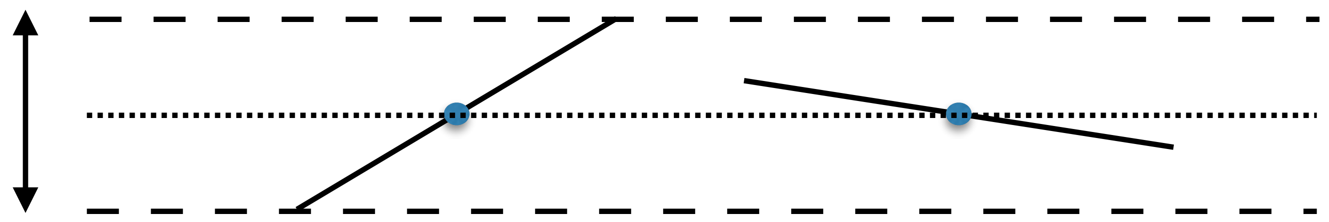
Angle & Crossing Distance

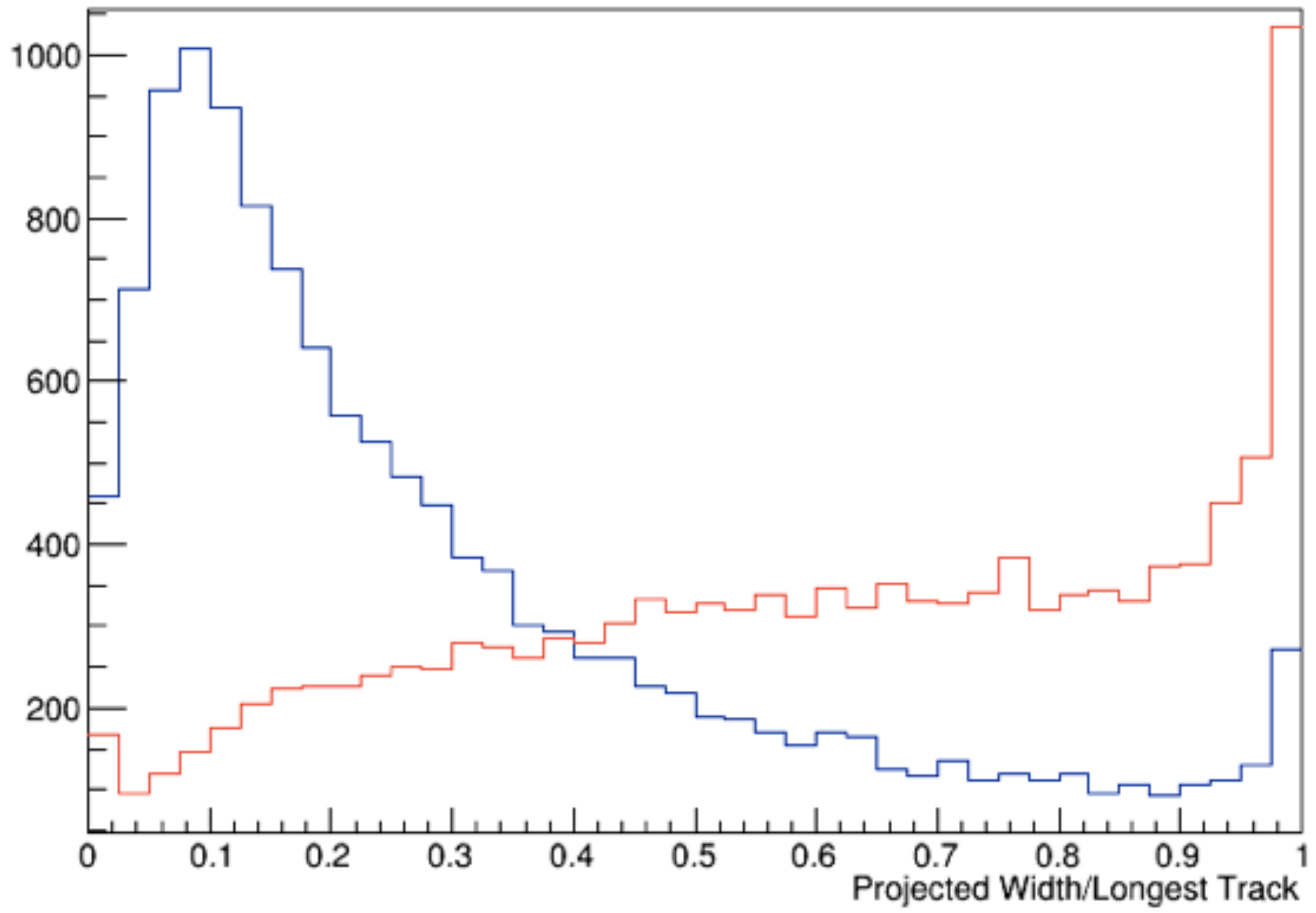
- Crossing Distance vs Angle (weighted by cluster size)
  - incorrect merge/correct merge (blue good, red bad!)
  - fit shows a tight cut (would rather lose merges than merge incorrectly)



Projected Width

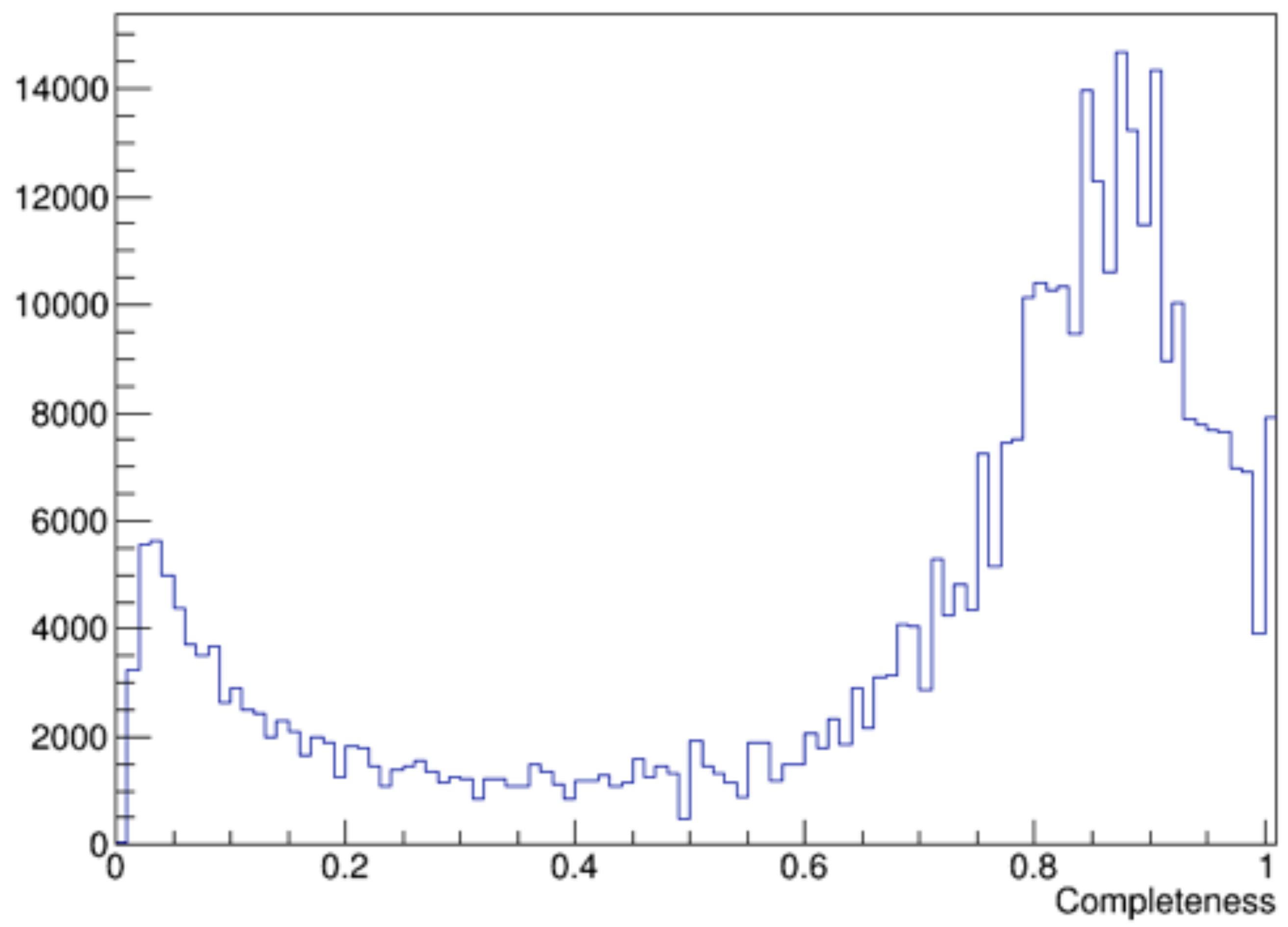
- Similar to the 'tube diameter' I showed last week but in 2D.
- Project width/ Longest cluster 'length':
- Red incorrect merge/ blur correct.

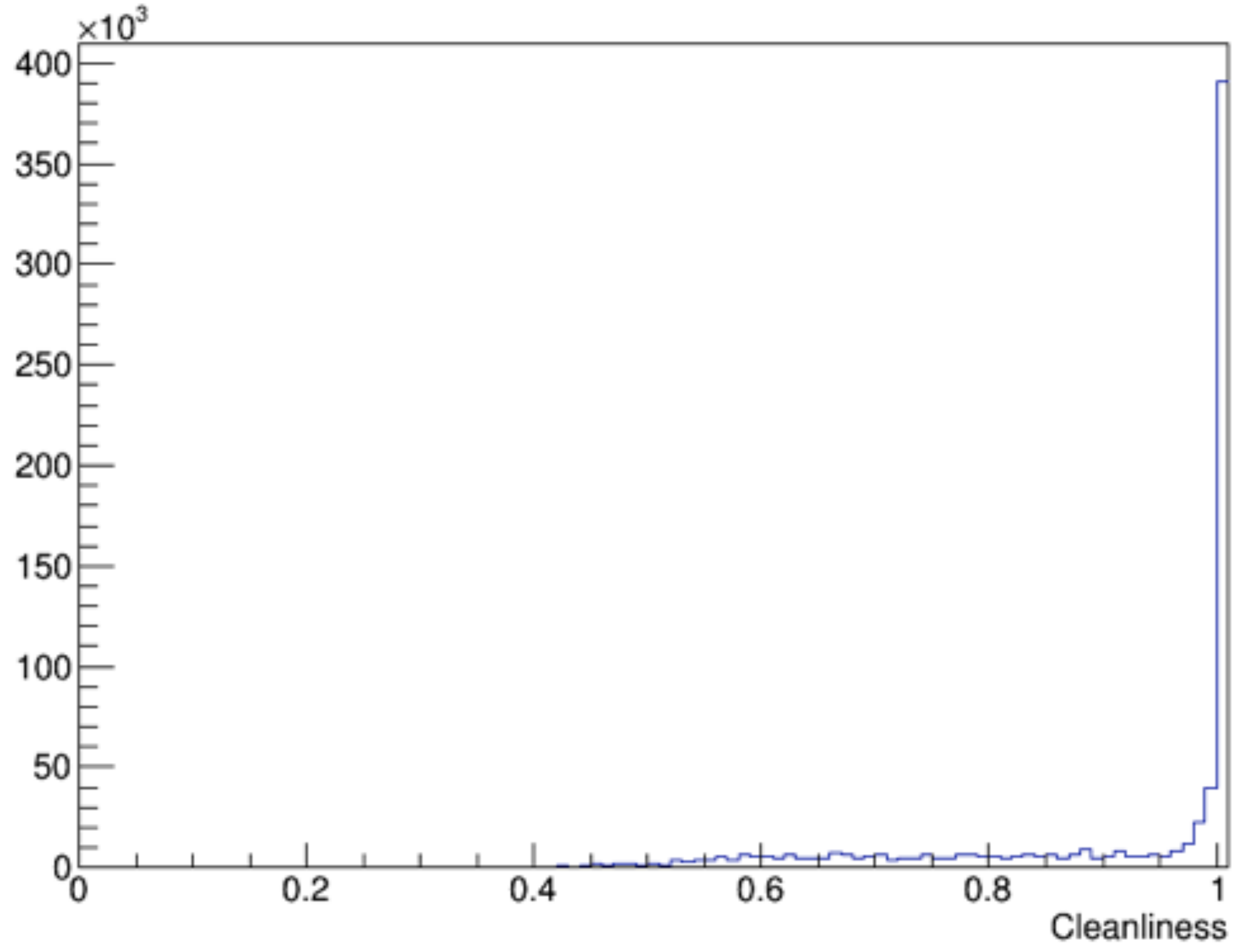




Improved Merging

- Used the same merging algorithm I wrote ~a month ago: takes the vector of clusters and forms pairs of possible merges. Merges if pass cuts. Runs recursively, considering this larger cluster next time (recalculating its properties).
- Added these cuts to the merging; it looks much better!





Computational Performance

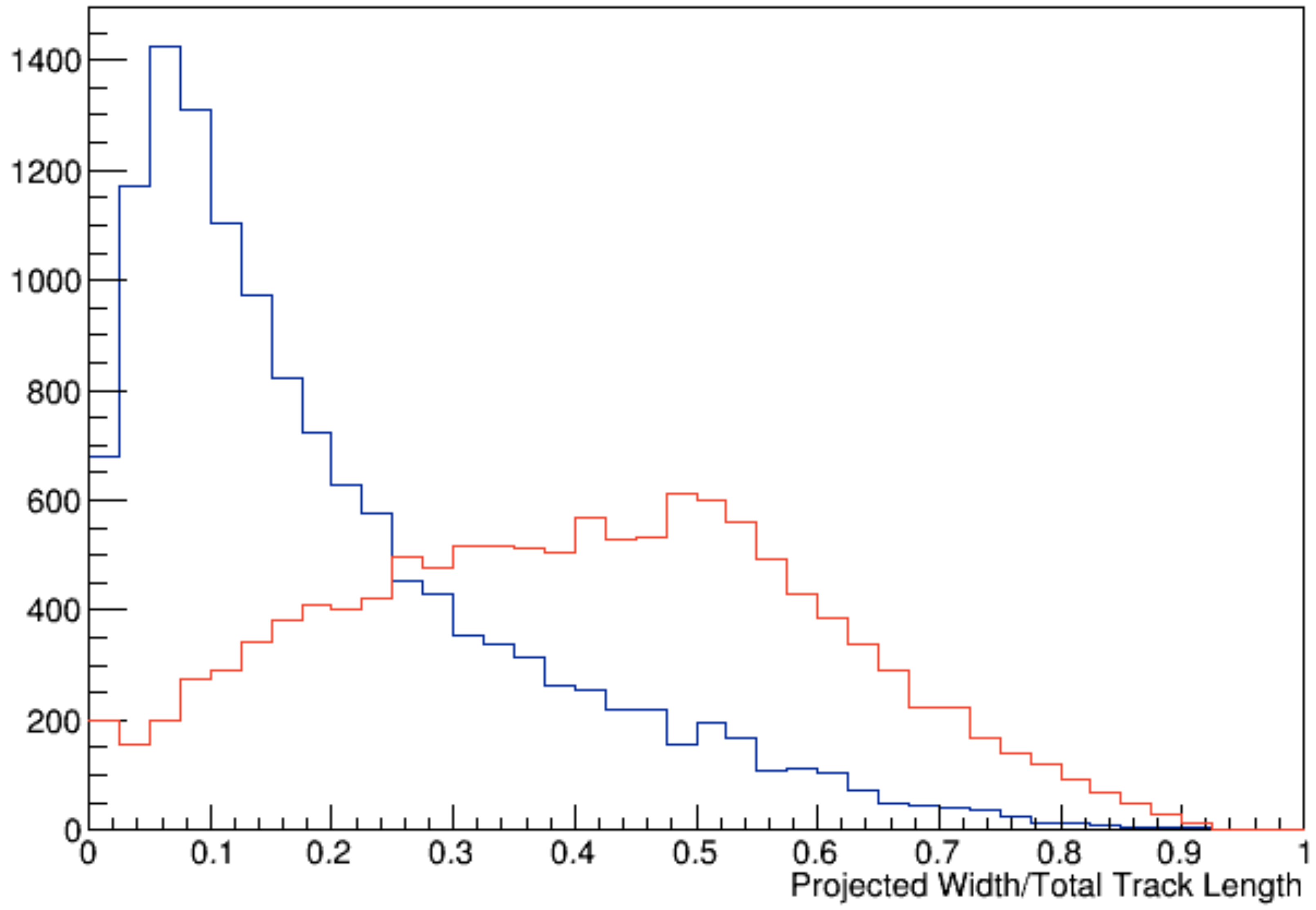
- Have been refining the clustering code to try to improve computational efficiency.
  - I haven't mentioned up to now the speed issue!
  - Was very slow… ~4 — 5 s per event :(.
- I've gone through the code and removed any redundant parts, reduced container sizes and changed everything from being passed by value!
- Now runs much, much faster! ~1.5 s per event (inc. merging) :).
- I have a few other things which I know will increase performance much more so will look at implementing these in the very near future…

Summary

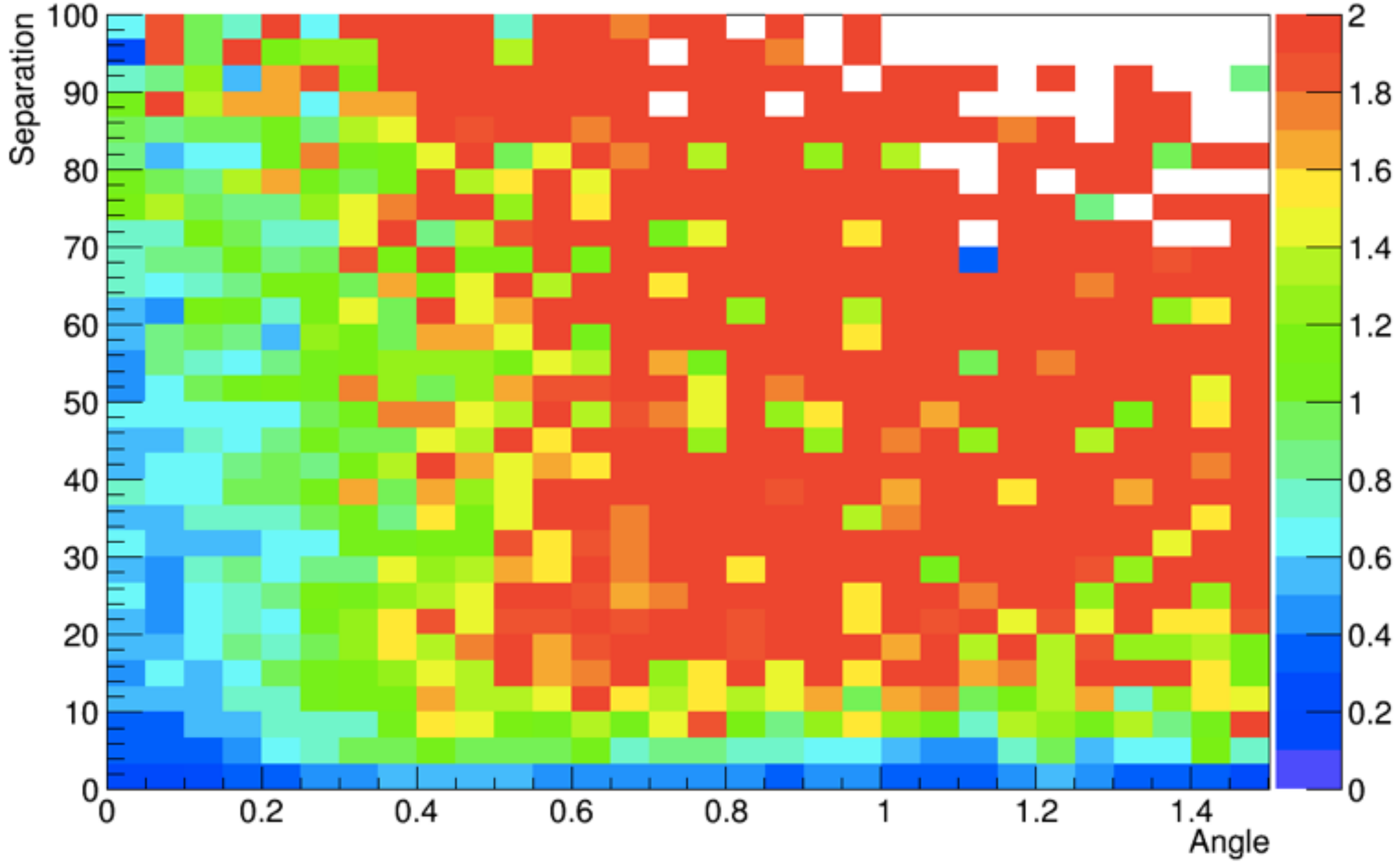
- Gone back to 2D reconstruction this week to try to improve things here.
- Now perform the reconstruction in a large 'global' TPC.
- Reconsidered the merging algorithm and found a few good discriminators for better merging; the clusters look very nice and complete now!
- Still further work to be done, but needed this in time for the MCC so improvements will be made after!
- Will go back to showering stuff now with R&D.

Couple of other plots...

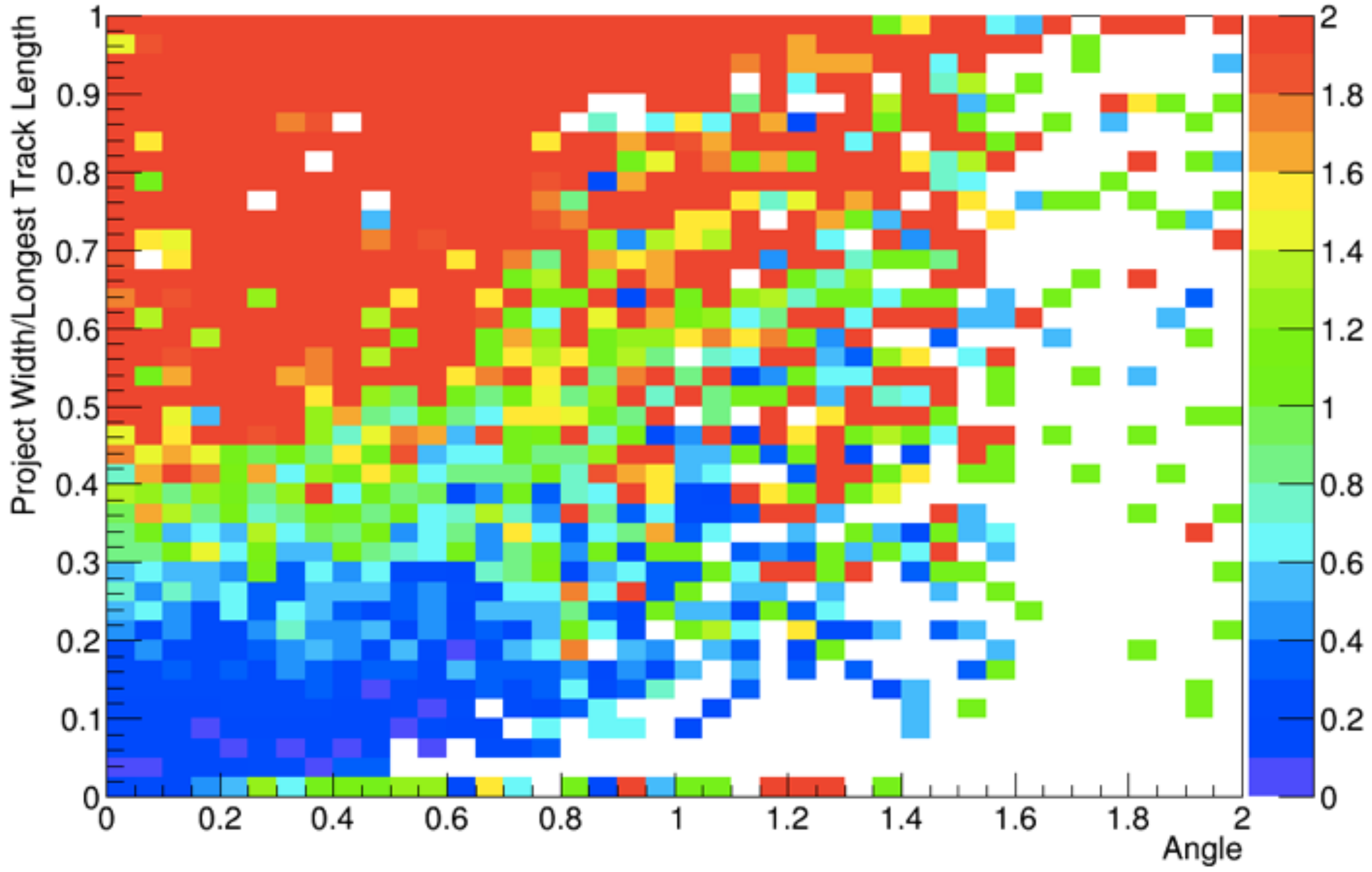
Width/Total Length



Angle/Separation



Angle/Projected Width


Project Width/Longest Track Length
Angle