Vertexing Algorithms with the ATLAS Detector for the HL-LHC Upgrade

Ian Lim, Ben Nachman, Simone Pagan Griso,
and Maurice Garcia-Sciveres

Lawrence Berkeley National Laboratory

Monday, April 16, 2018

BERKELEY LAB

ATLAS
EXPERIMENT

How well do existing vertexing algorithms perform at large mu?

And how can they be improved?

n.b. mu is the number of simultaneous proton-proton collisions per bunch crossing.

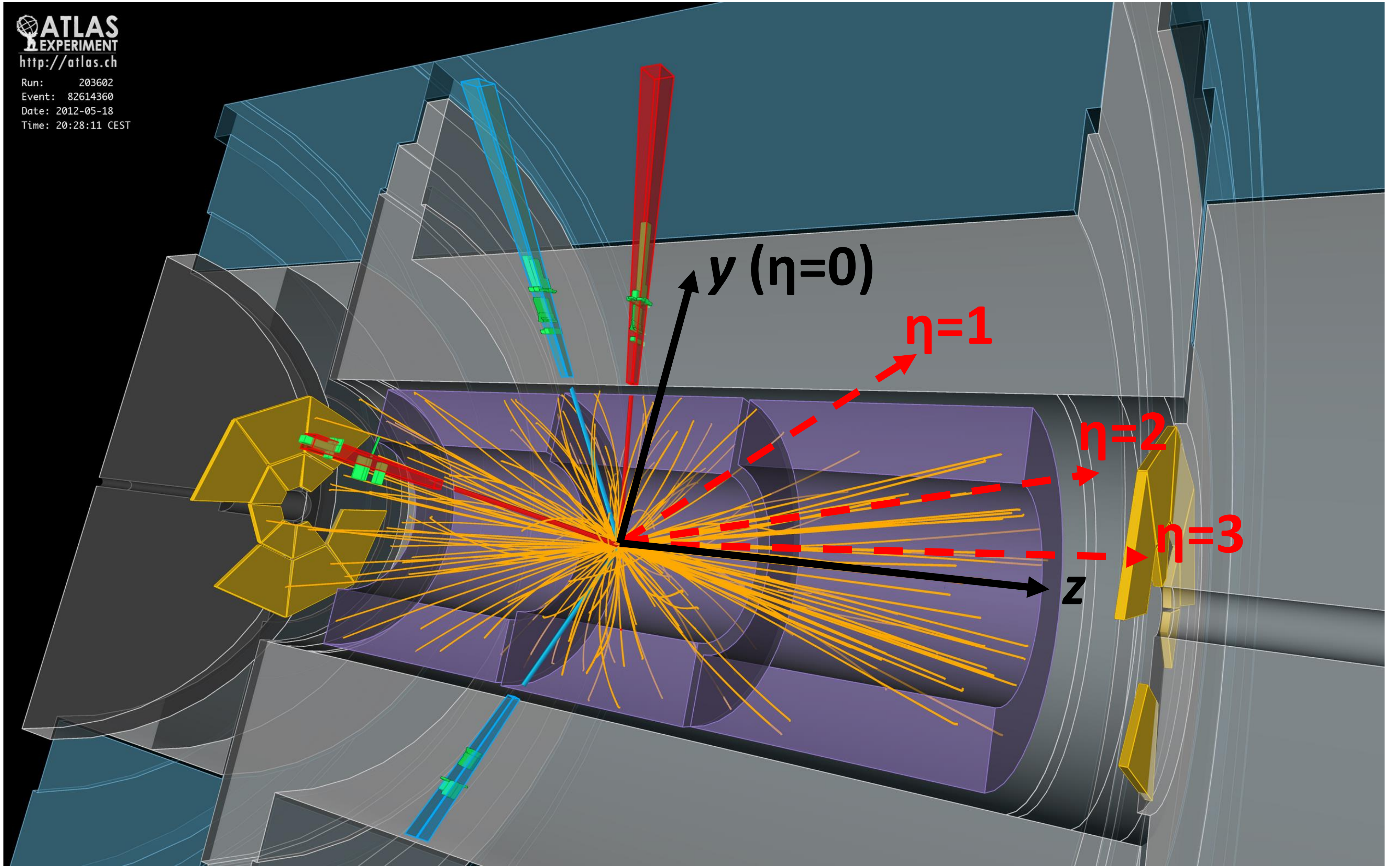
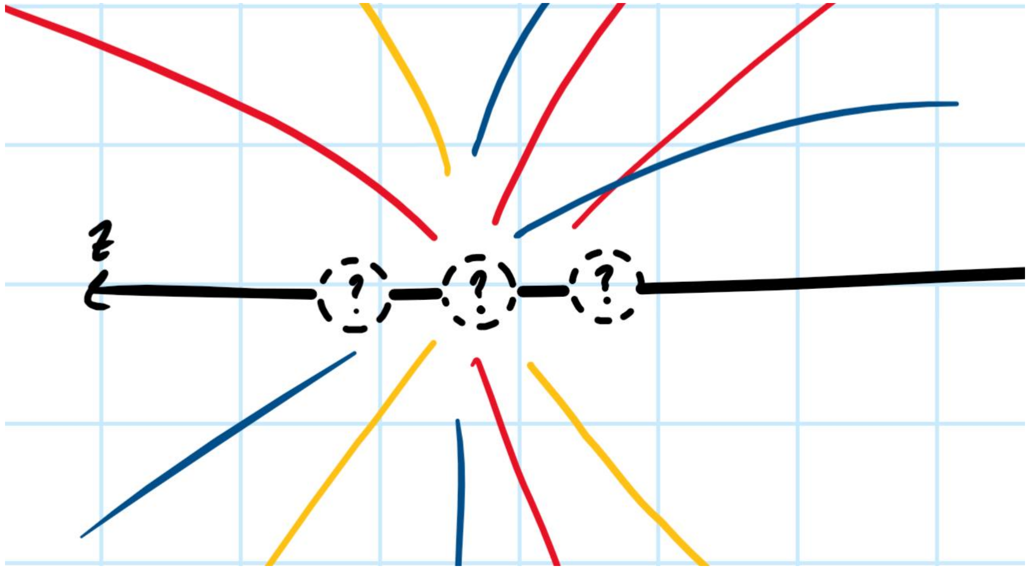
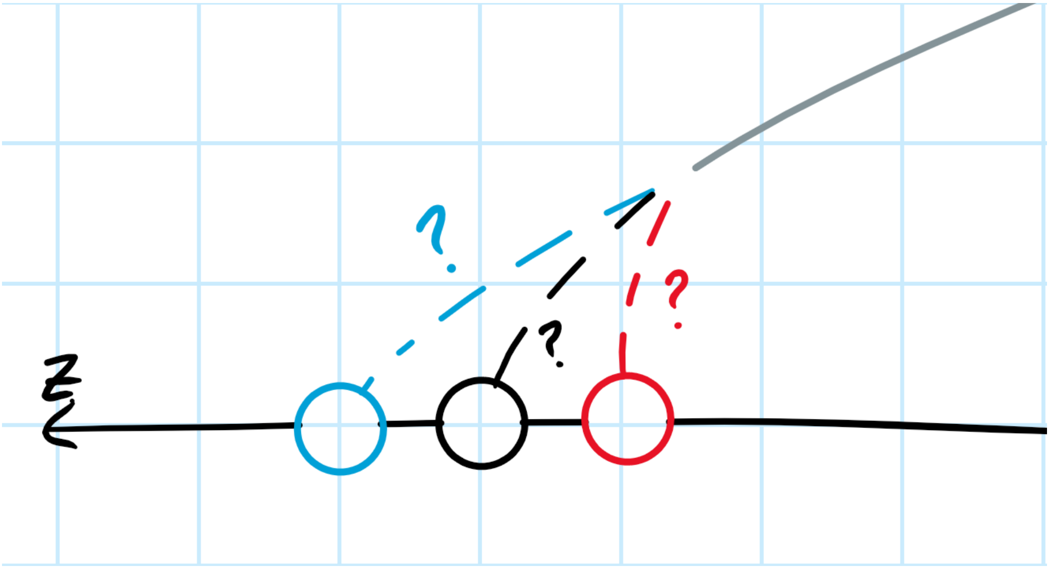


Image credit: http://atlasexperiment.org/photos/events-collision-proton.html

What is vertexing?

- Primary vertices are locations of proton-proton collisions in the detector
- Two main goals– position reconstruction and track association
  - How well can we determine where a collision happened in space?
  - Given the tracks left in our detector by collision products, how well can we associate them to the correct vertex?





Vertexing and the Upgrade

- At the HL-LHC, we expect $\mu \sim 200$– a tenfold increase!
- With increased vertex density, performing a clean reconstruction becomes significantly harder.
  - Hard scatter is obscured by 10x more pile-up
  - More tracks to assign
  - Greater likelihood of merging
- Two vertexing algorithms: Iterative and Adaptive Multi-Vertex Fitter (AMVF). In AMVF (compared to iterative):
  - Greater number of vertices reconstructed
  - Improved spatial resolution between adjacent vertices
  - Track-vertex association somewhat worse

Iterative vs. Adaptive Fitter

- Iterative fitter:
  - Generates seeds one-by-one
  - Iteratively assigns weights to tracks and refits vertex position
  - All tracks incompatible by more than 7σ are removed from the fit
  - Repeat with remaining tracks until no more tracks are left
  - Seeding runtime is quadratic in mu
- Adaptive Multi-Vertex Fitter (AMVF):
  - Generates all seeds simultaneously by imaging process
  - Vertex candidates compete for tracks
  - Seeding runtime approx. constant in mu (depends on bin size)

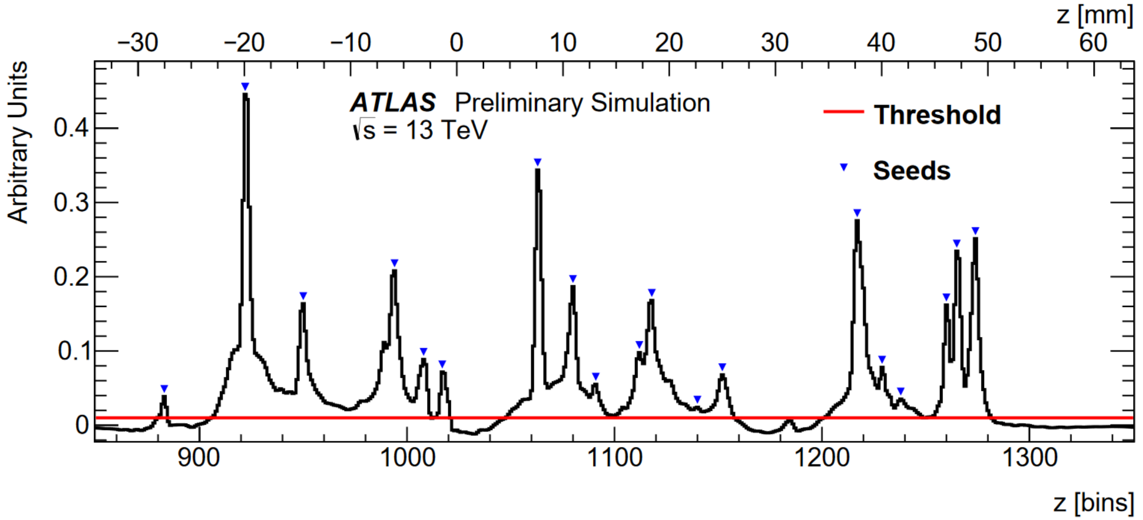
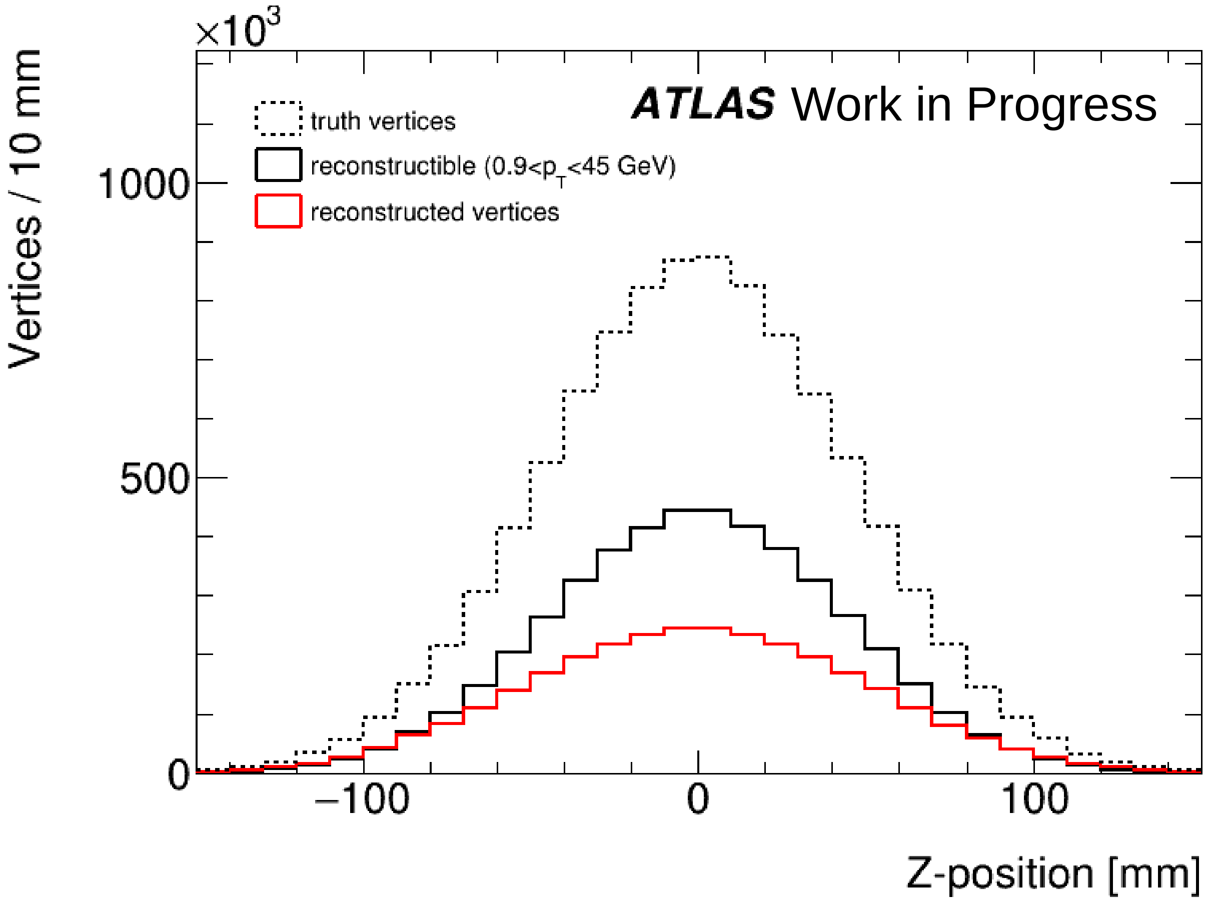


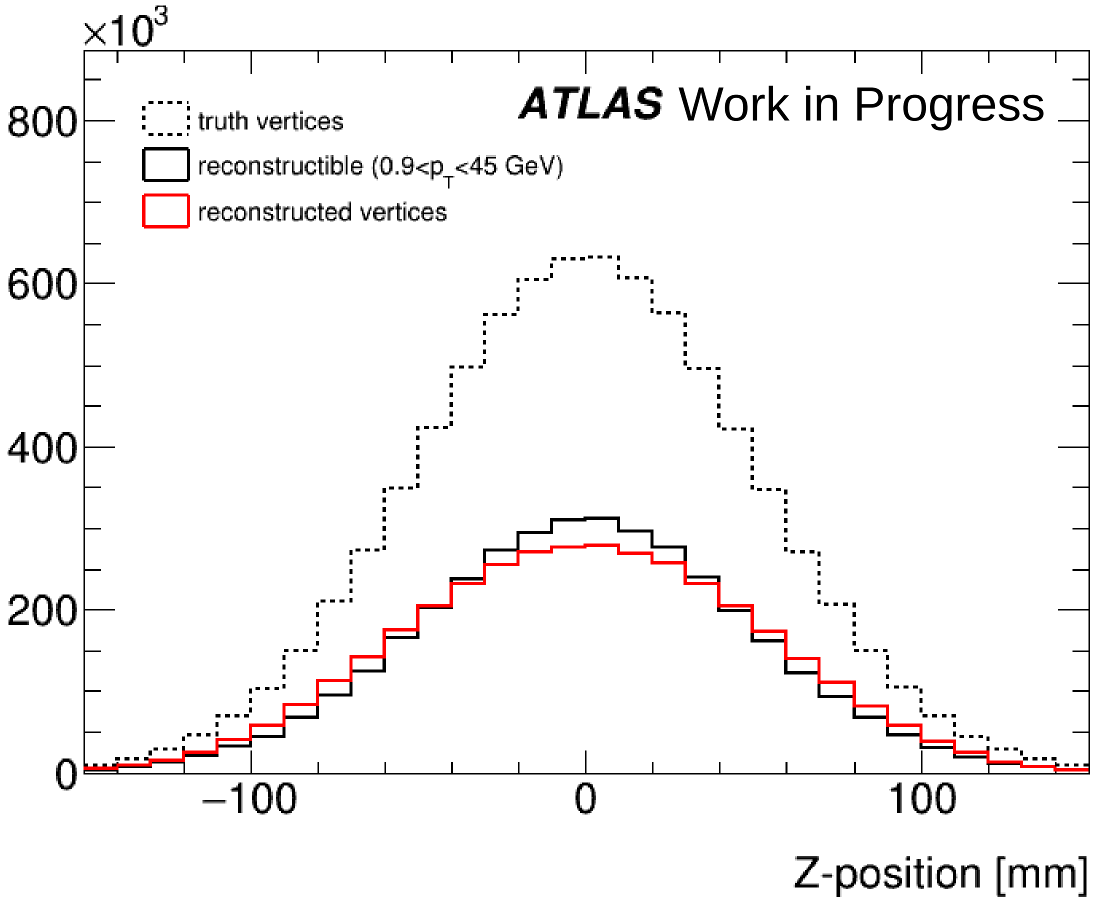
Image credit: doi:10.1088/1742-6596/664/7/072041

Reco vertices vs. reconstructible

Iterative



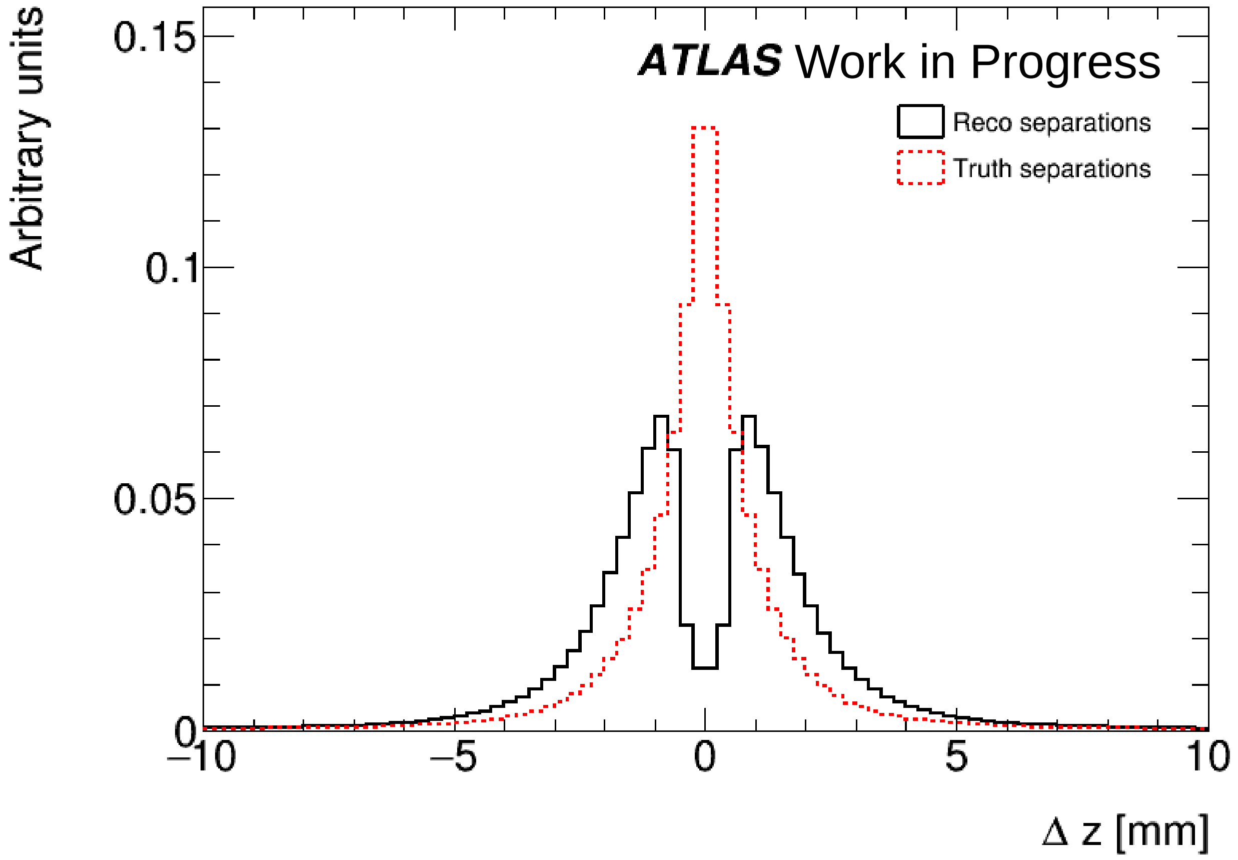
AMVF



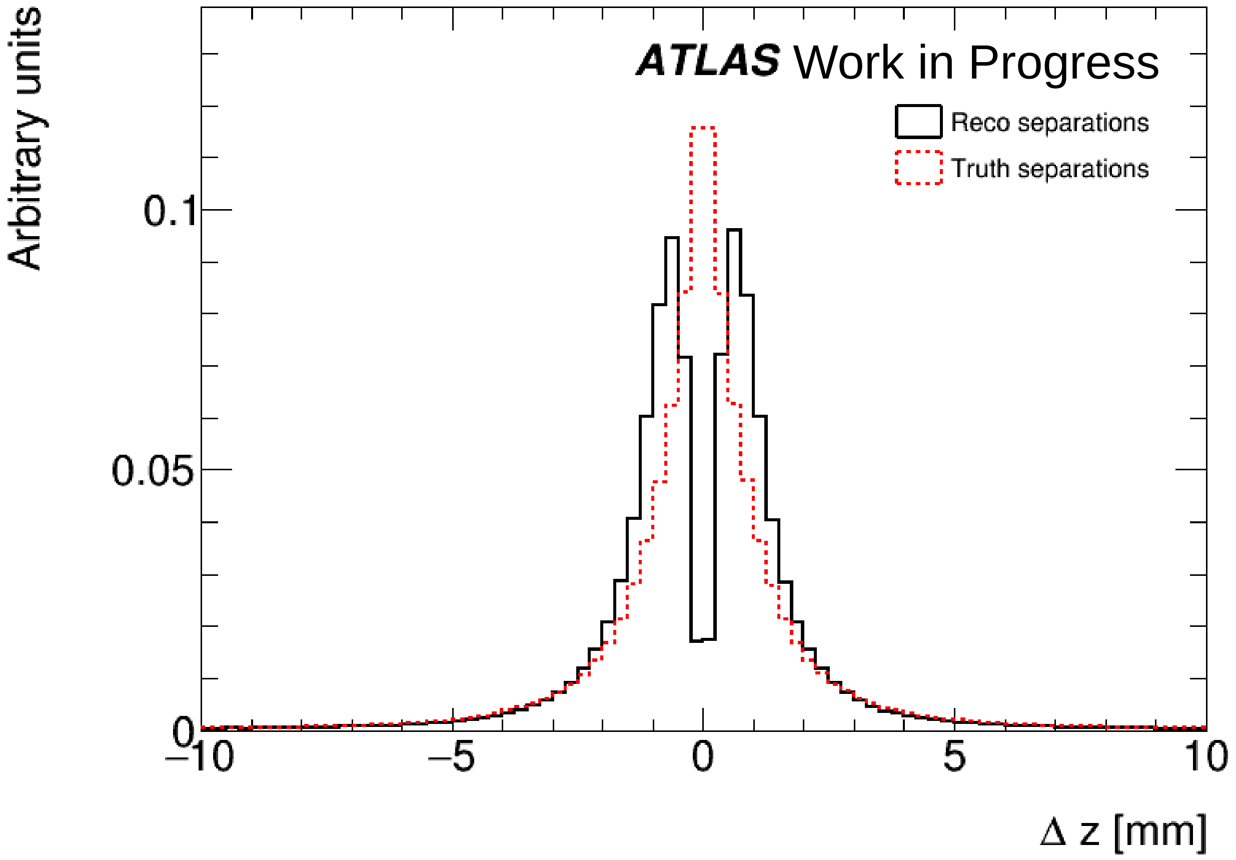
~30 more reco vertices per event in AMVF (increase from 60→90) and more accurate spatial distribution, but more recos are split (have tracks from multiple truth interactions)

Vertex resolution

Iterative



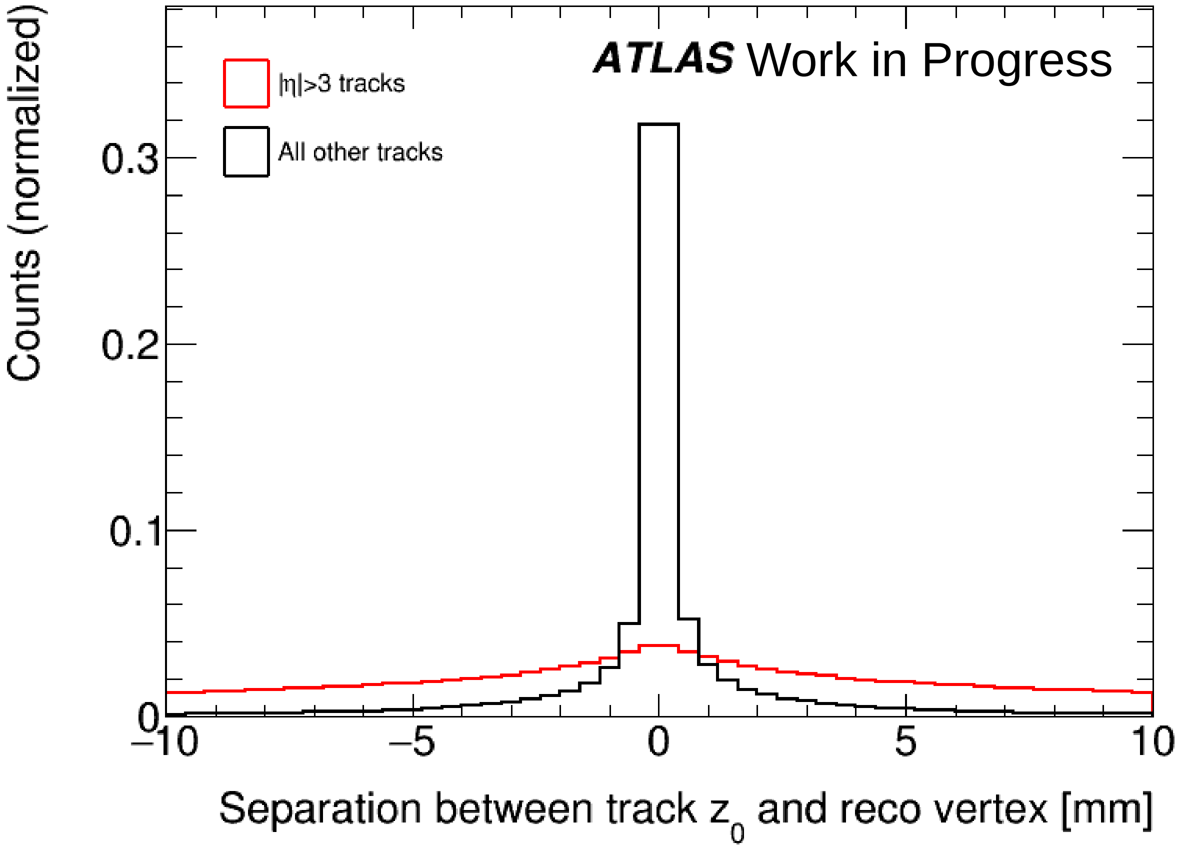
AMVF



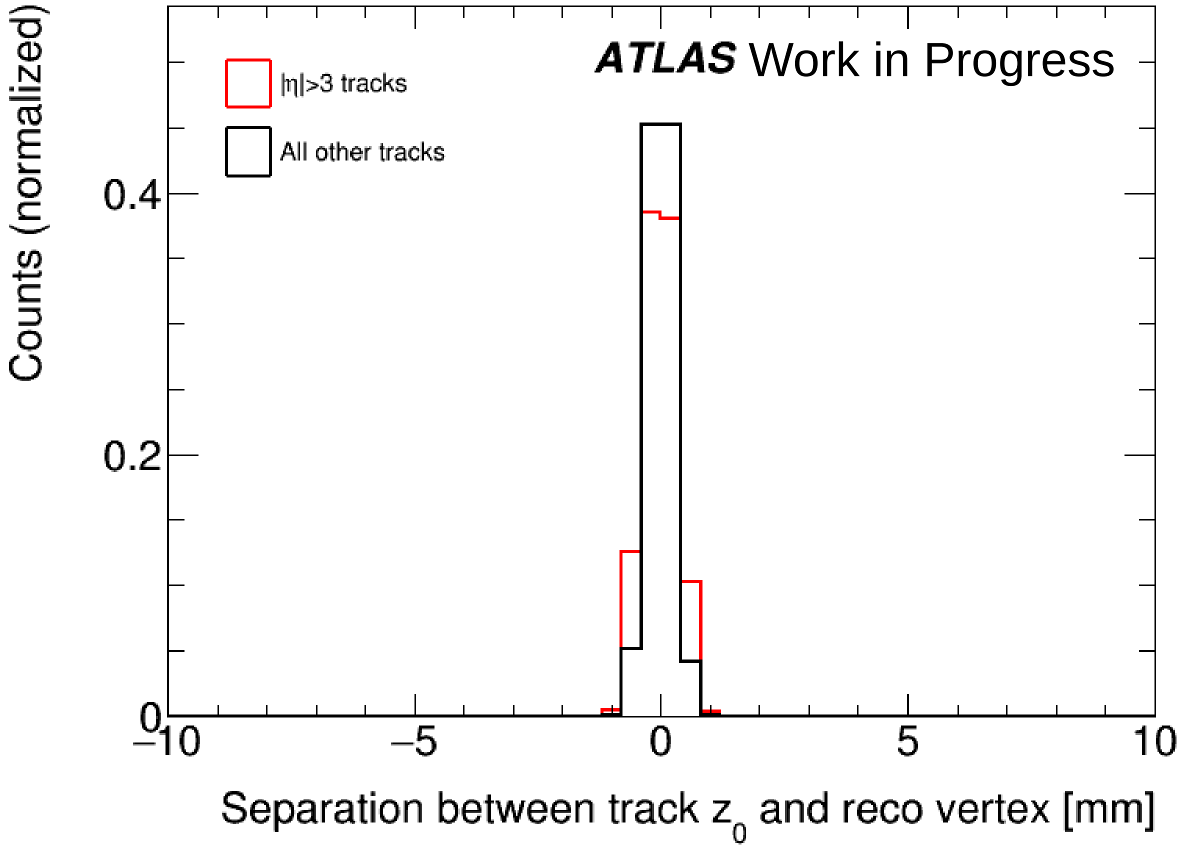
The change in the size of the central dip for the z-separations between neighboring reco vertices indicates improved z-resolution in AMVF.

Track – reco spreads

Iterative

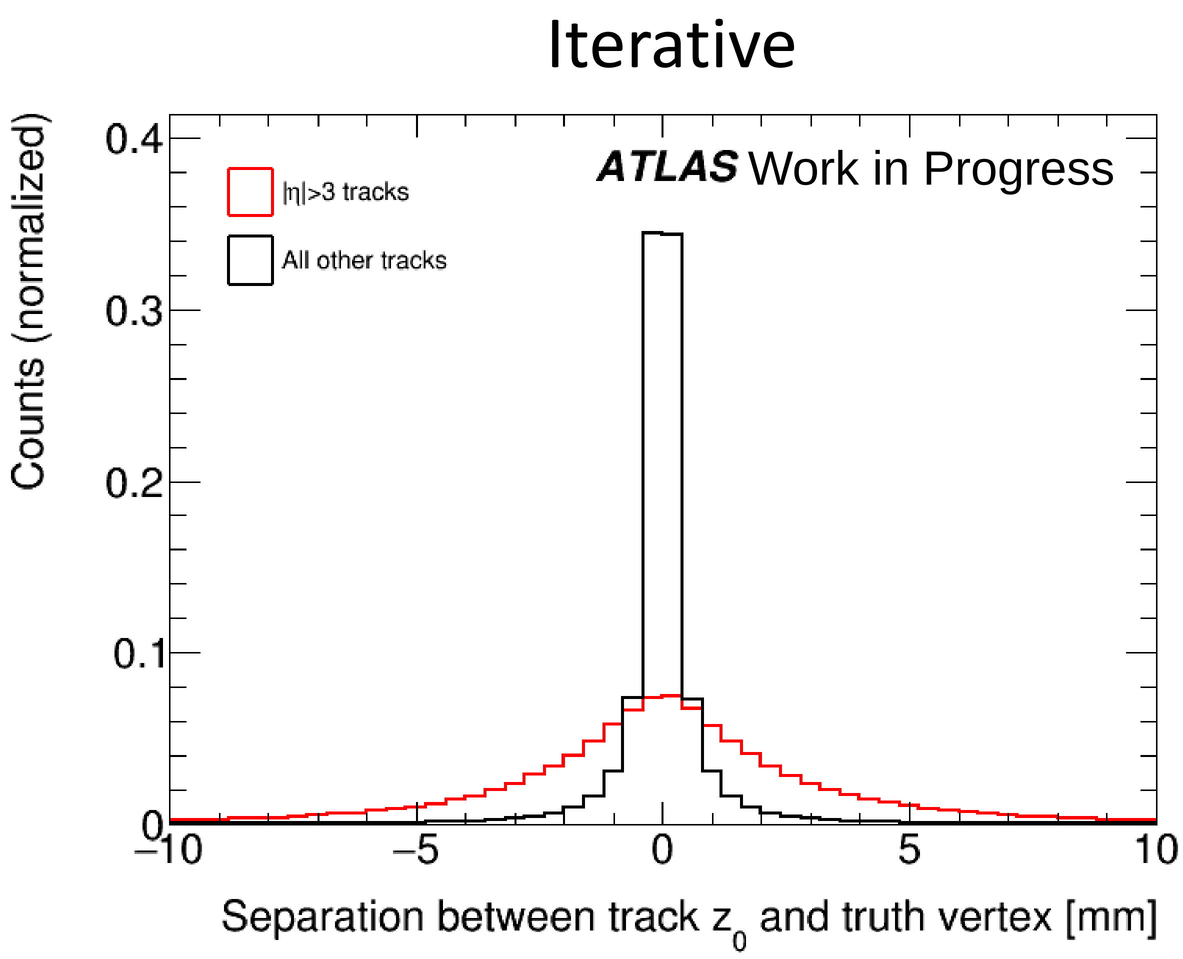


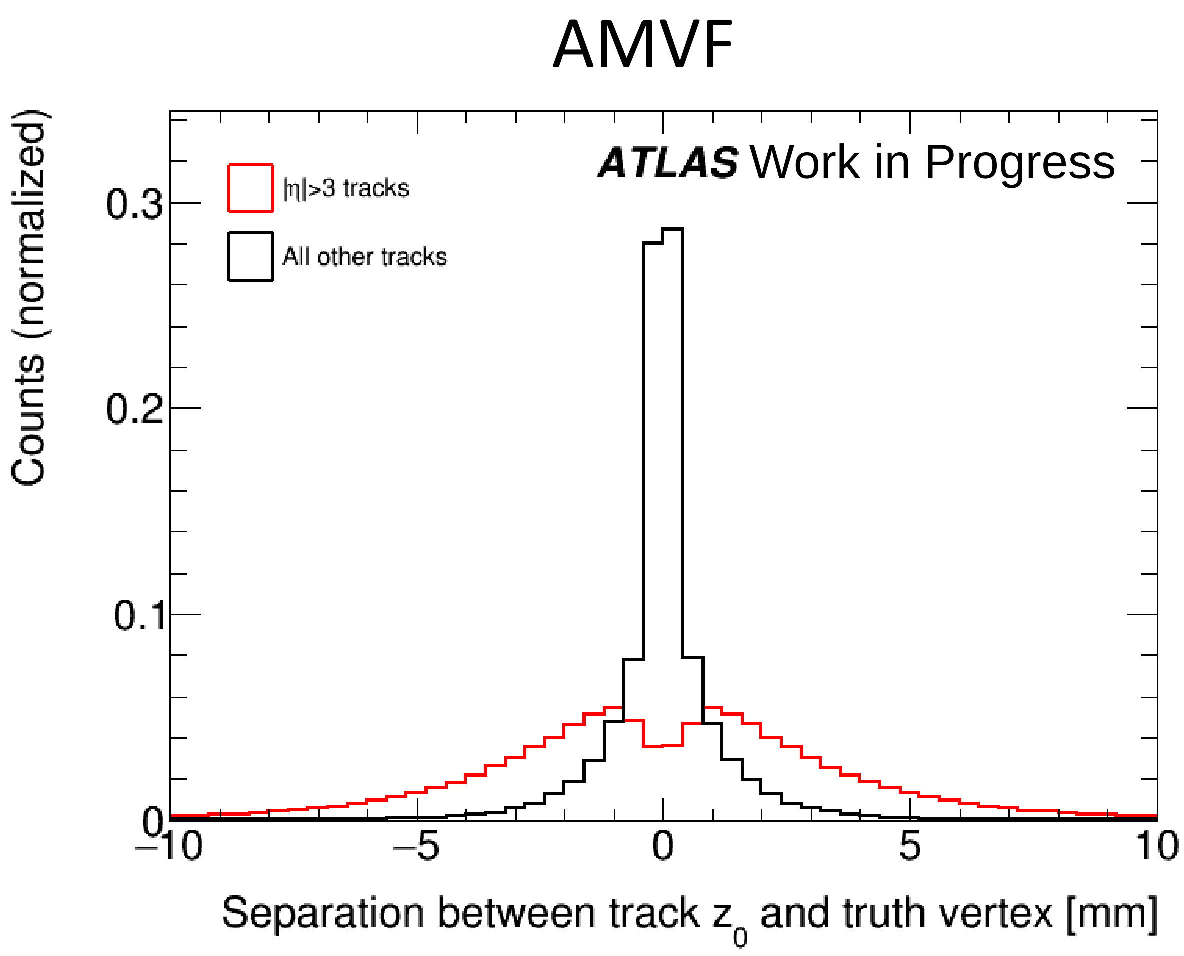
AMVF



In the AMVF, there is a hard cut on track-reco separation (< 1 mm), so tracks cannot be assigned to recos too far away.

Track – truth spreads (mismatched tracks)





The hard z_0 cut in AMVF ends up hurting track assignment for high-eta tracks, which are naturally spread farther from their truth vertices. The cut tracks end up contaminating otherwise clean reco vertices.

Conclusions

- Lots of changes due to the new vertexing code!
  - More seeds = better z-resolution but worse track-vertex association
  - Overall more split vertices (one truth, multiple reco)
  - Hard z_0 cut on track-reco especially hurts high-eta tracks because they are generally farther from the truth vertex
- Due to the pileup-independent runtime and improved z-resolution, the AMVF seems to be the future of vertexing.
- More studies are being conducted on the AMVF to understand the costs of these changes in terms of vertex splitting and TVA.

Backup

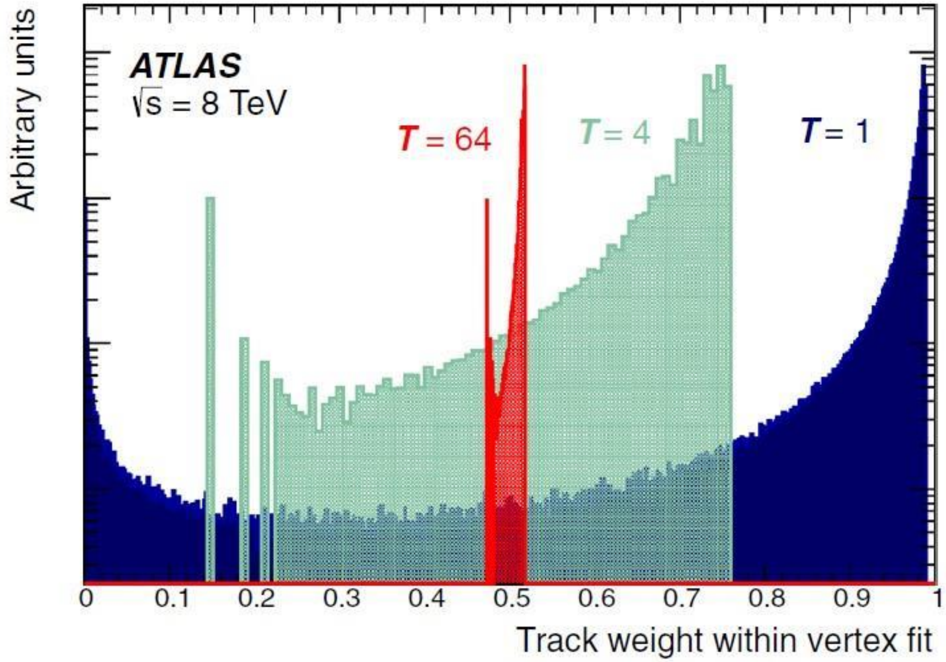
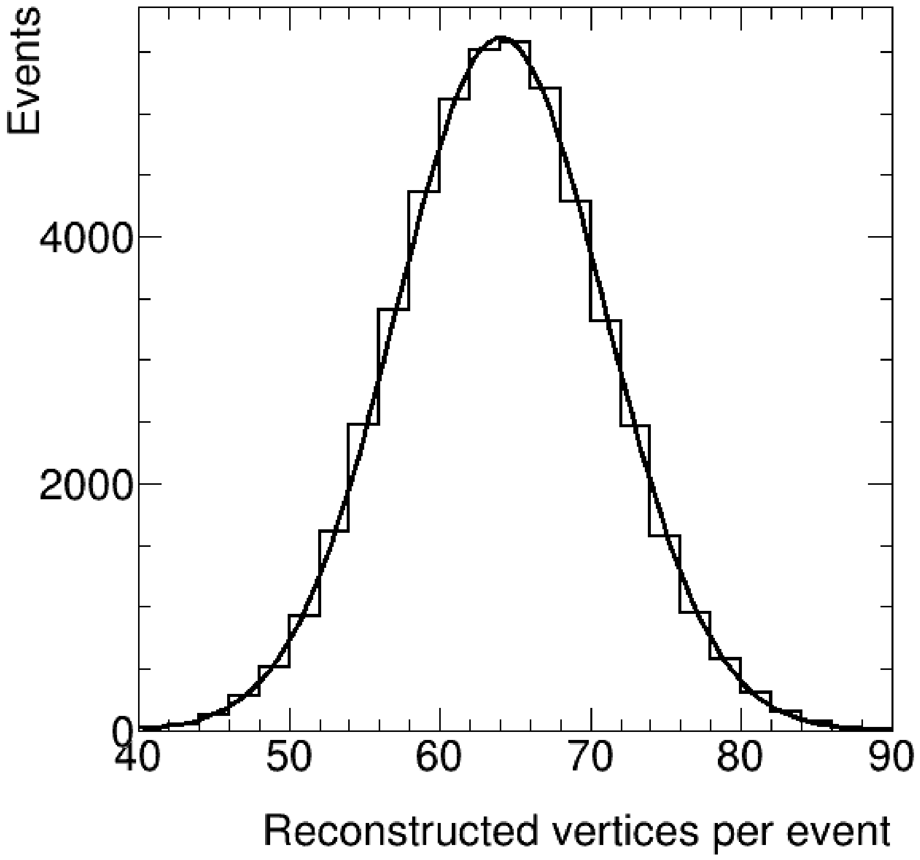


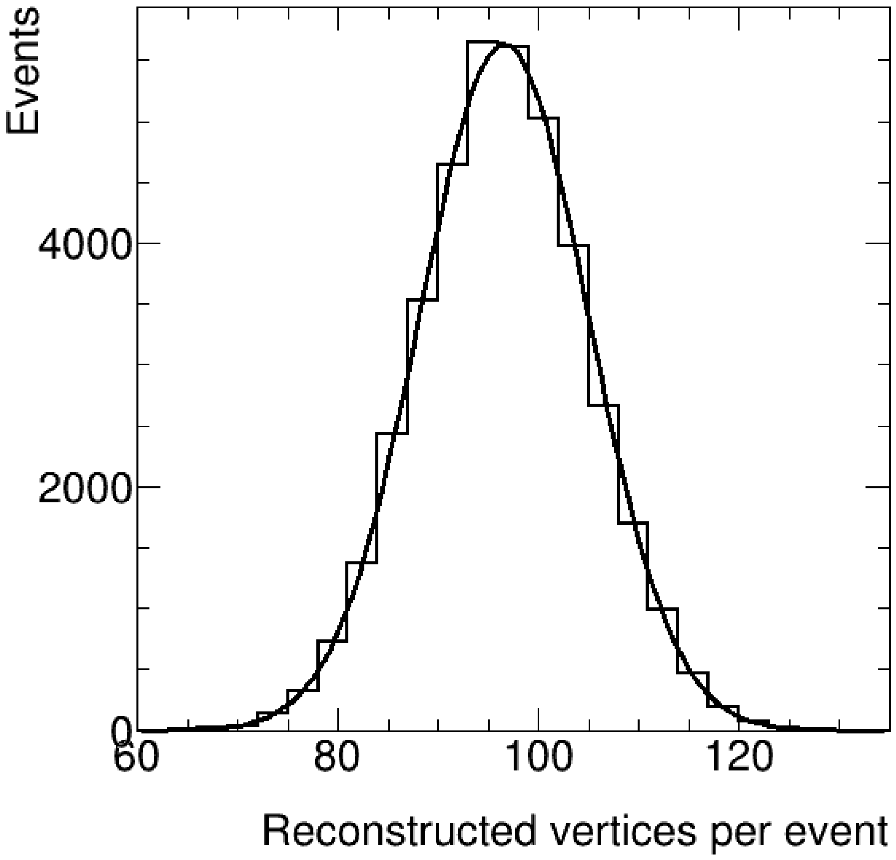
Fig. 3 Histogram showing the weights applied to tracks in the vertex reconstruction fit. The fitting algorithm iterates through progressively smaller values of the temperature T, effectively down-weighting outlying tracks in the vertex fit. The vertical axis is on a logarithmic scale

Reconstructed (reco) vertices per event

Iterative

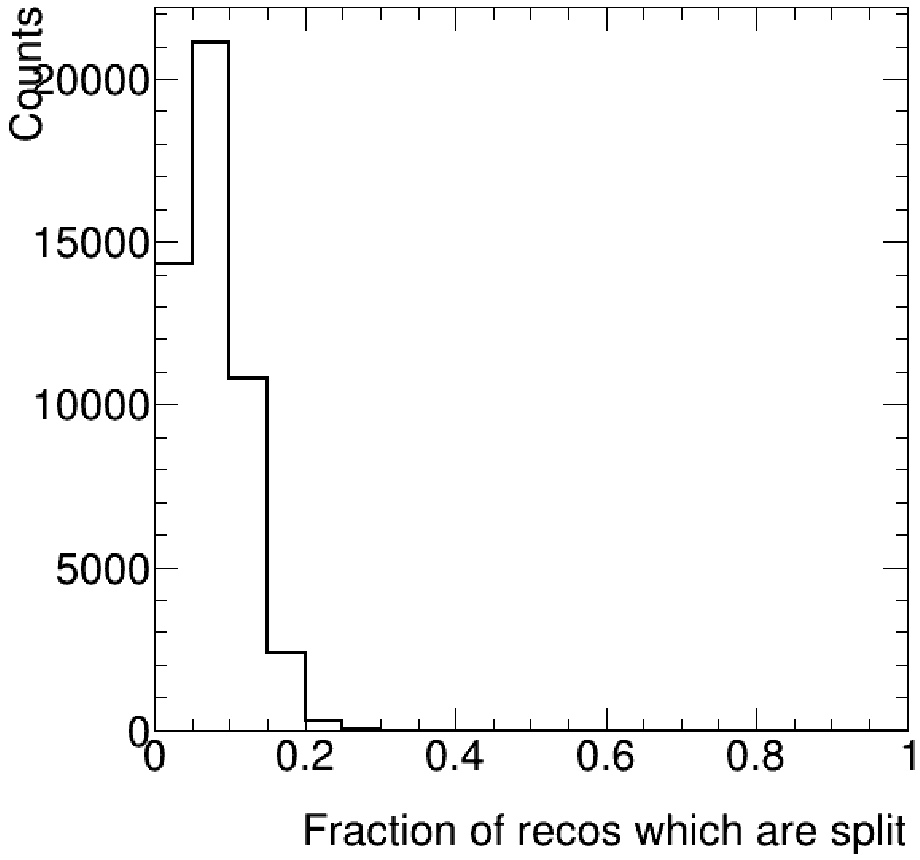


AMVF

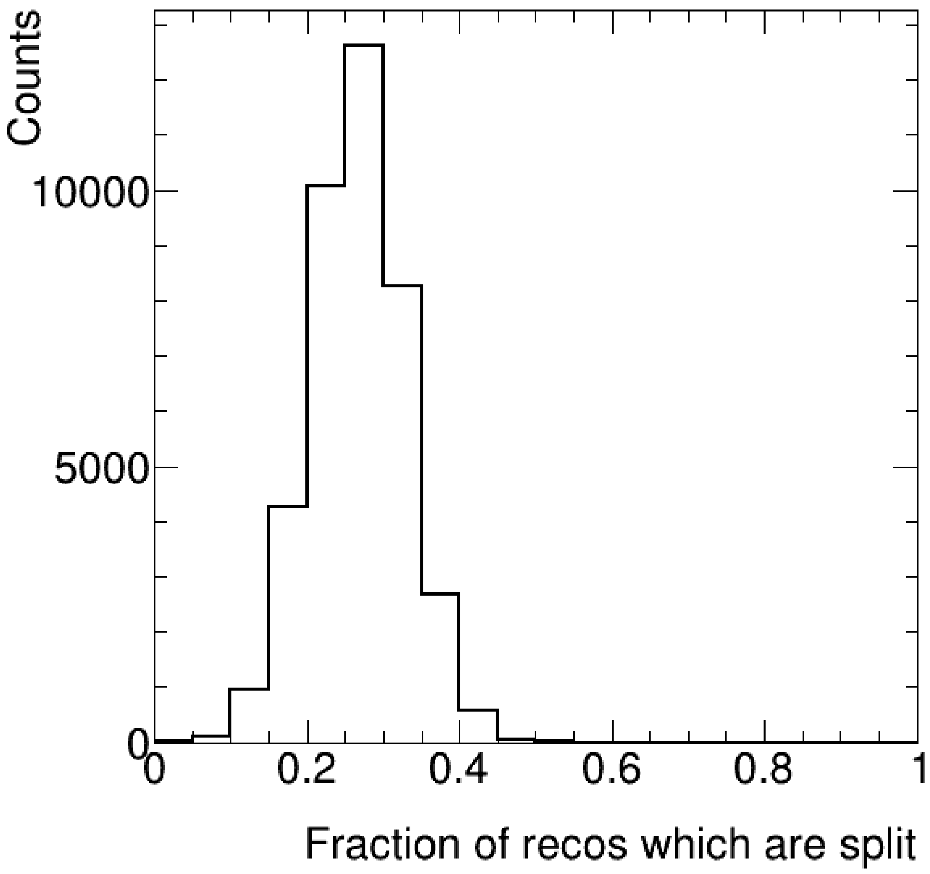


Split vertices

Iterative

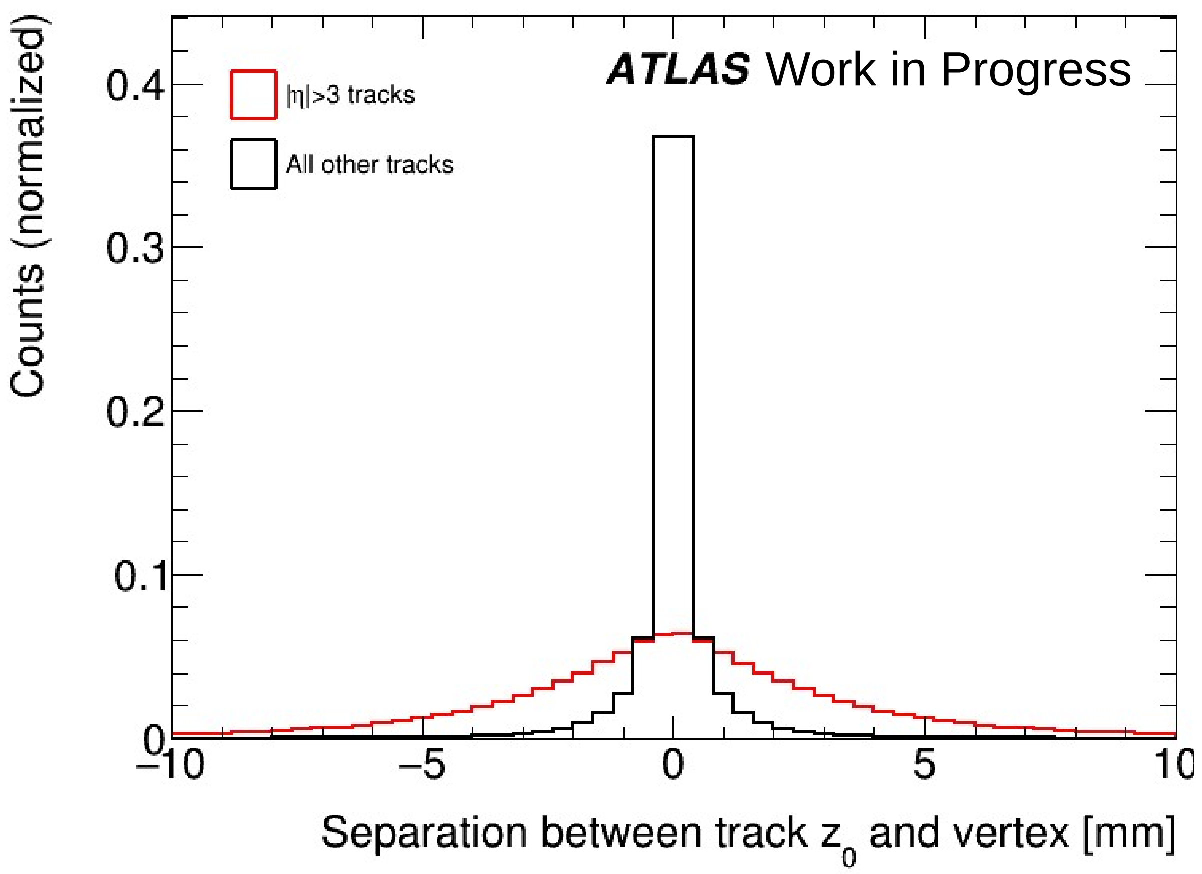


AMVF

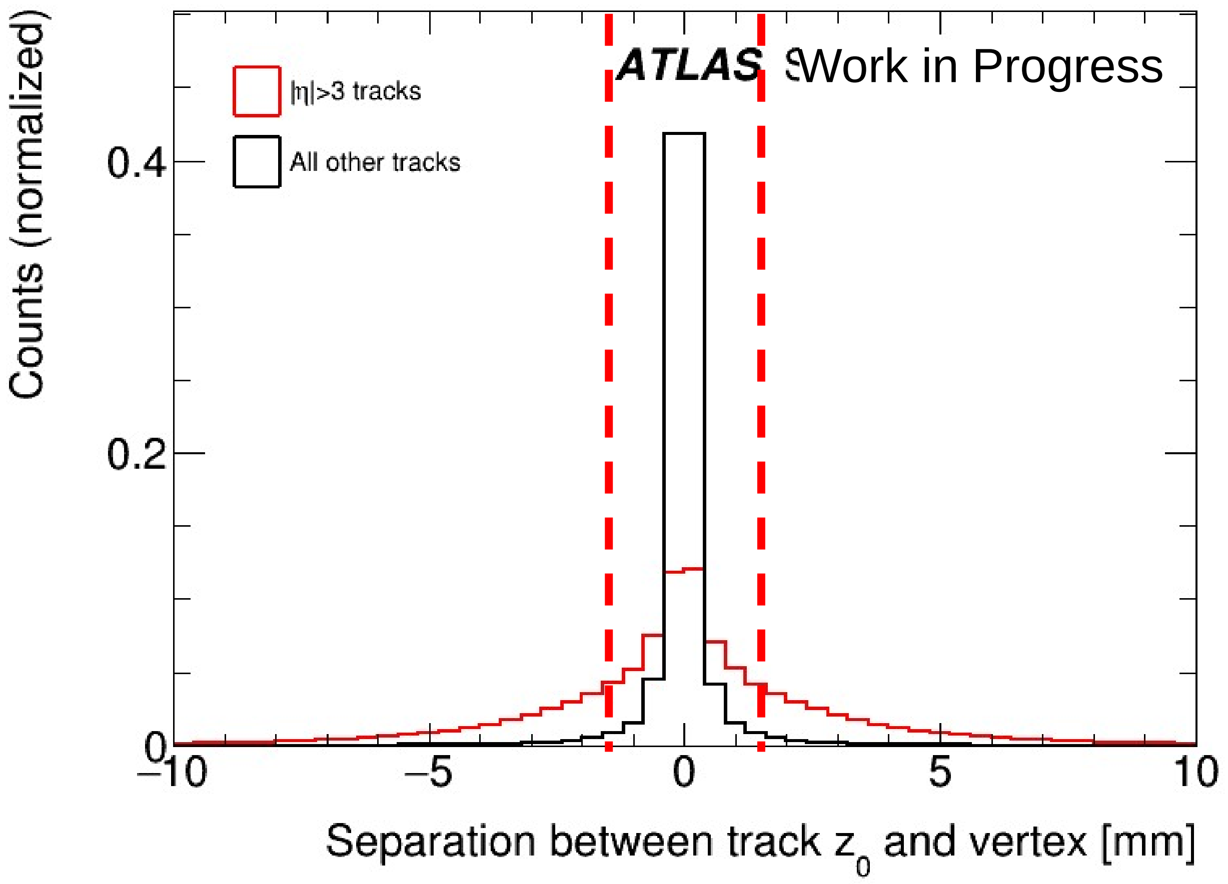


Track – truth spreads (all tracks)

Iterative

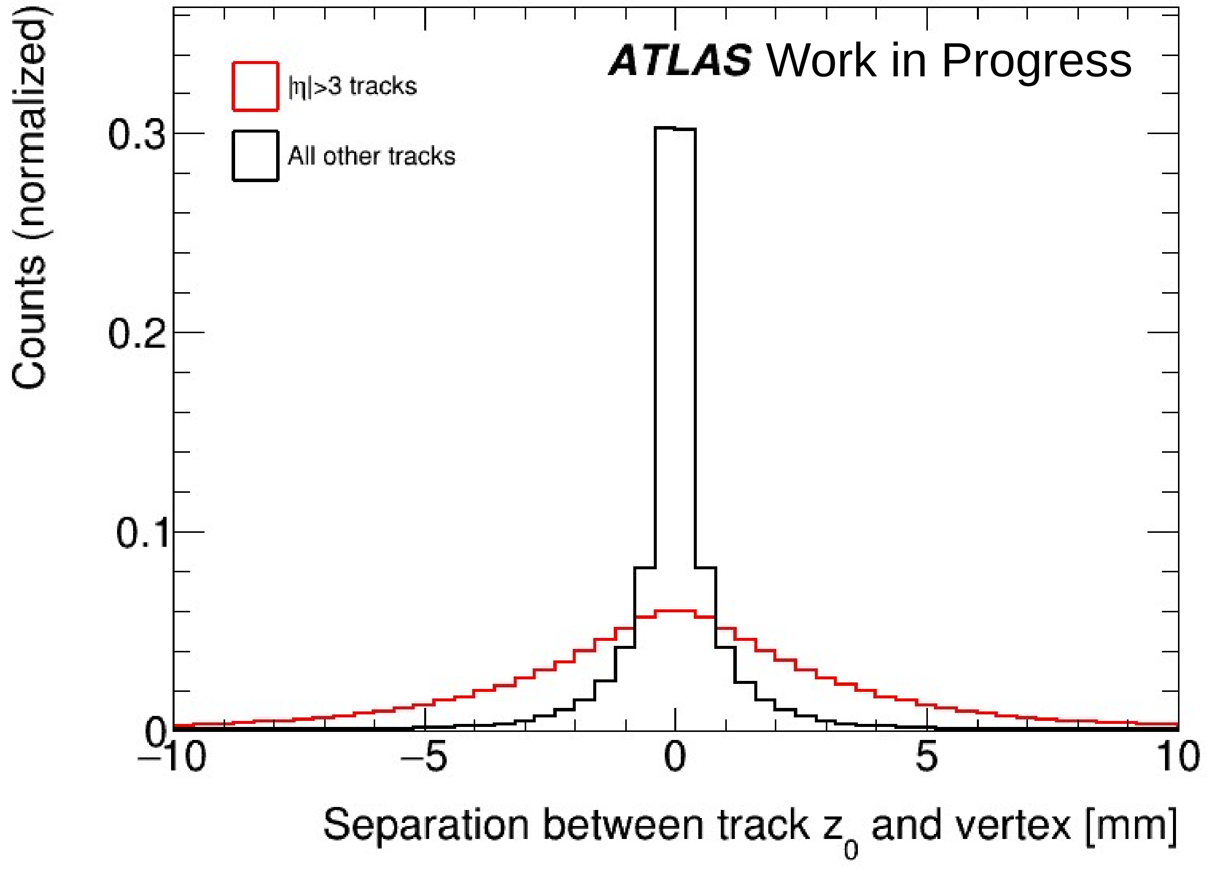


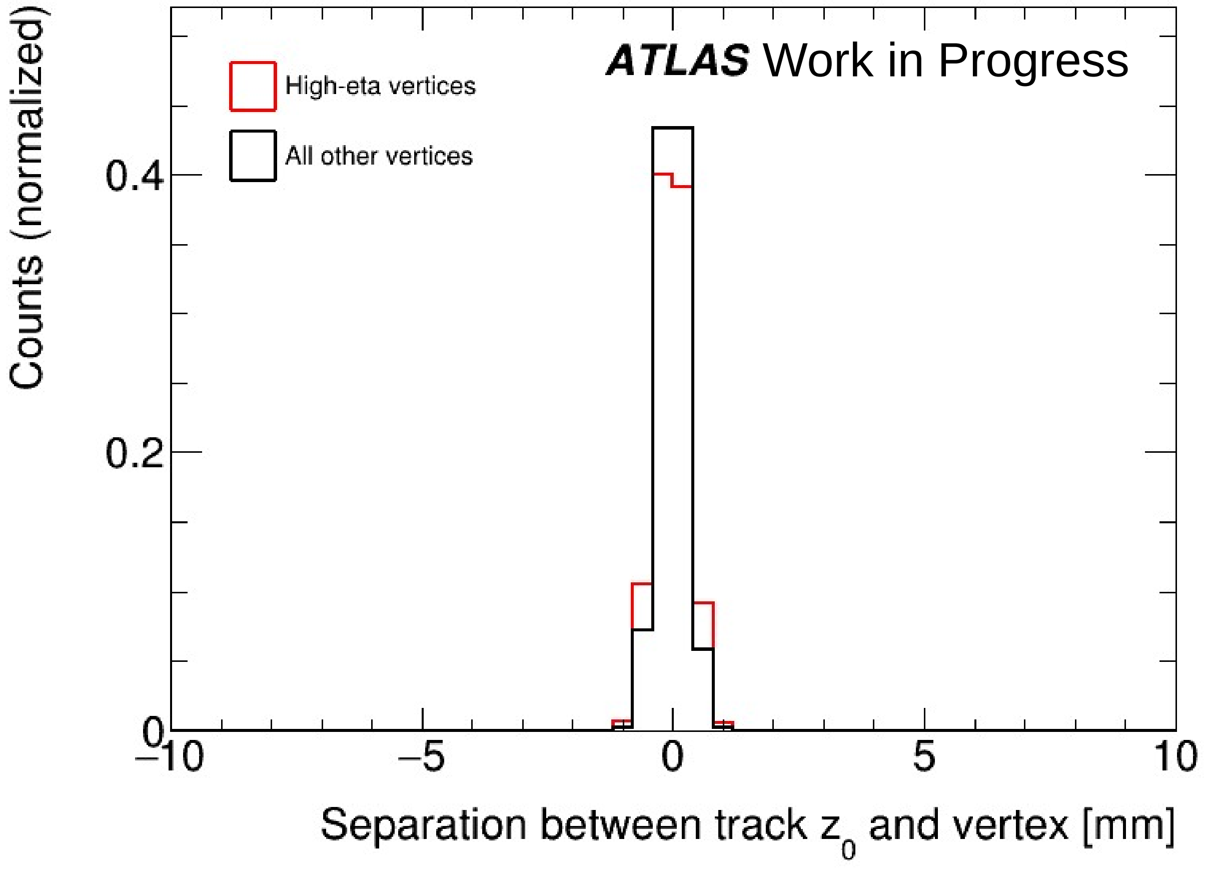
AMVF



The tail of the red z_0 distribution is much larger for high-eta tracks because of resolution and detector effects, so a hard z_0 cut hurts those tracks more.

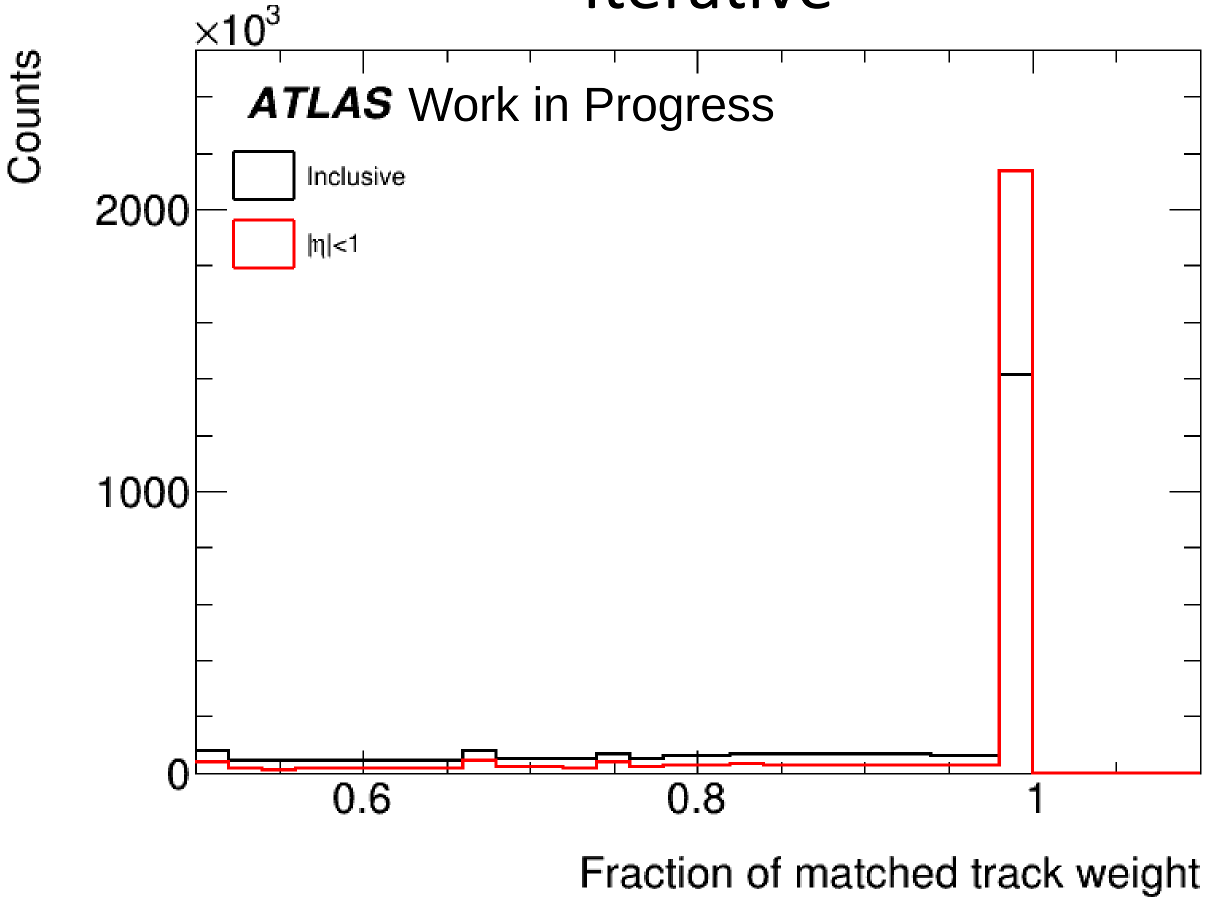
Track-reco (matched only)



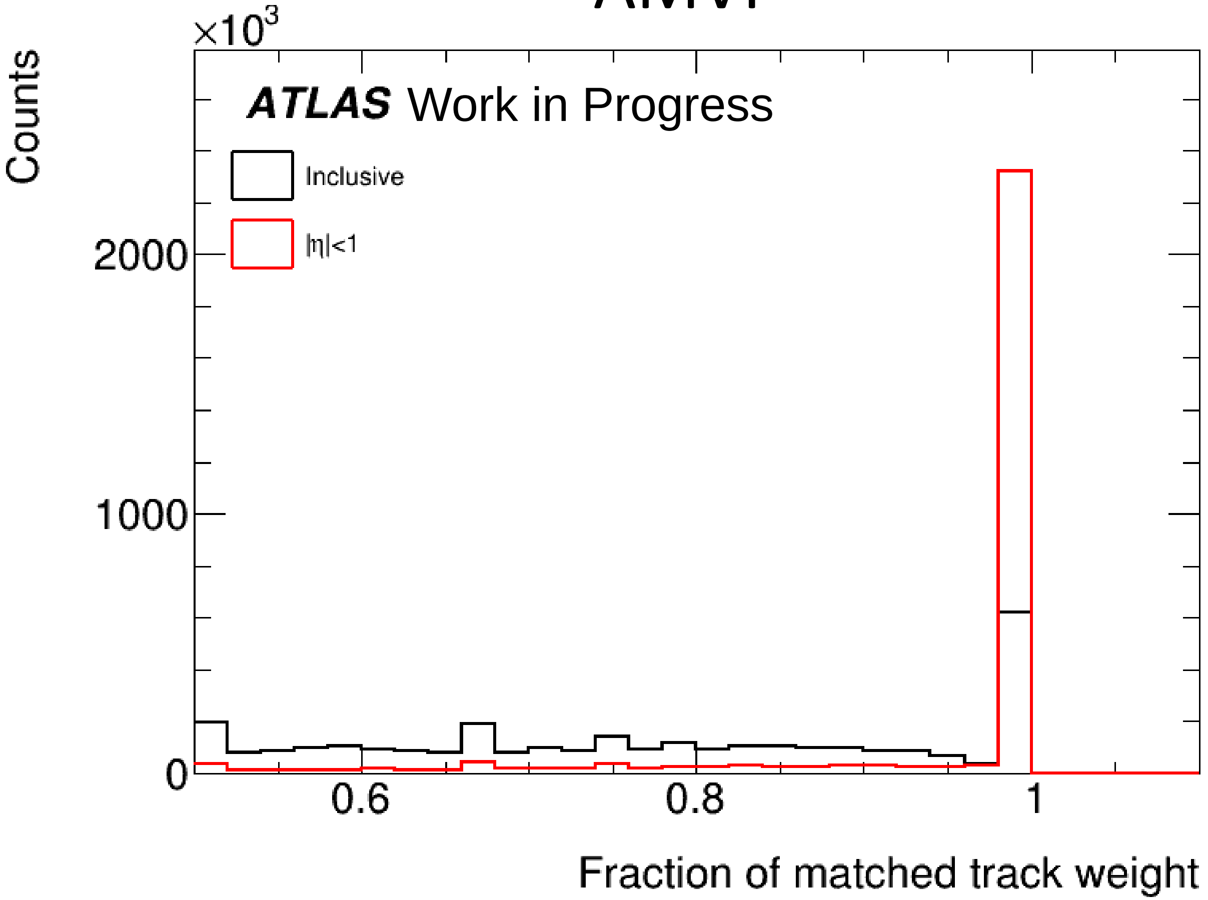


High-eta contamination

Iterative



AMVF



The hard z_0 cut means that by default, we will always incorrectly assign some tail fraction of the high-eta tracks to other reco vertices. This leads to a significant amount of track weight contamination (the drop in the black curve at 1).