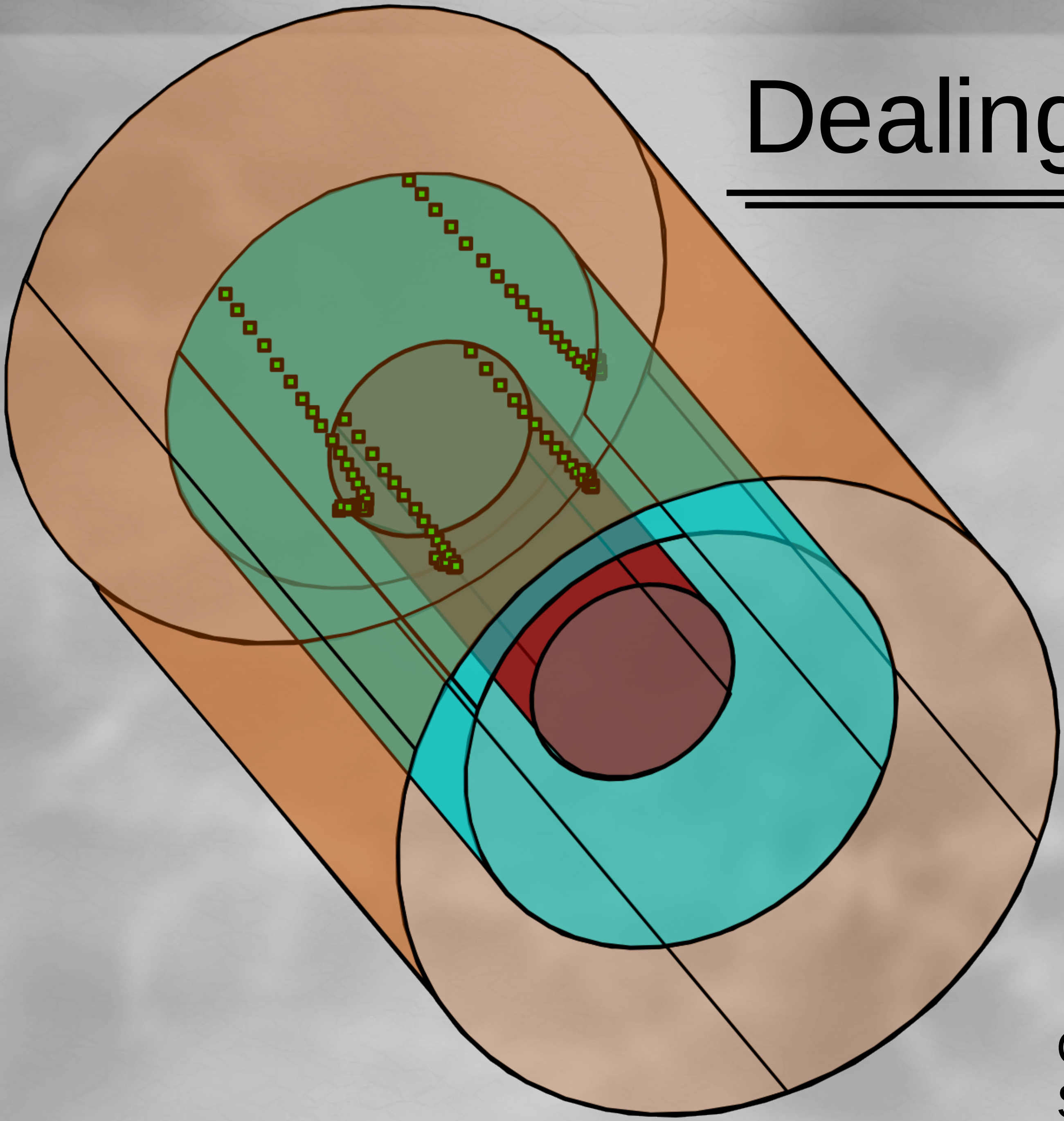


Dealing with curlers

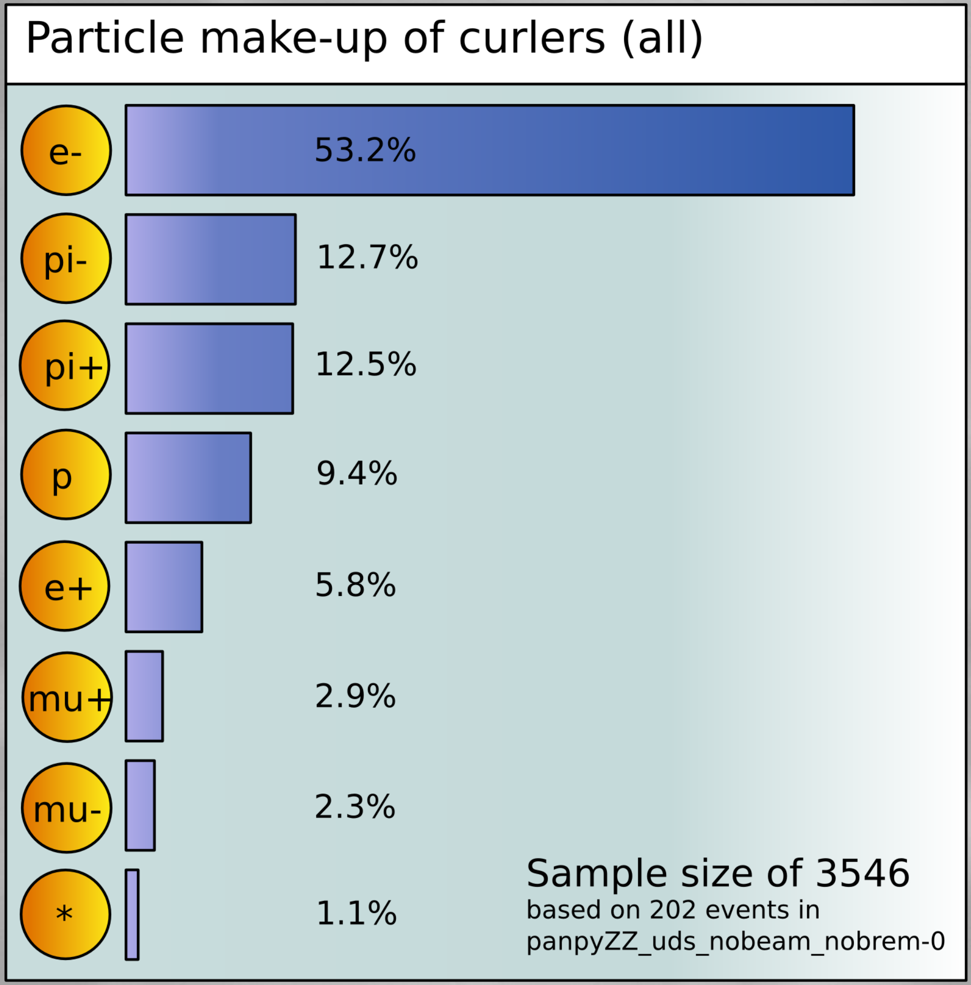
Cosmin Deaconu
Stanford University/SLAC

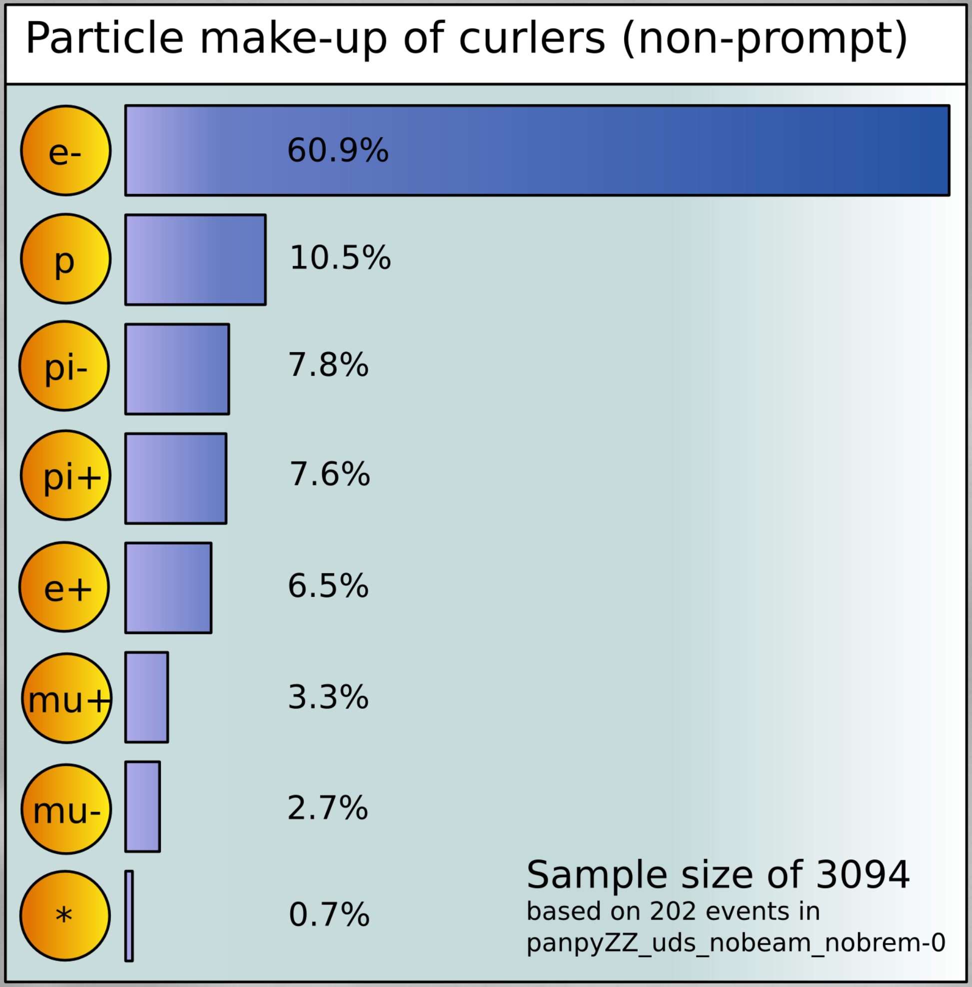
Dealing with curlers - Introduction

What is a curler?

A curler is a low-momentum particle that has multiple hits in the same barrel layer.

Dealing with curlers - Introduction





Dealing with curlers - Introduction

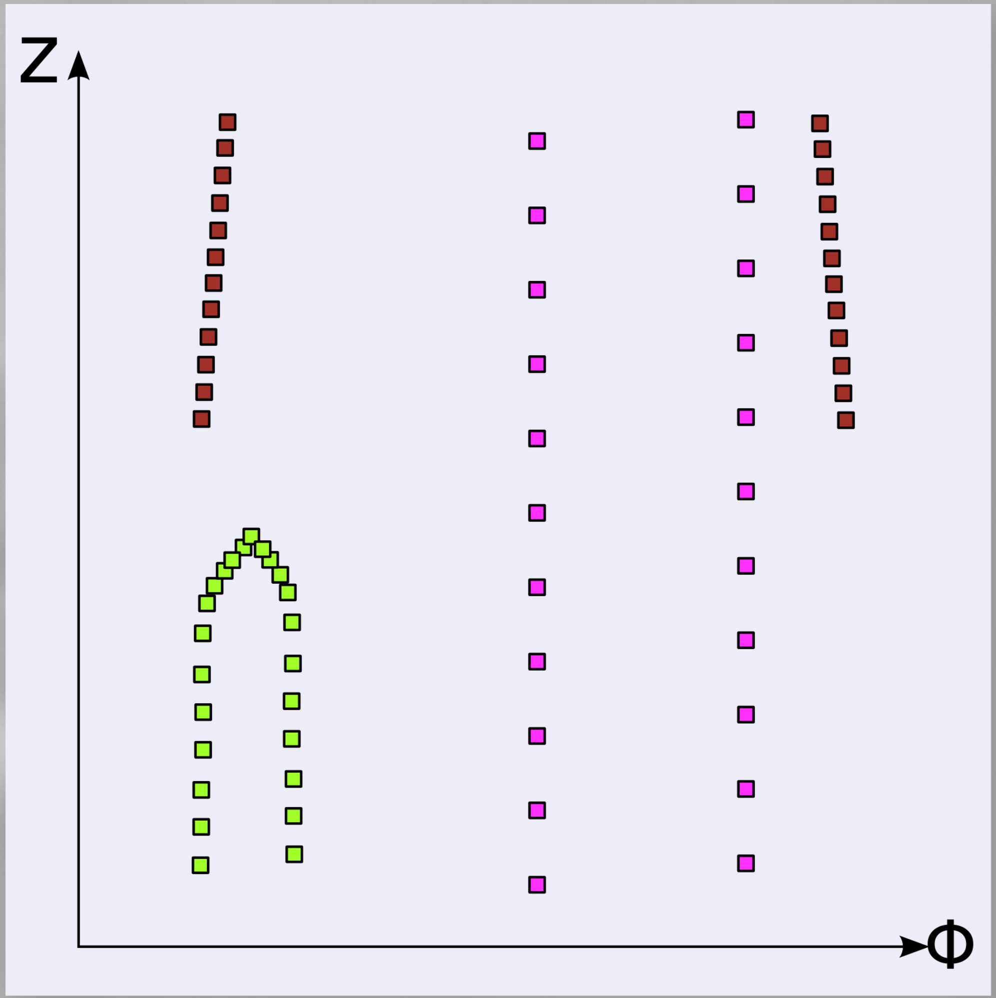
Why do we care about them?

- They are hard to make tracks from
- They have a lot of hits, slowing down tracking
- They get in the way of track findng algorithms

Dealing with curlers - Identification

What do they look like?

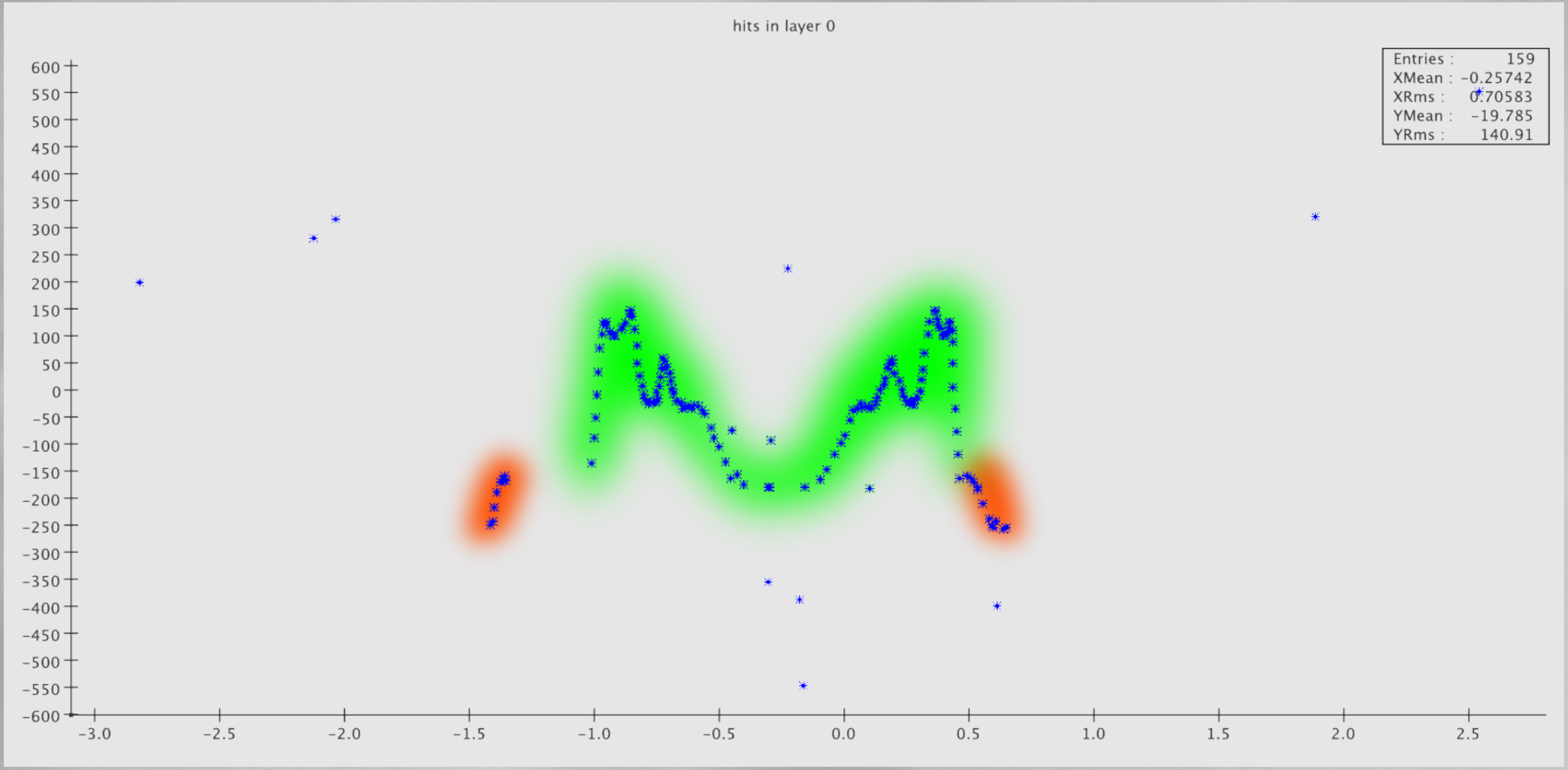
- They make symmetric patterns in the φ-z plane.



Dealing with curlers - Identification

What do do they look like?

- Here is a more complicated example



Dealing with curlers - Implementation

CurlerKiller.java

- In my contrib area (contrib.CosminDeaconu)
- Really messy, not documented very well
- See CombinationDriver.java for an example of how to use it (also in my contrib area)
- Outputs 4 hit collections:
  - Hits belonging to curlers found by algorithm
  - Hits belonging to curlers found by cheating
  - The complements to both of the above

Dealing with curlers - Implementation

Cheating

- Cheating works by flagging the MCParticles that have more TrackerHits than layers. The hits belonging to those particles are flagged as curler hits.
- A lot of these hits are unfindable by the actual algorithm.
- I suspect that the cheater might miscount the number of hits. More on that later.

Dealing with curlers - Implementation

The algorithm

- First, cluster hits in φ. The φ-distance cutoff is settable.
- If a hit is too separated in z, it is removed from the cluster. (settable parameter)
- Clusters must have a minimum number of hits to be considered. Lowering the number of minimum hits finds more curlers but increases the false positive rate.
- Can ignore clusters with Z-$separation_{rms}$ too high, though this might not be a good idea

Dealing with curlers - Implementation

The algorithm

- A polynomial fit ($z = f(\varphi)$) is made in each cluster. First, a linear fit is tried, if that doesn't have a good fit, we try a quadratic and then a cubic. The code doesn't do anything above a cubic right now, for two reasons:
  - Cubic fits already seem to have problems with the covariance matrices. Diagonal terms sometimes come out negative. This is probably a limitation of double precision.
  - Transformations and error propagation were calculated “by hand”

Dealing with curlers - Implementation

The algorithm

- Hits with large residuals from the fit are removed. Clusters with “bad” fits are ignored.
- Since we expect curlers to be made up of a symmetric arrangement of two clusters, each pair of clusters close in z is tested.
  - We test by reflecting one of the polynomial fits around the line $\varphi = \varphi_0$, where φ_0 is the average φ value of this hits in both clusters. A χ^2 test is done on the difference of coefficients in the two polynomial fits.

Dealing with curlers - Implementation

The algorithm

- To account for curlers that meet in the middle, any clusters remaining after matching pairs are split in two around the average φ and are matched by the same means above. Note that will only work with clusters that have at least four hits on each side, since four hits are required for a cubic fit.

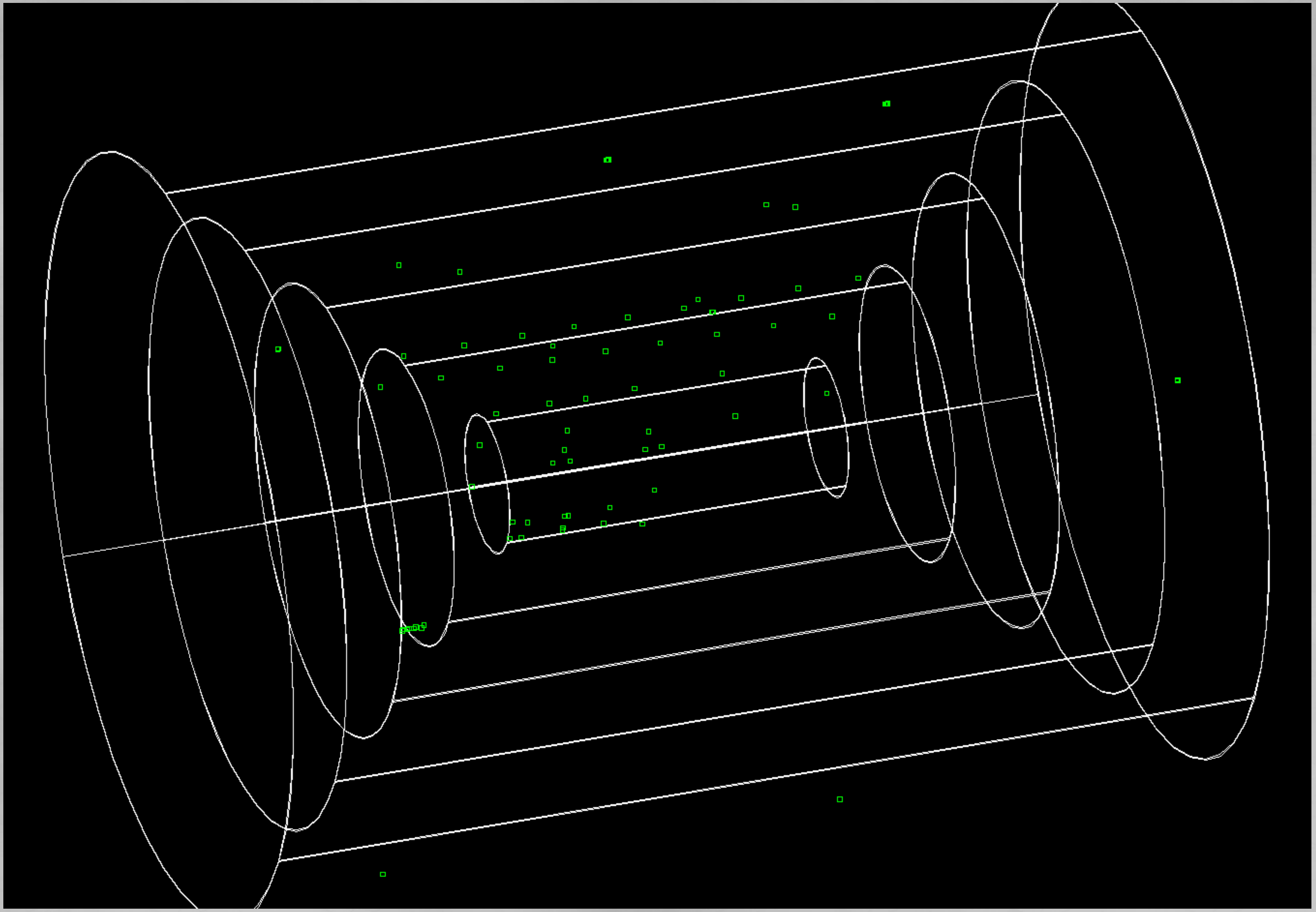
Dealing with curlers - Results

Some numbers, based on pythiaZPoleuds7:

- With a minimum cluster size of 6, about 35 percent of the hits flagged as curlers by the cheater are flagged by the algorithm with an average false positive rate of a little over 1 percent.
- With a minimum cluster size of 8, only 30 percent of the hits are flagged, but false positives decrease.
- However, inspection in Wired shows that most curlers obvious to an observer are taken out.

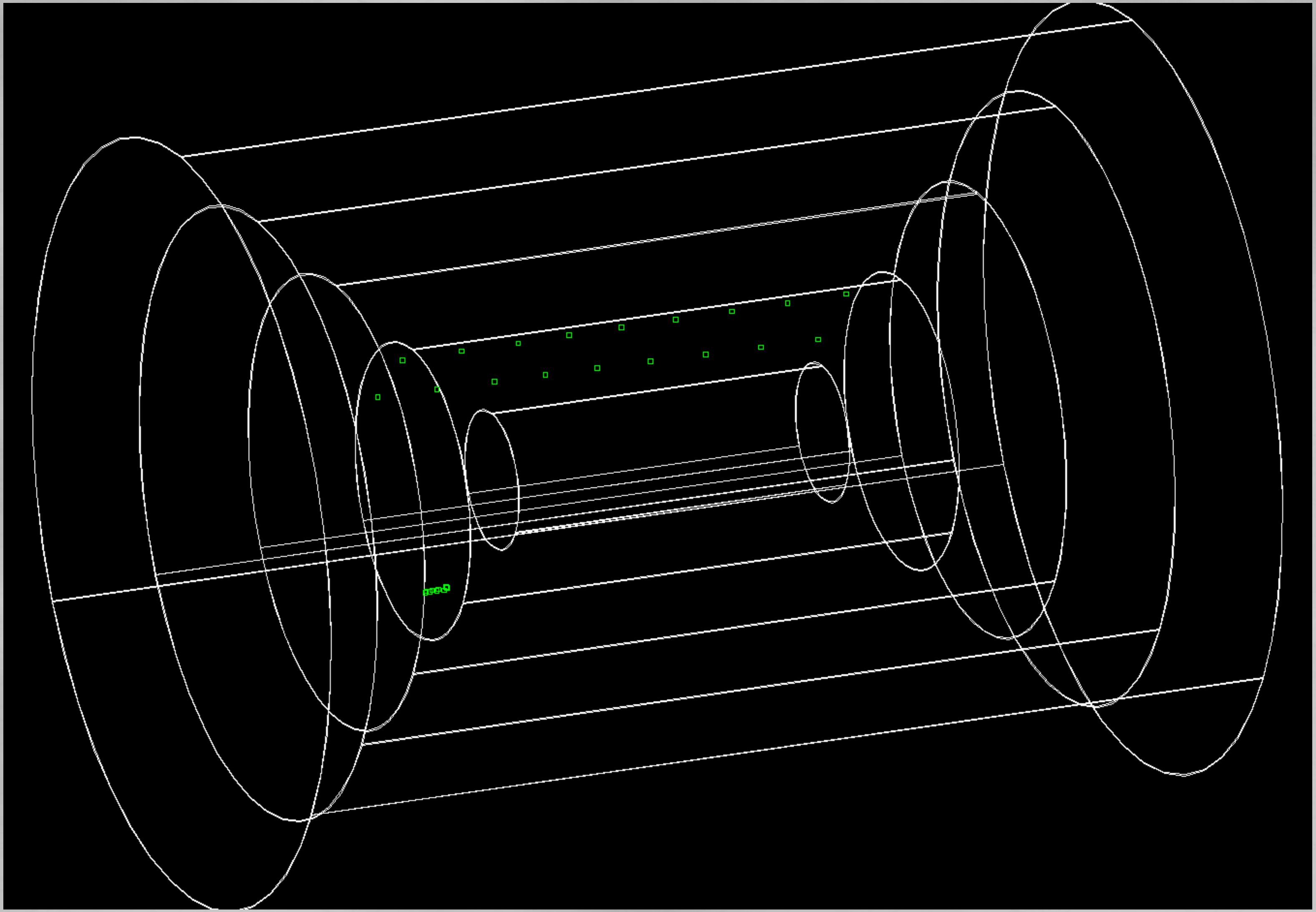
Dealing with curlers - Results

Example 1 (cheating):



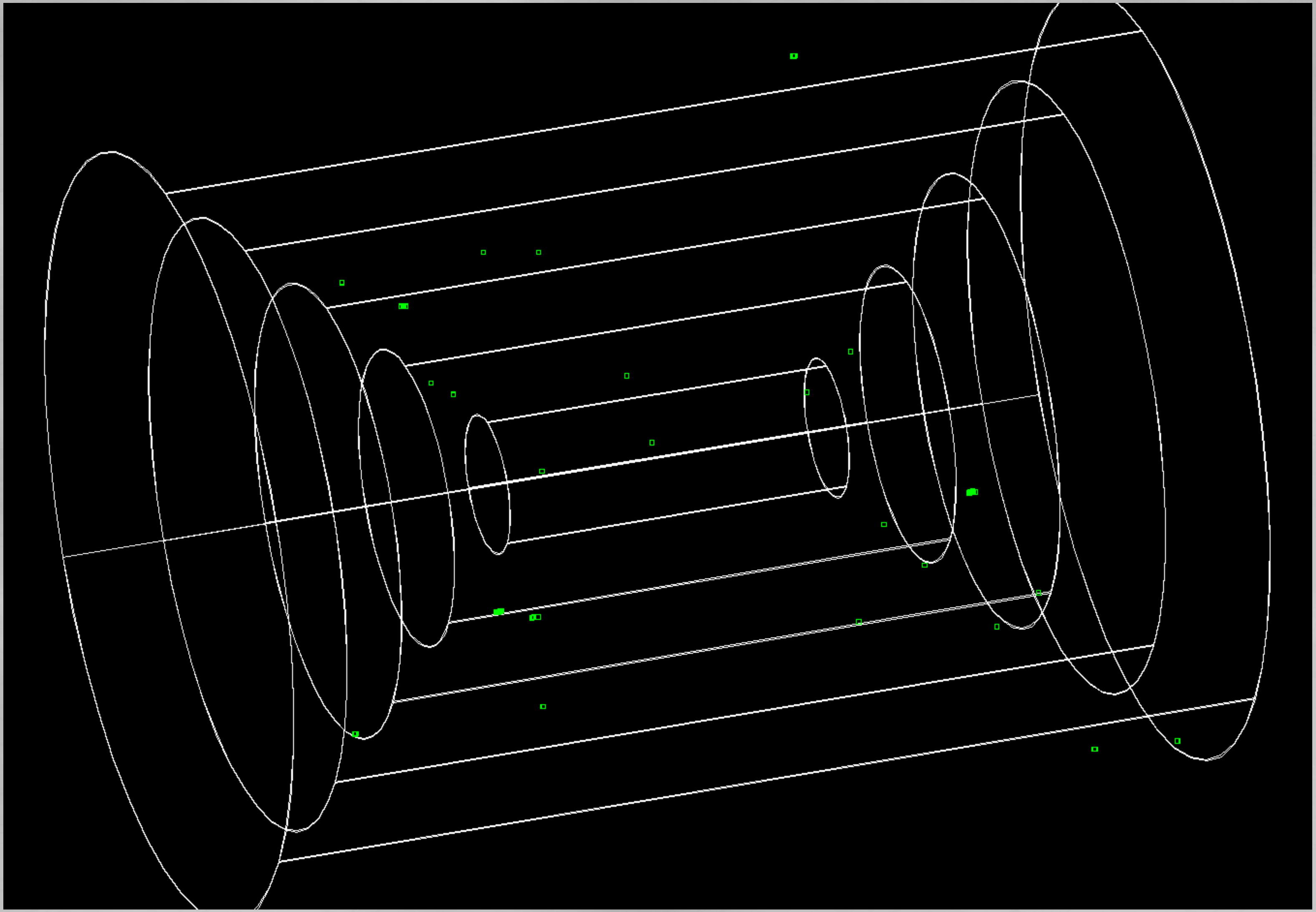
Dealing with curlers - Results

Example 1 (algorithm):



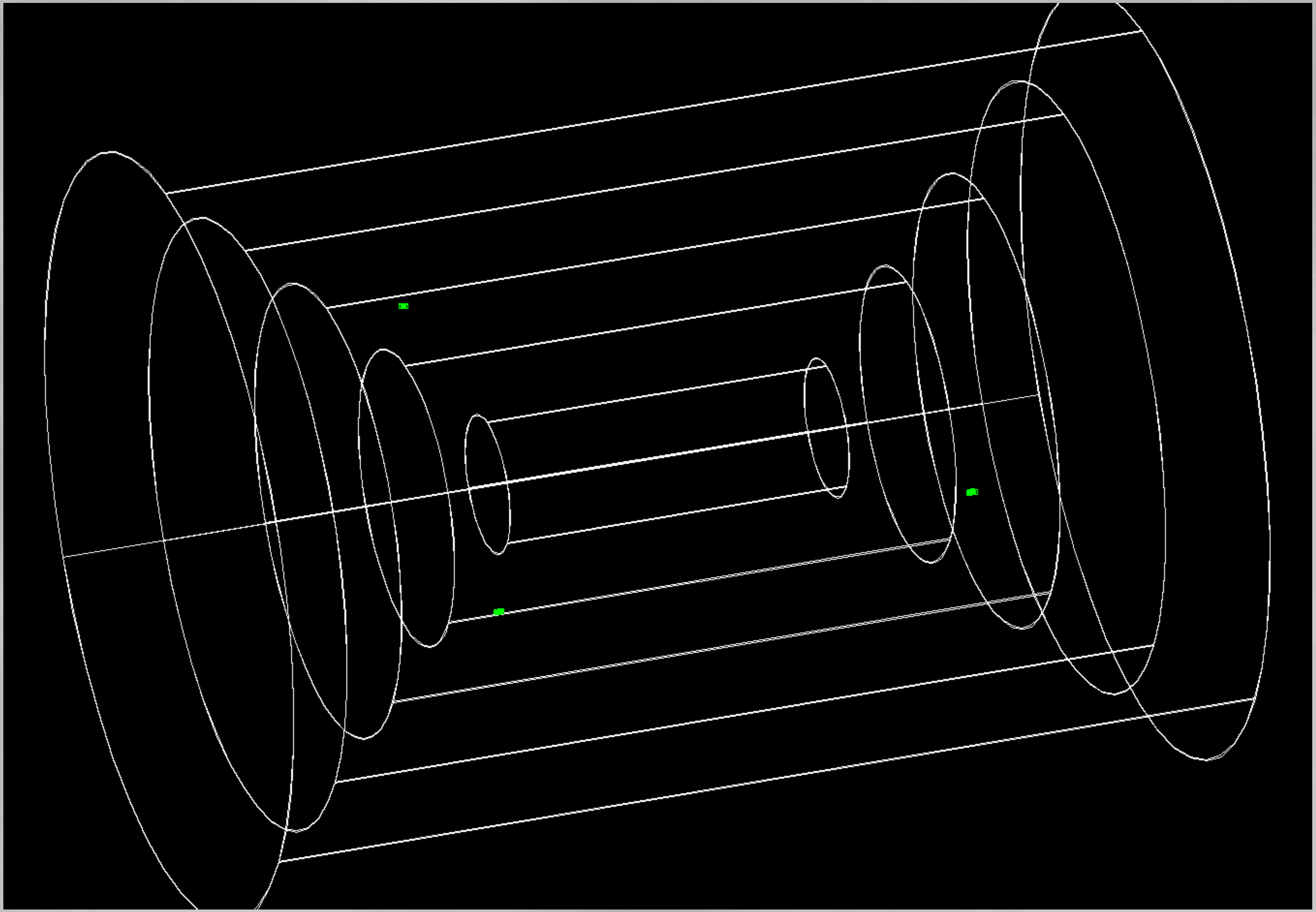
Dealing with curlers - Results

Example 2 (cheating):



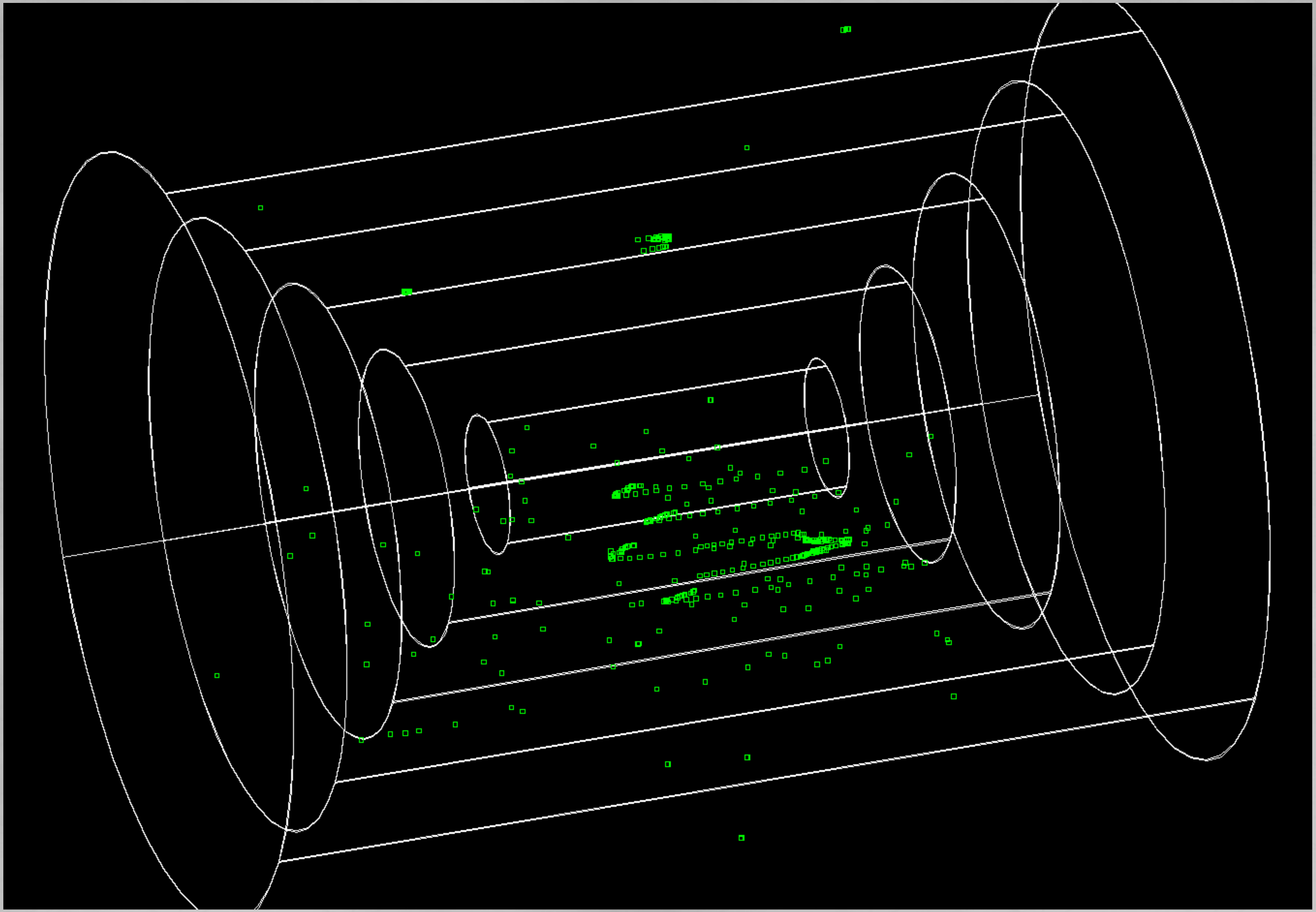
Dealing with curlers - Results

Example 2 (algorithm):



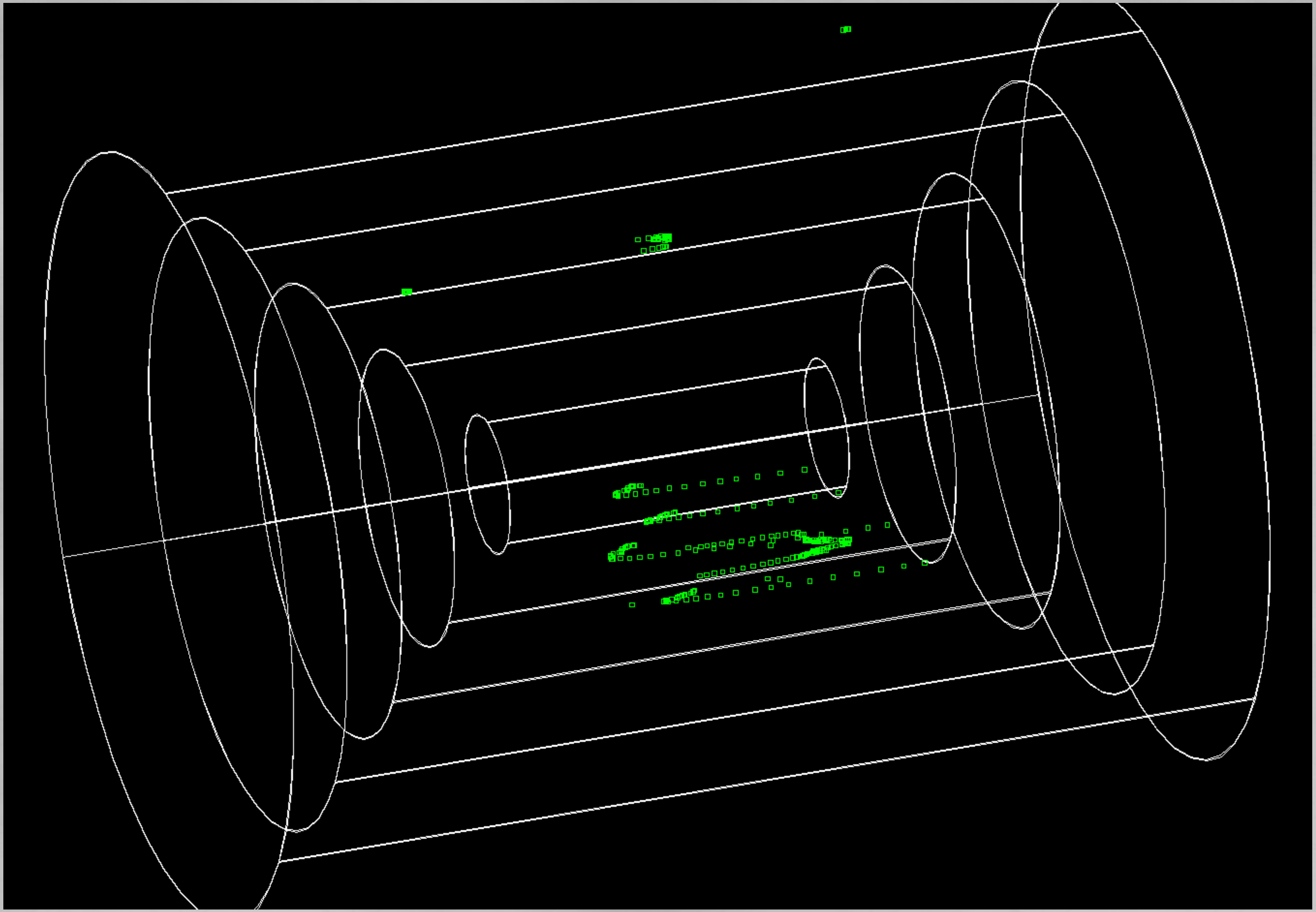
Dealing with curlers - Results

Example 3 (cheating):



Dealing with curlers - Results

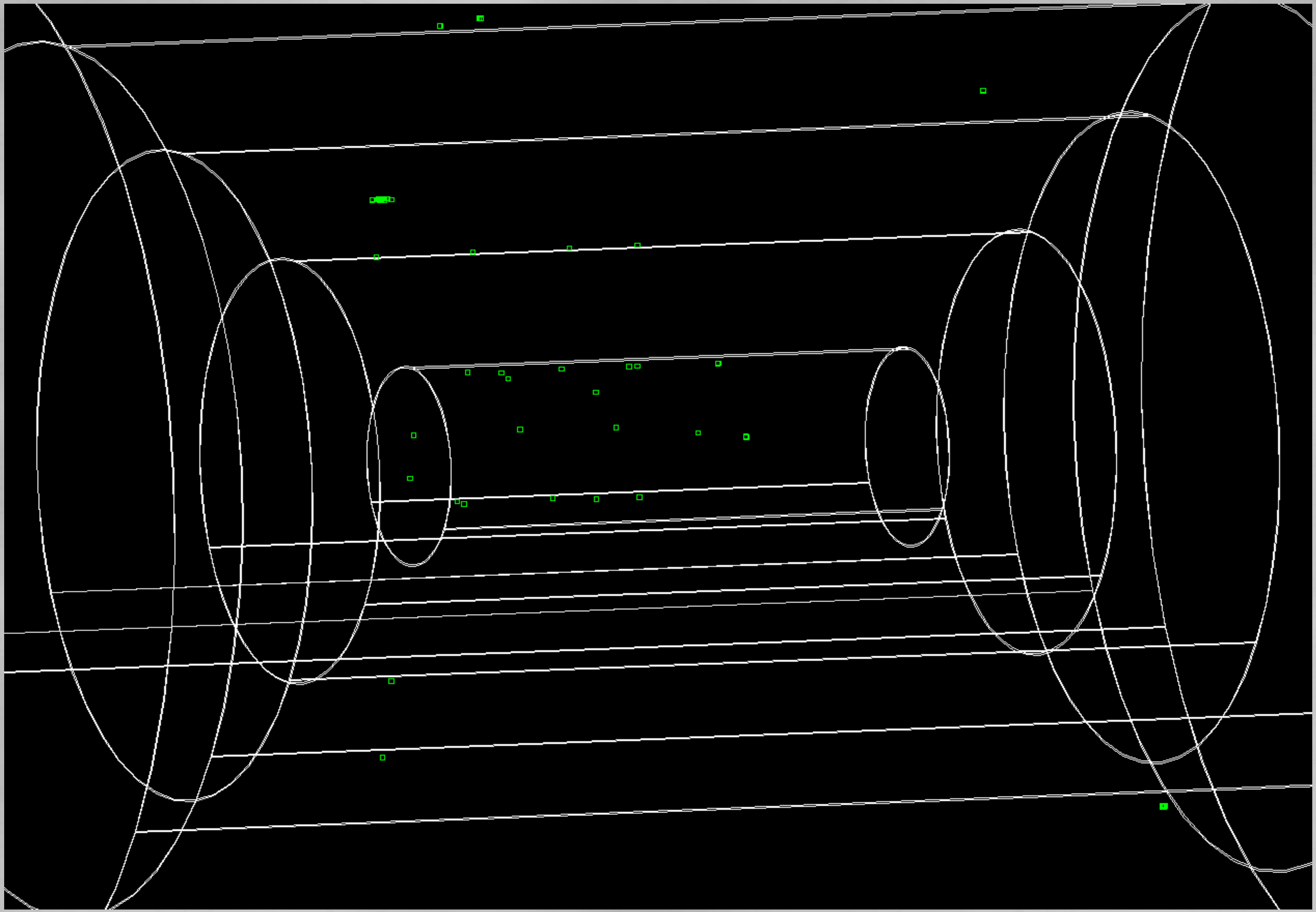
Example 3 (algorithm):



Dealing with curlers - Results

Example 4 (cheating):

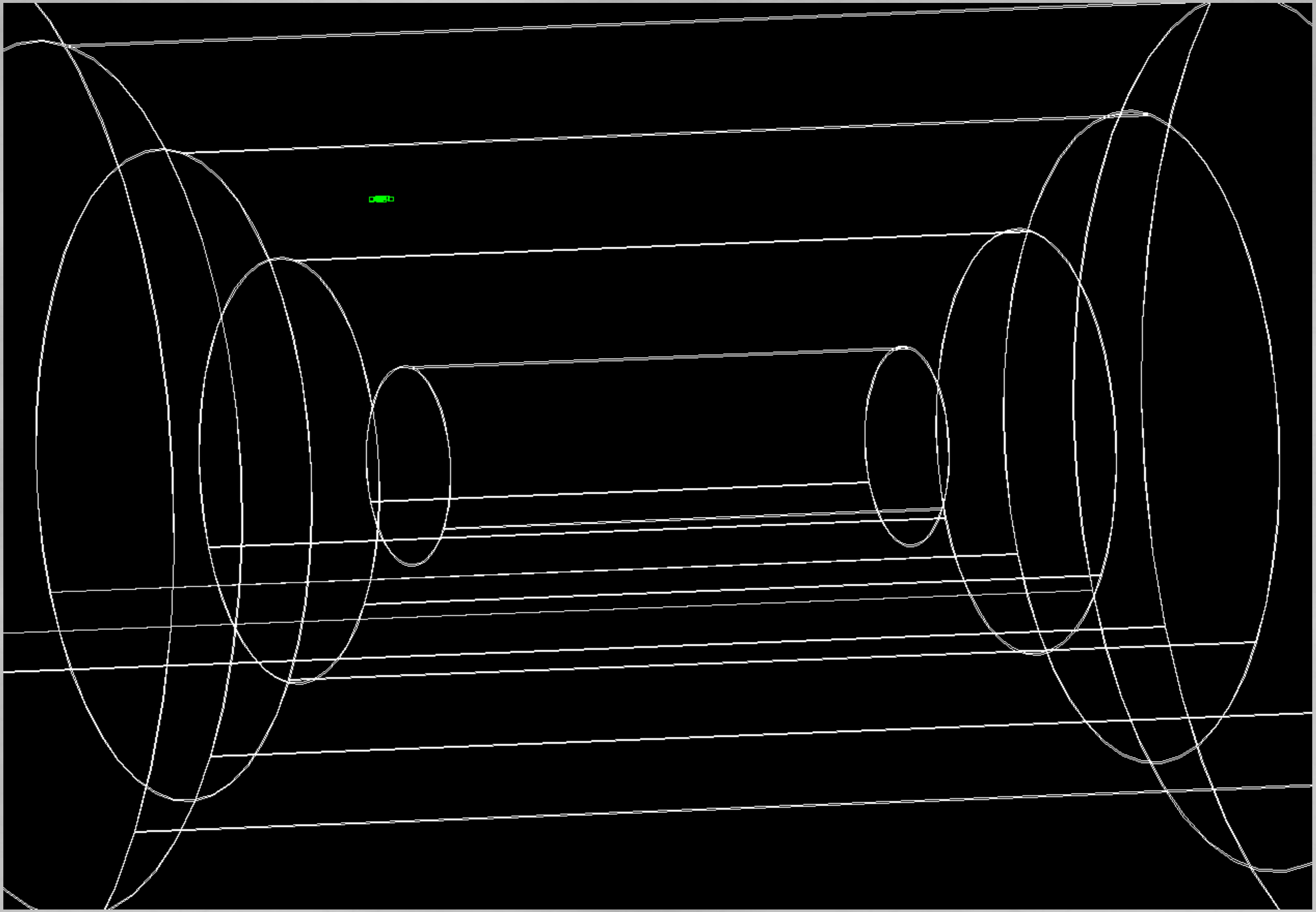
Here, there are supposed to be 73 hits in the collection



Dealing with curlers - Results

Example 4 (algorithm):

And here, 14.



Dealing with curlers - Results

Why many hits aren't found:

- The curler hits that are “missed” usually seem to belong in the following classes:
  - Curlers that don't have enough hits.
  - Hits that belong to a curler, but not in the same layer as the curling behavior. A particle may pass through several layers before it starts curling.
  - Curlers that are more complicated than cubics
  - Curlers that have hits nearby that prevent a good fit from being made.

Dealing with curlers - Results

About false positives:

- False positives are defined as hits found by the algorithm but not by the cheater.
- They happen when a non-curler hit occurs very close to a cluster.
- The z-separation and residual cuts are meant to reduce them, but ultimately, if they are close enough, they are very hard to get rid of.

Dealing with curlers - Applications

Finding new tracks:

• Curler tracks are hard to find. Most are non-prompt and they often don't form helical tracks. TrackerHitTrackFinder can find well-behaved curlers that do have helical tracks, but there are no facilities right now (as far as I know) to detect others. By knowing which hits belong to a given curler, it should be possible to fit non-helical tracks to them... somehow.

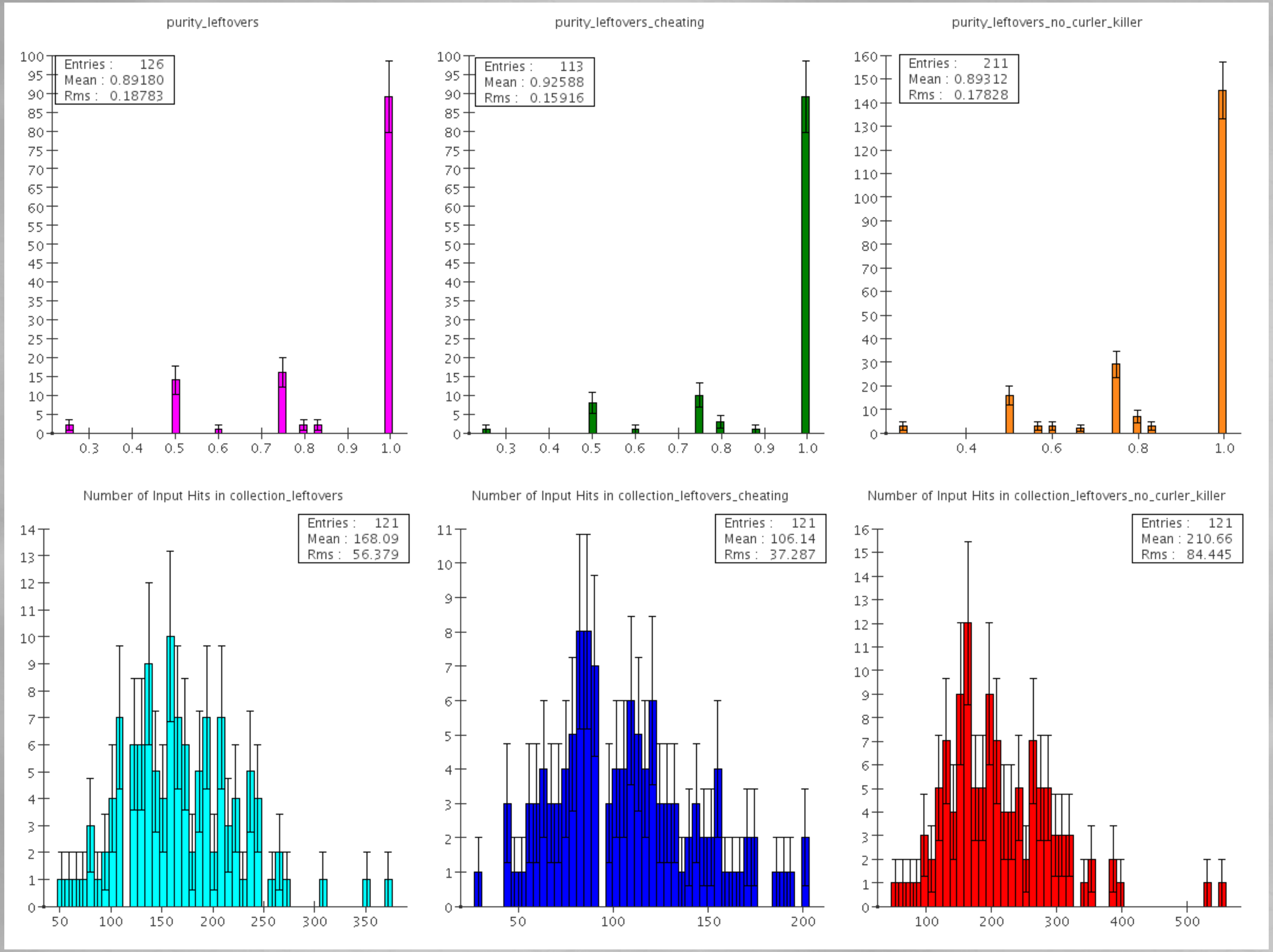
Dealing with curlers - Applications

Speeding up TrackerHitTrackFinder:

- This was the original motivation for writing CurlerKiller (to reduce the number of input hits to THTF).

Dealing with curlers - Applications

Speeding up TrackerHitTrackFinder:



Dealing with curlers - Applications

Speeding up TrackerHitTrackFinder:

- The number of tracks found by THTF was reduced because curlers were removed from the input, so THTF didn't find as many curler tracks.
- Curler tracks tend to have higher purity in THTF (there's a positive correlation between having many hits and high purity), so the purity probably went down because those tracks weren't being found, though false positives probably helped.

Dealing with curlers - Caveat

However:

- CurlerKiller relies on z information in the barrel. That is not something that will actually be known precisely.
- The algorithm can still be implemented (since we can still make φ clusters and we know a little bit of z information), but the false positive rate will be higher.
- Some curlers have z-separation > 10 cm, so those should have no problem.

Dealing with curlers – The End

Conclusion

- Identifying curlers is useful for tracking
- There is plenty of room for improvement in CurlerKiller, but it does do something.
- Due to reliance on barrel z-information, CurlerKiller is not directly applicable to the "real thing," but it is also not completely useless.