

Cosmic Ray Track Reconstruction ClusFilter

Science Board Meeting

12 / 04 / 2017

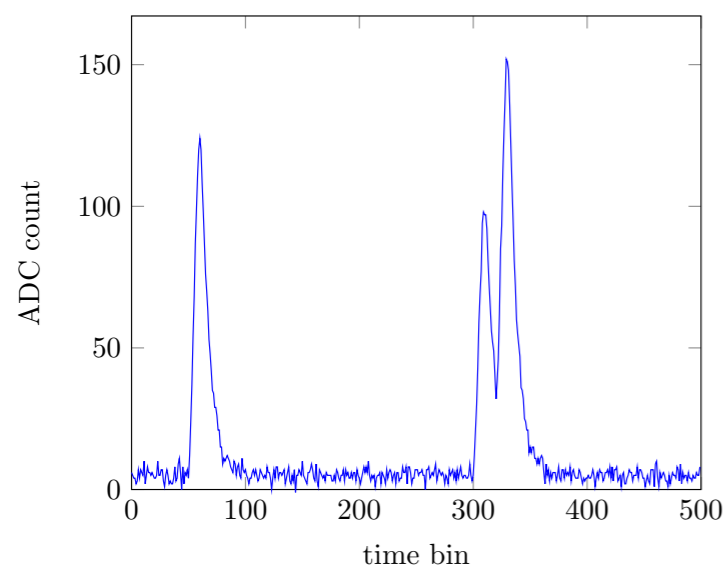
Laura Zambelli (LAPP)

Reconstruction Flow in QScan

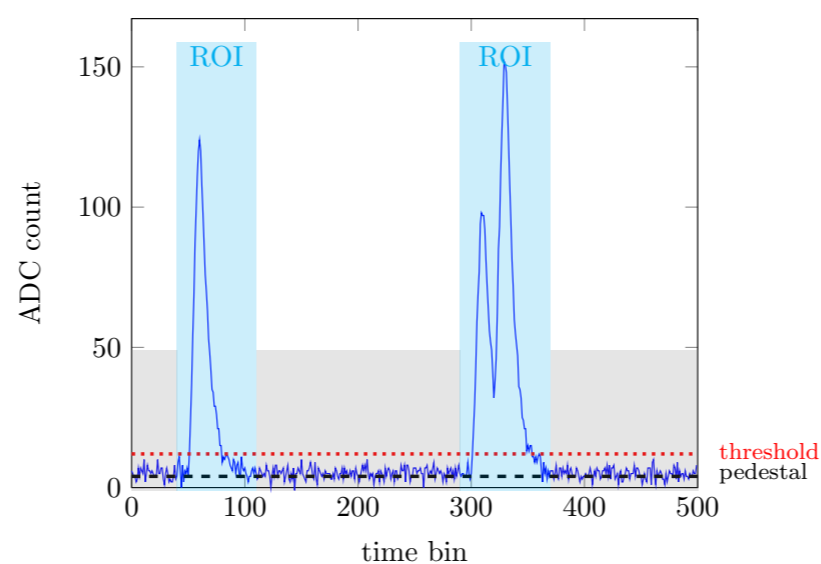
- Hit Finding

- For each channel, compute the pedestal mean & RMS
- If signal fluctuates a lot wrt to pedestal, considered as a hit

Waveform

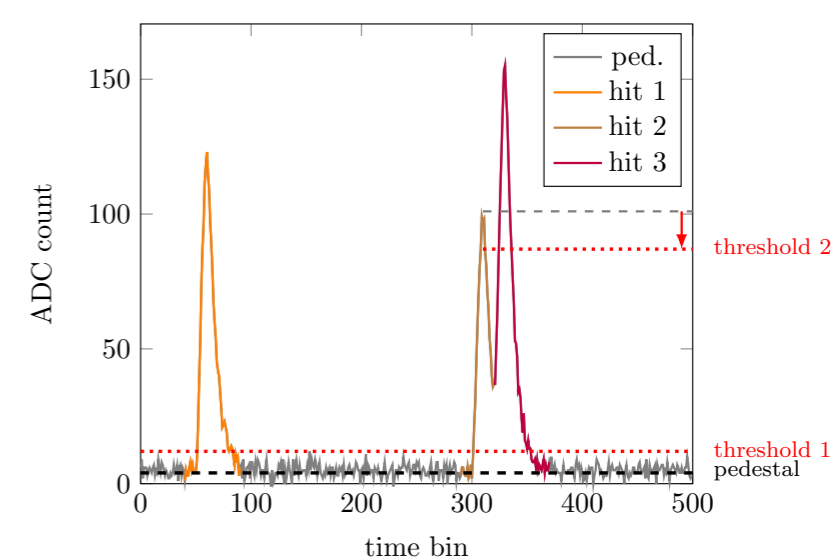


Compute pedestal mean & RMS



Starts with initial guess on the pedestal, and defines ROIs.
Refine the pedestal & ROIs with multiple loops on the whole waveform.

Defines hits



threshold 1 defines start & stop of a hit
if multiple hits close by, a threshold 2 is defined wrt hit maximum

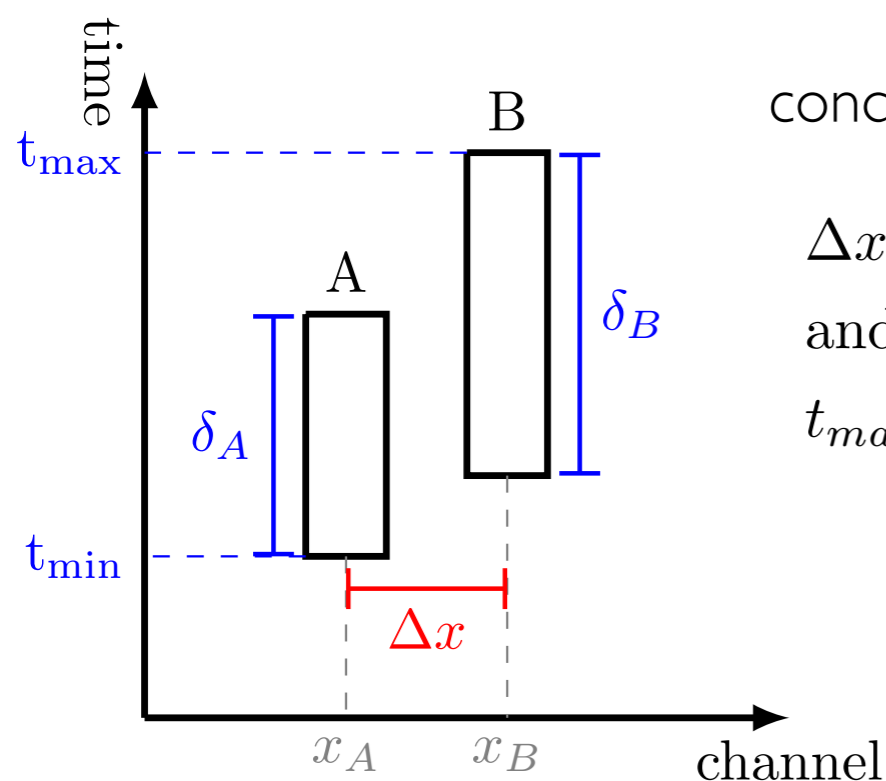
- Hit charge defined as bin integral in the hit time window + pads on each side (when possible)

In the 3x1x1 : 1280 channels with 1667 time bins
In the 6x6x6 : 7680 channels with 10000 time bins

Reconstruction Flow in QScan

- Clustering

- Found hits are ordered in increasing channel number and, within a channel in increasing time (done for each CRM in each view)
- Cluster together close-by hits (NNCluster)



conditions for hit A and hit B to clustered together :

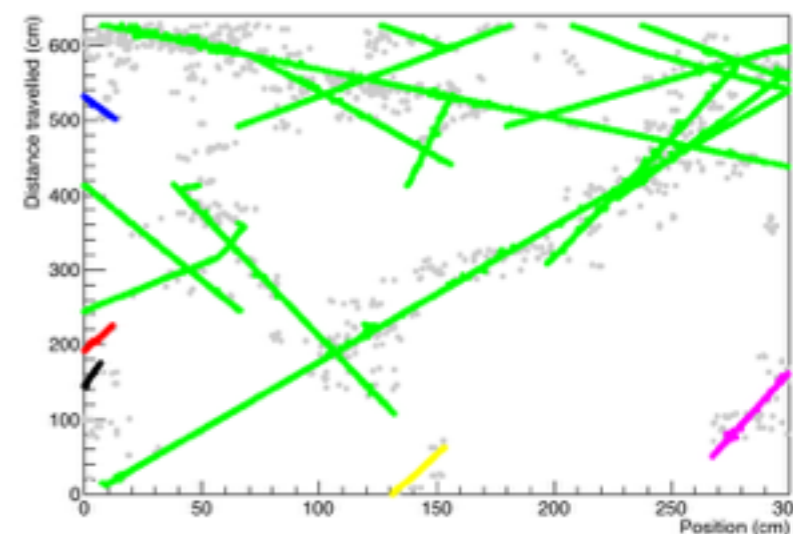
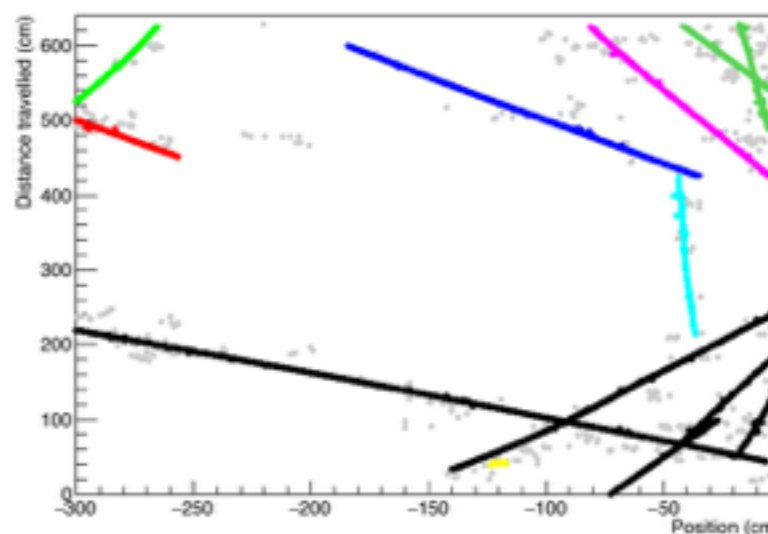
$$\Delta x < \varepsilon_x [= 2 \text{ channels by default}]$$

and

$$t_{max} - t_{min} < \delta_A + \delta_B + \varepsilon_y [= 2 \text{ time bins by default}]$$

gray : unclustered hits
| colour = | cluster

Examples



→ The clustering is CPU time consuming due to multiple loops on the found hits

Reconstruction Flow in QScan

- Track Finding

- From clusters, three algorithms available :

Hough Transformation

Imagery technique to find lines in a set of points.

TrackBuilderSm

Very naive track builder which assumes that one cluster = one track.

Gather all the hits together and smooth them to make a track.

TrackBuilderMTC

Use Kalman-Filter from the hits in the cluster to build tracks. See Slavic's presentations at previous SB for more details

Alternative CR Track finder 'ClusFilter' - general idea

The algorithm uses the fact that the hits are ordered in increasing channel number, and within a channel by increasing arrival time.

The code do not need clusters, only found hits.

Seeding : Start with 3 neighboring points (each hits are in a different channel) and fit a line using least square method. If the correlation is good enough (≥ 0.8 , hardcoded so far) start the track filtering

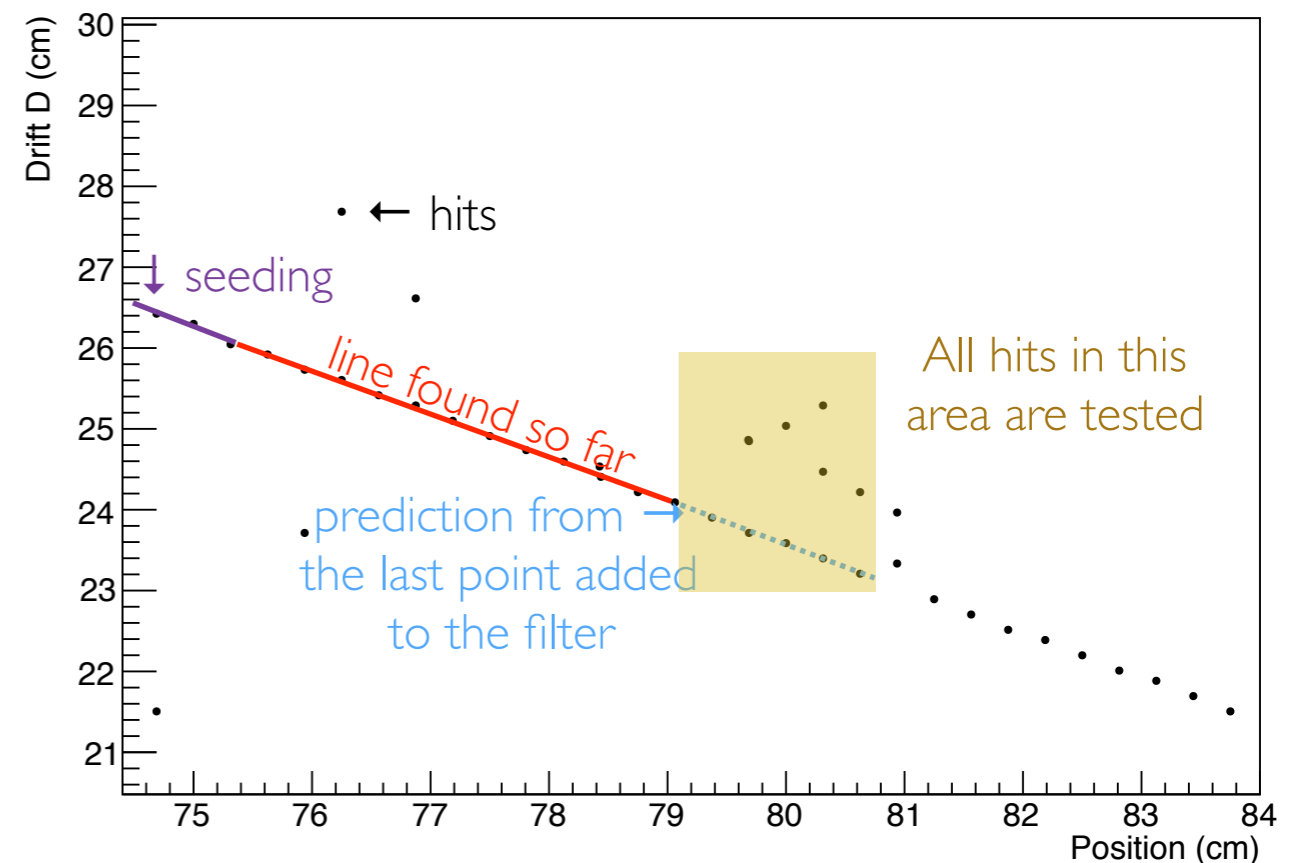
Filtering : Search for neighboring hits in increasing channels [default is 5 channels apart] in some time windows [default is $\pm 2\text{cm} * (\Delta\text{channel})$].

For each hit, compute:

$$\chi^2 = \frac{(y_{\text{predicted}} - y_{\text{measured}})^2}{\sigma_{y,\text{data}}^2 + \sigma_{y,\text{filter}}^2}$$

and considered if $\chi^2 < \text{cut value}$ [default is 10].

NB :The time window search definition implies that no tracks with a slope ($\Delta y / \Delta x$) higher than 6.4 (2/0.3125) can be found



Alternative CR Track finder 'ClusFilter' - general idea

Considered hits are then ordered in channel and X^2 .

The closest and 'best' one is used for updating the filter.

If no hits can be considered, the filtering stops and the track is constructed if the number of attached hits is high enough [default is 20 hits].

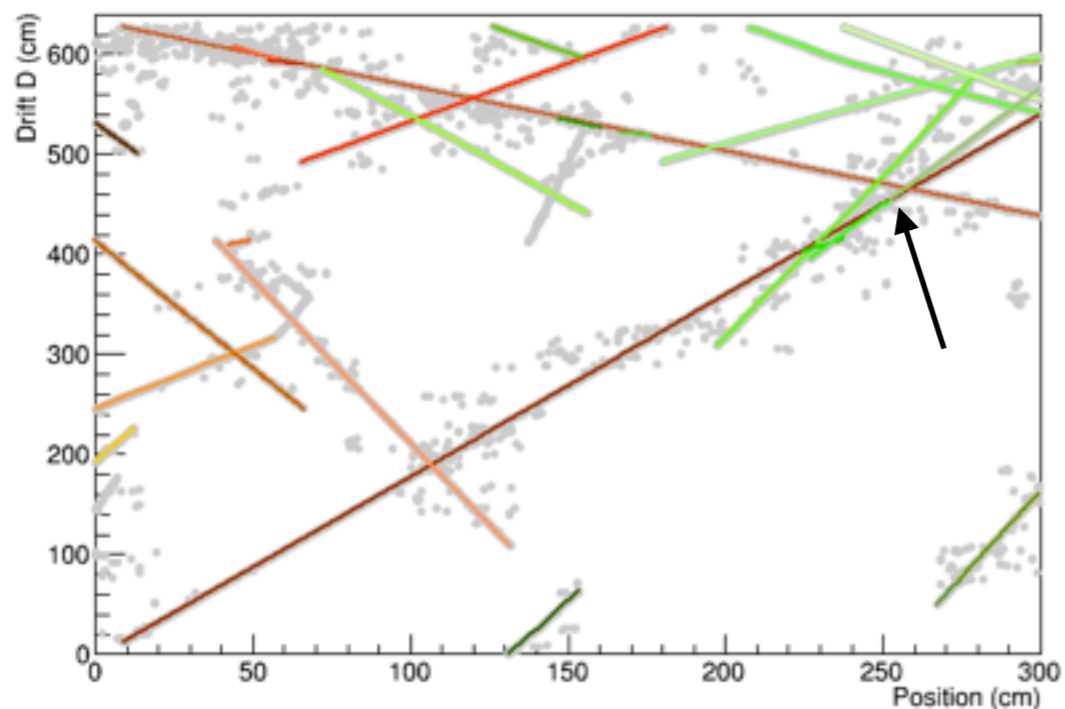
Using the attached hits to the track, a backward filter is performed to get the initial and final slope of the track.

The filtering is a Kalman-like filter (developped by Pierre Billoir, NIM A225 (1984) 352-366, mathematically equivalent to Kalman Filter)

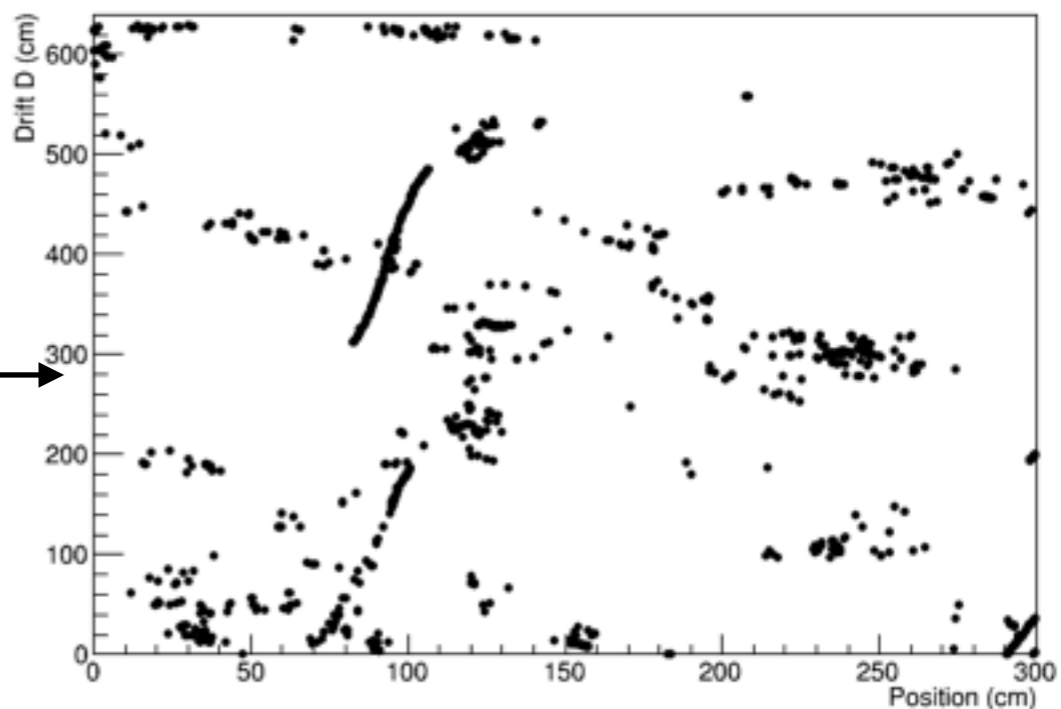
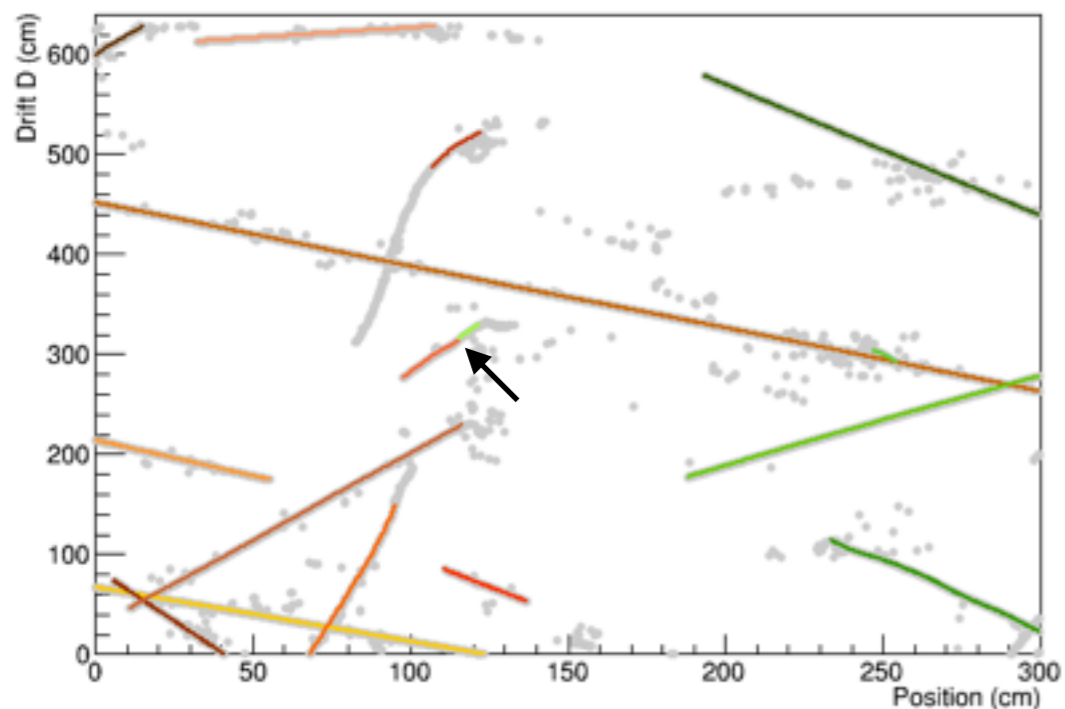
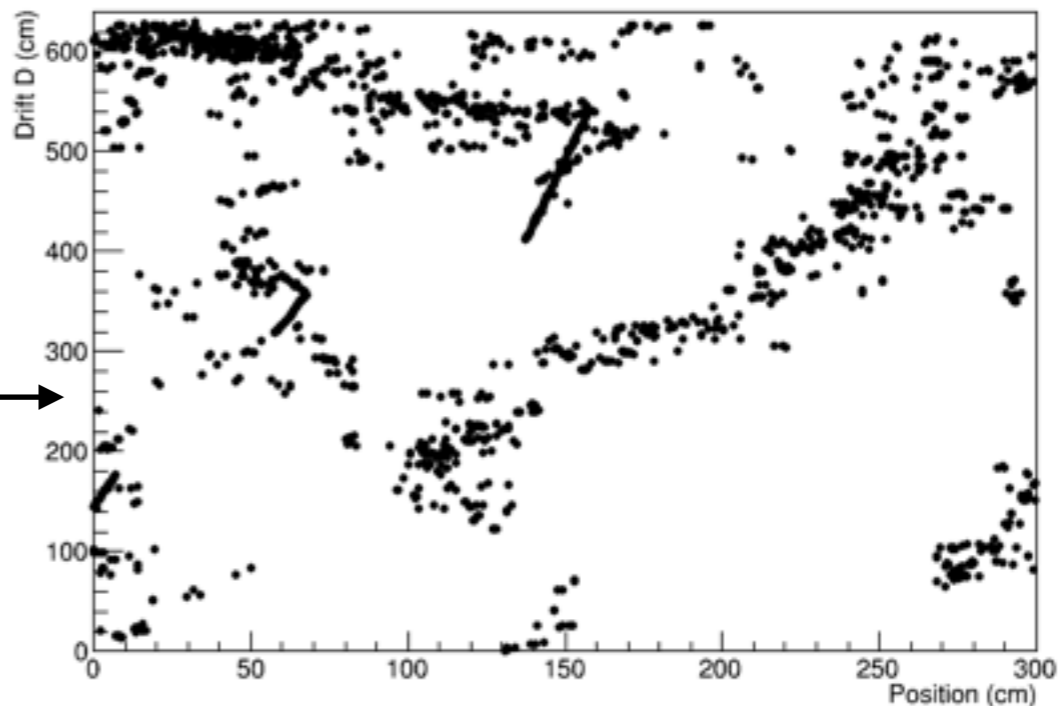
It's a 2D filtering, at each new point the slope and y position is updated, taking into account multiple scattering given an assumed track momentum [default is 1 GeV]

Alternative CR Track finder 'ClusFilter' - Examples

Tracks found (one CRM)



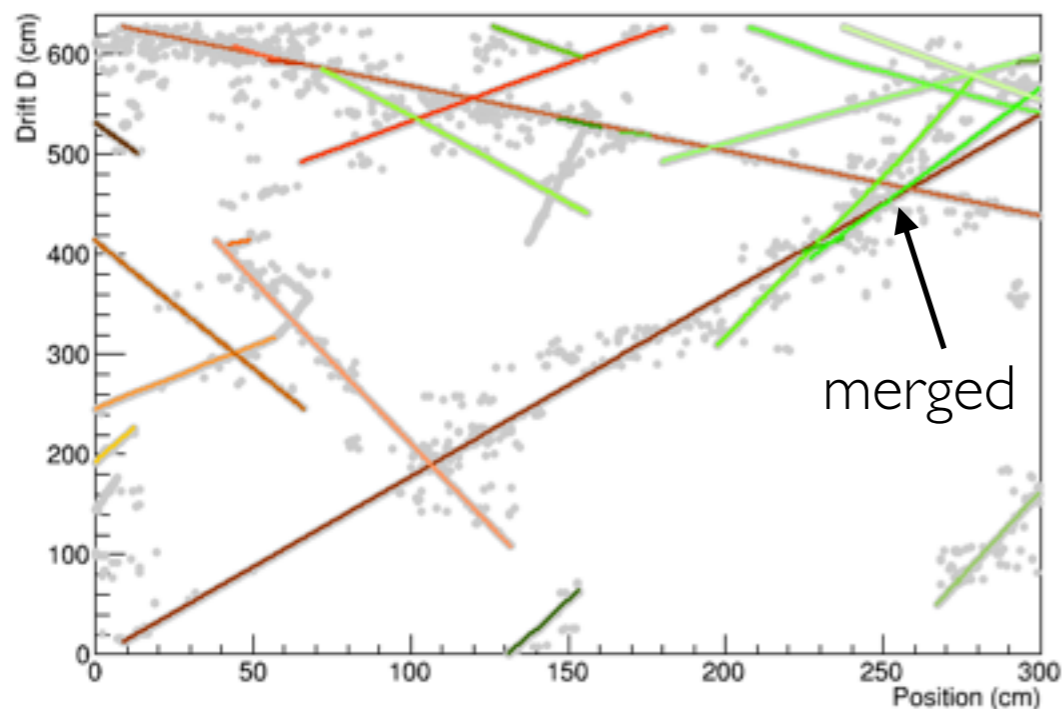
Unmatched hits



one line = one track (colors are random)

Track Merging

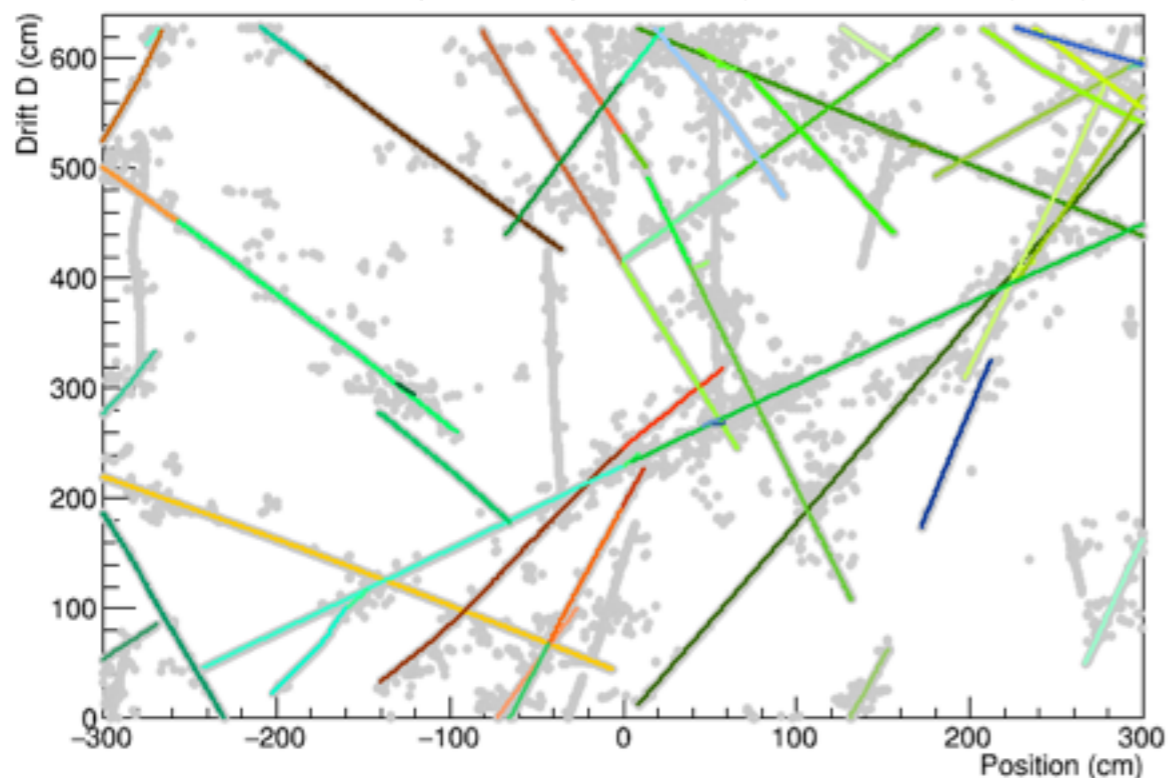
Tracks found & merged within CRM



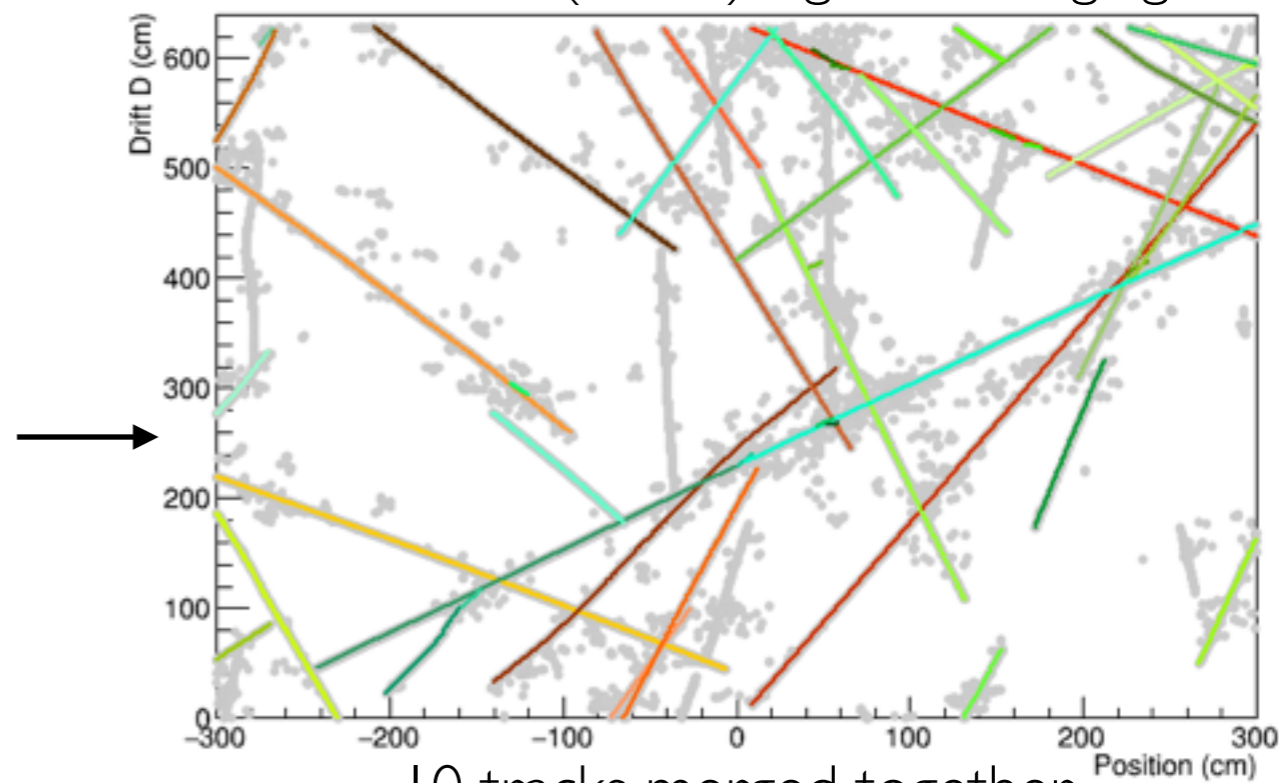
- Within a CRM, try to merge broken lines.
- In between CRM, try to stitch together tracks crossing several CRM.

2 tracks are merged together if the end-point distance is shorter than a distance [default is 5 cm] - can handle separated or superimposed end points - and if their slopes are compatibles within a certain number of sigma (slope error) [default is 5σ]

All CRMs (view0) - no global merging



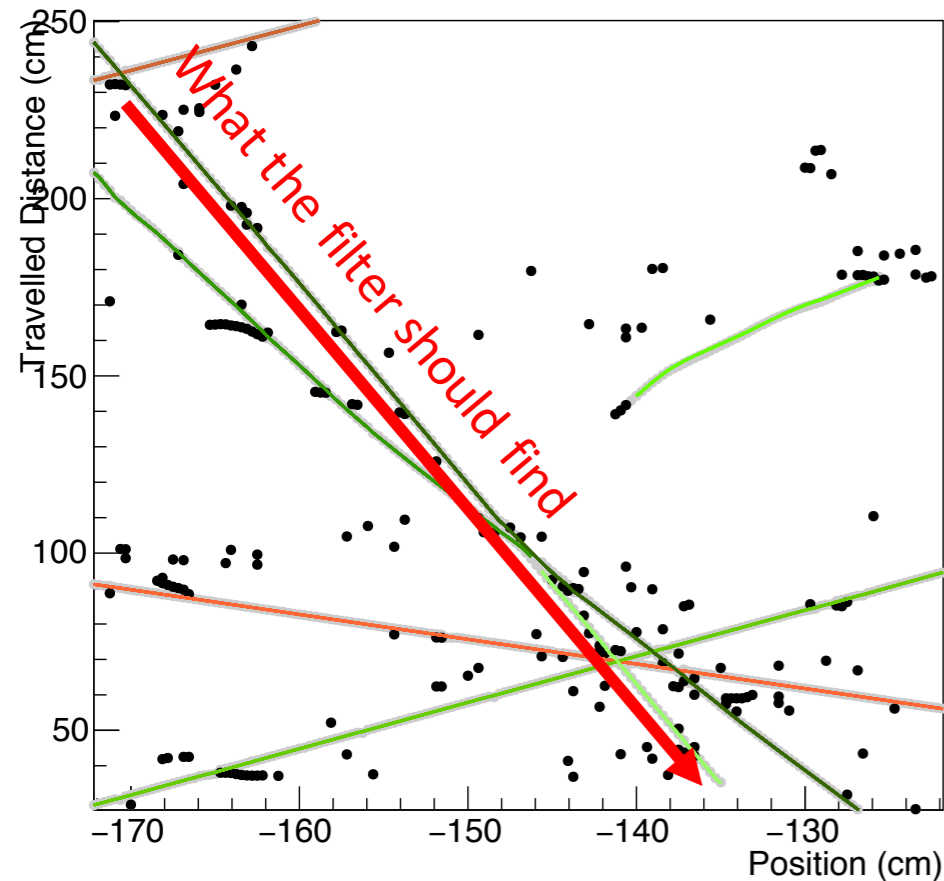
All CRMs (view0) - global merging



10 tracks merged together

Alternative CR Track finder - issues

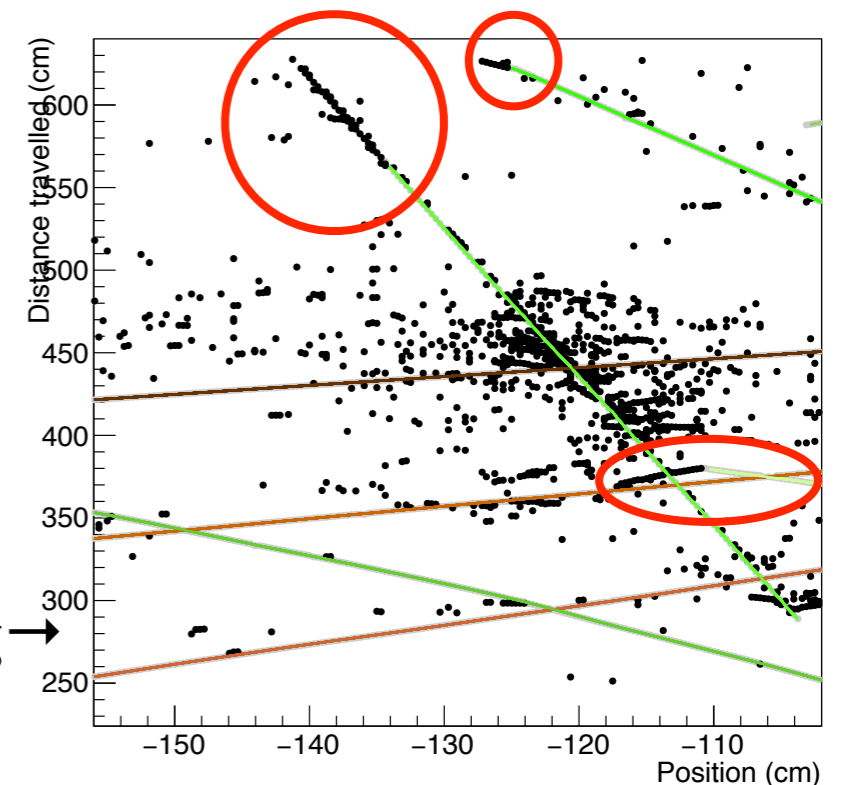
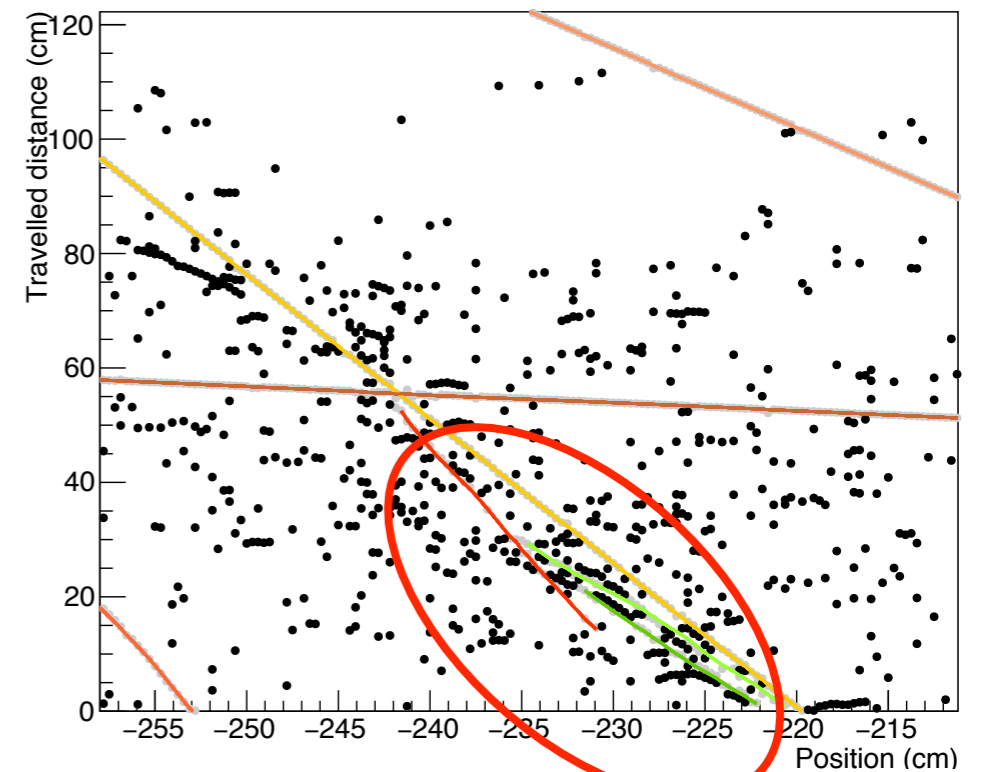
A few issues remains, although most of the tracks are correctly found.



↑ Due to some allowance in track bending from MS, sometimes the filter gets the wrong direction when 2 tracks with similar slopes crosses each other

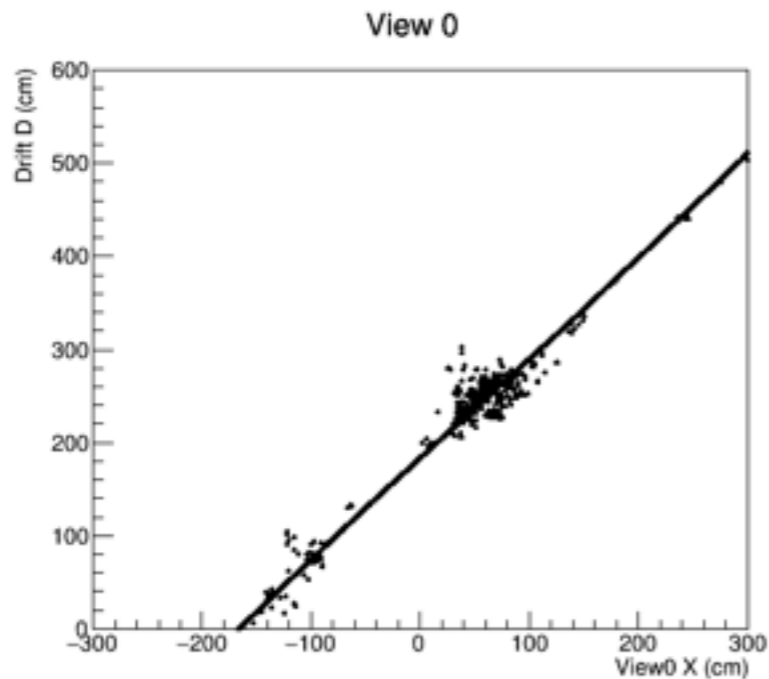
+ vertical tracks

In crowded areas, random tracks can be found ↓

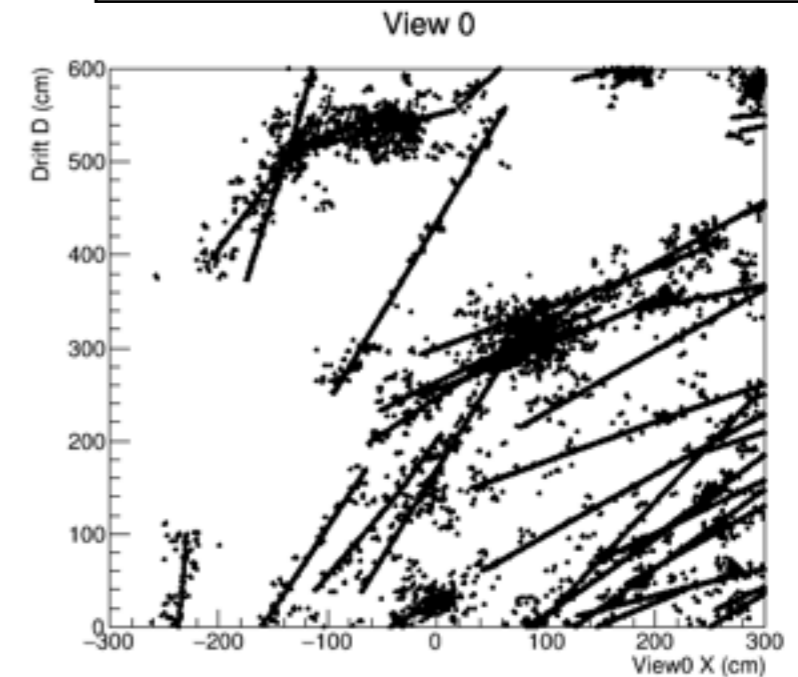


Testing code CPU performances (in the 6x6x6)

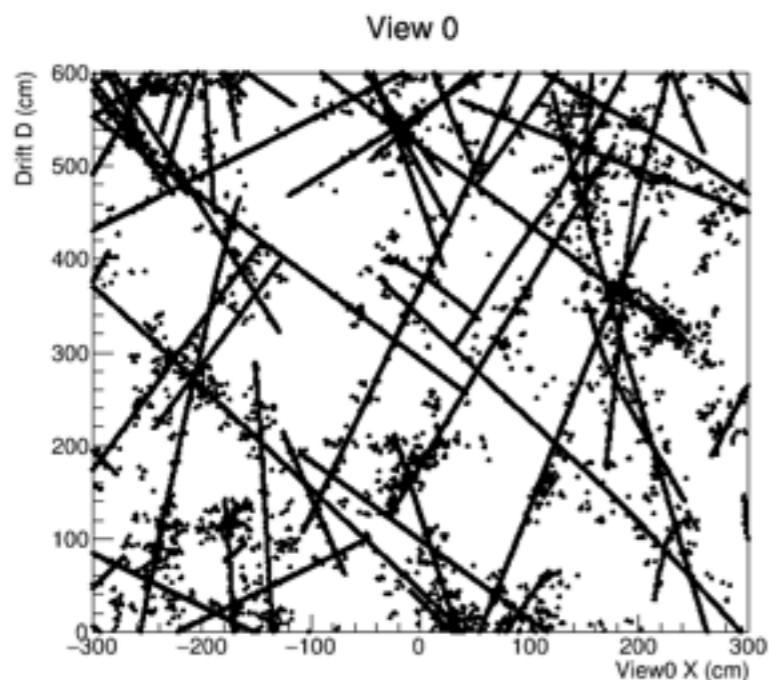
4 GeV through going muons
~900 hits found per view



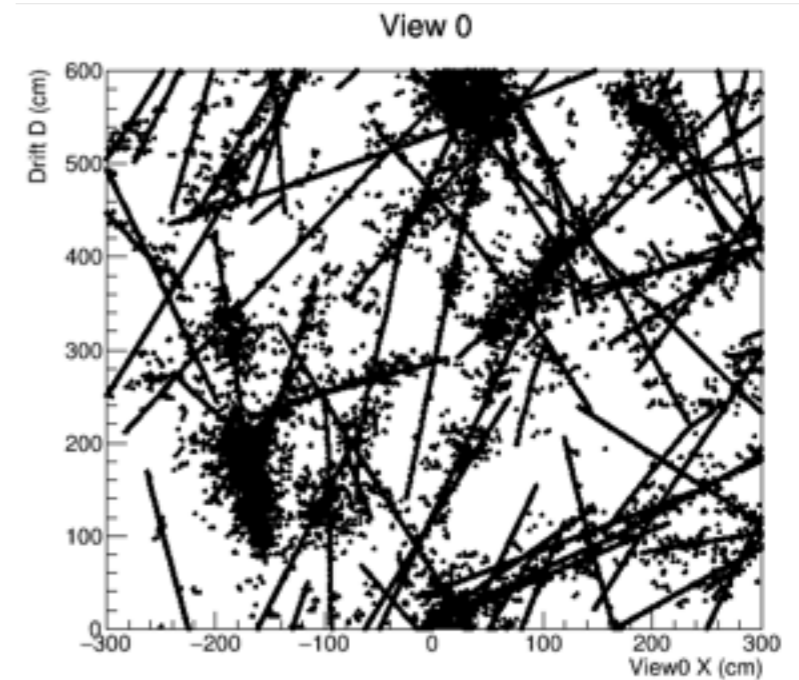
8ms beam halo
~34 000 hits found per view



8 ms cosmic rays
~27 000 hits found per view



8 ms cosmic rays + beam halo
~58 000 hits found per view



CPU time for reconstruction (in the 6x6x6)

NB : Time given here are not absolute number, but should be considered as trends

Time in seconds	4 GeV through going μ [averaged for 100 events]	8 ms CR [averaged for 50 events]	8 ms Beam Halo [averaged for 50 events]	8 ms CR + beam halo [averaged for 20 events]
Mean number of hits per view	~900 hits	~27 000 hits	~34 000 hits	~58 000 hits
Hit Finding*	8.6 s efficiency = 91%	13.2 s efficiency = 93%	12.8 s efficiency = 92%	13.9 s efficiency = 82%
Hit Finding + Clustering	8.4 s efficiency = 95%	32.1 s efficiency = 97%	52.1 s efficiency = 97%	87.7 s efficiency = 97%
Hit Finding + Clustering + TrackBuilderMTC	8.9 s efficiency = 93%	76.1 s efficiency = 99%	71.5 s efficiency = 98%	349.9 s efficiency = 98%
Hit Finding + ClusFilter	8.3 s efficiency = 88%	19.6 s efficiency = 95%	20.1 s efficiency = 95%	31.5 s efficiency = 92%

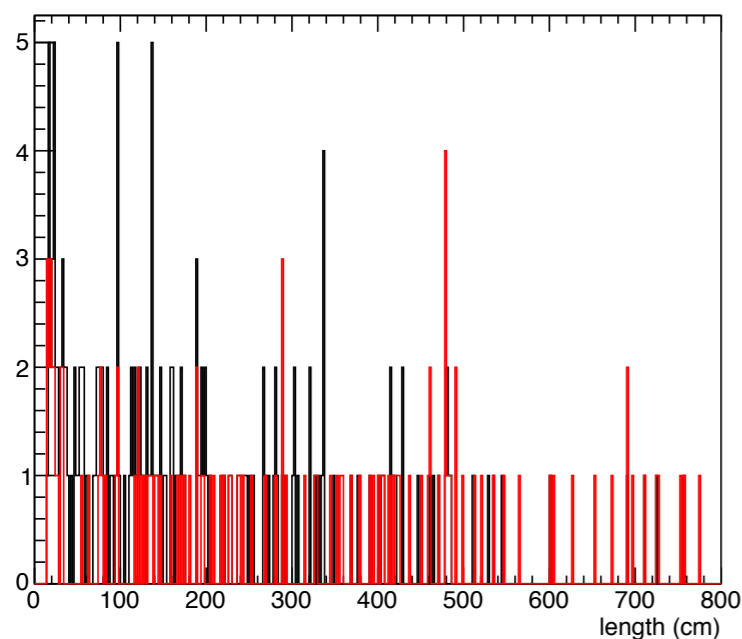
* voxel information (matching hit mc truth) commented due to memory mapping problem when occupancy is getting too big (to be investigated)

efficiency is cpu efficiency

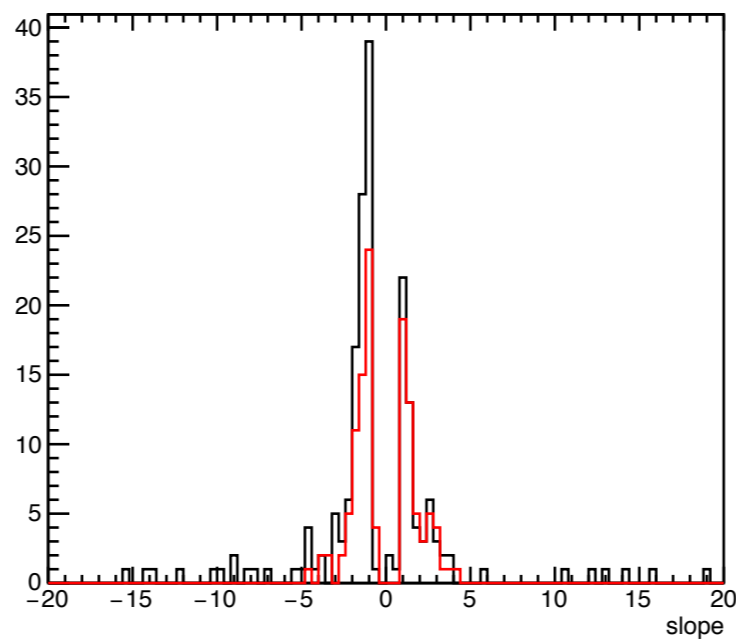
Comparison of track reconstruction - through going μ

TrackBuilderMTC
ClusFiltering

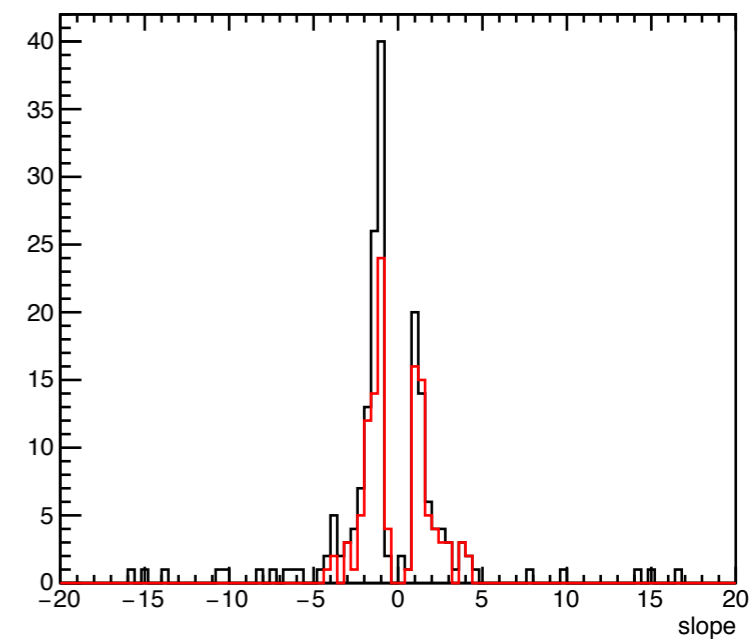
Track Length (View0)



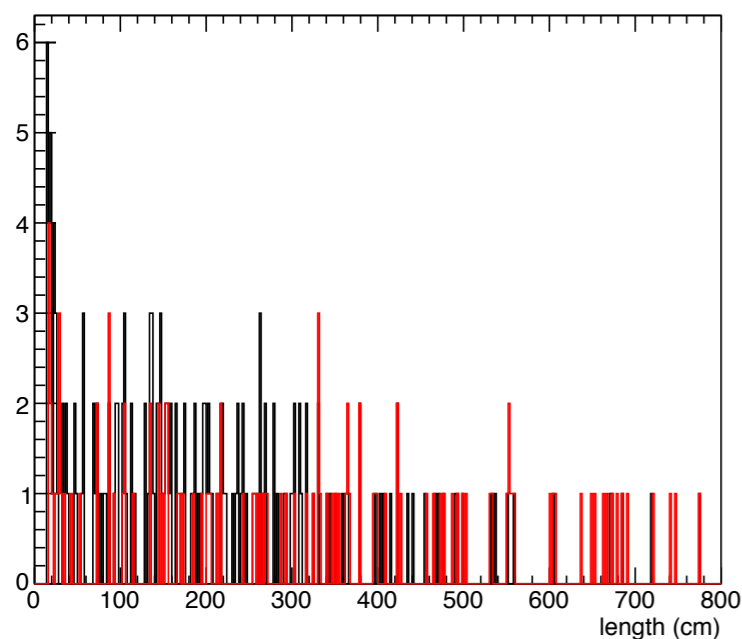
Initial Slope (View0)



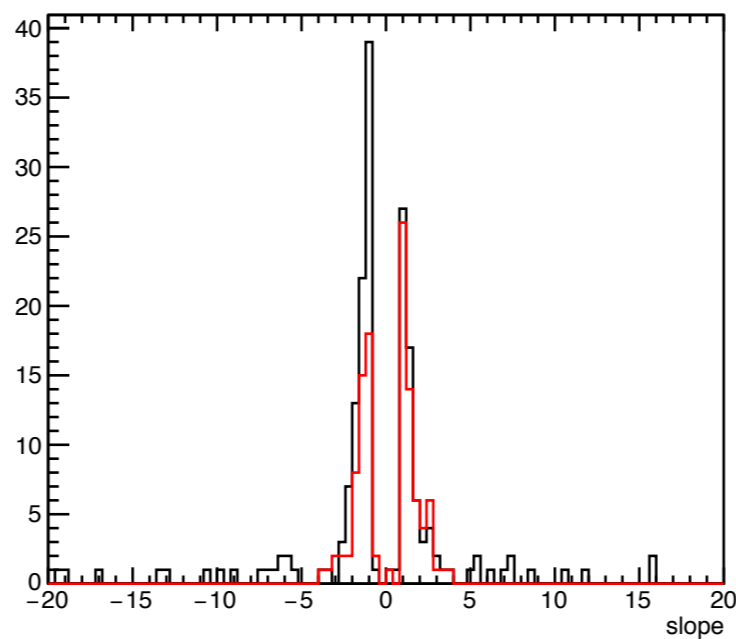
Final Slope (View0)



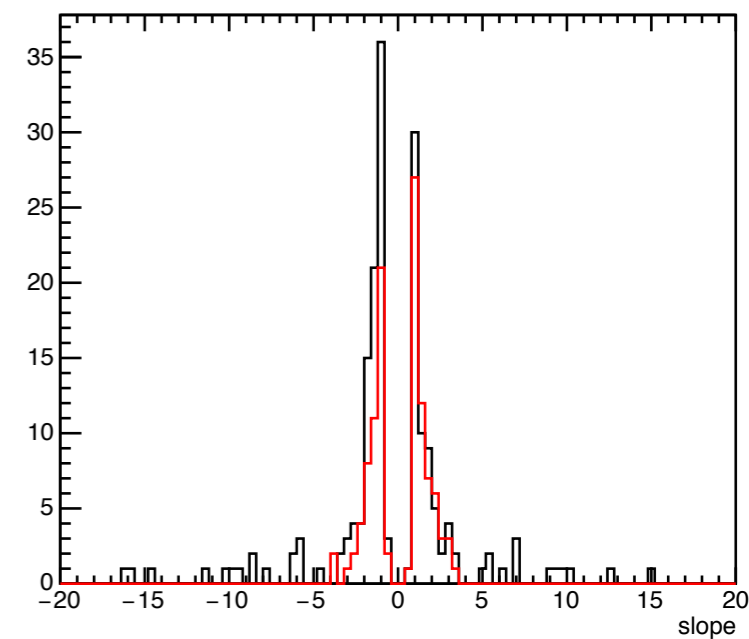
Track Length (View1)



Initial Slope (View1)



Final Slope (View1)

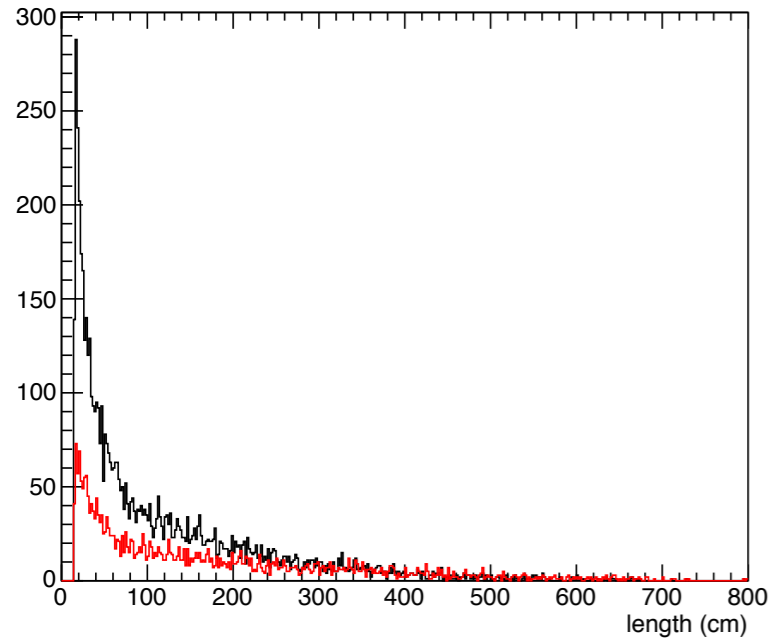


For clarity, only tracks longer than 15cm are considered here

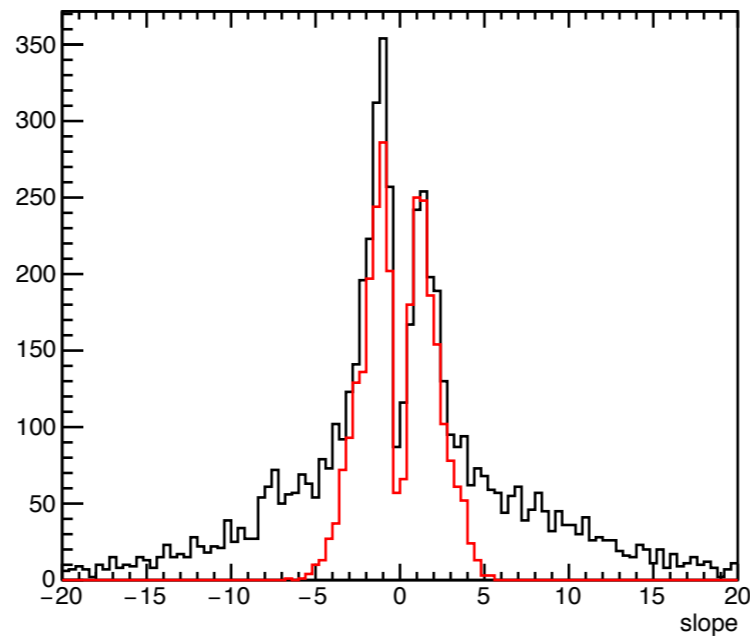
Comparison of track reconstruction - 8ms CR

TrackBuilderMTC
ClusFiltering

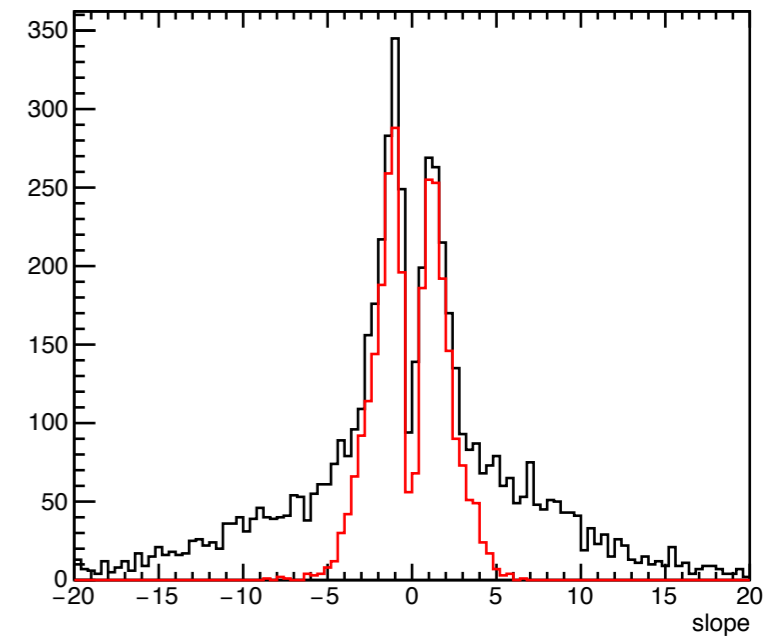
Track Length (View0)



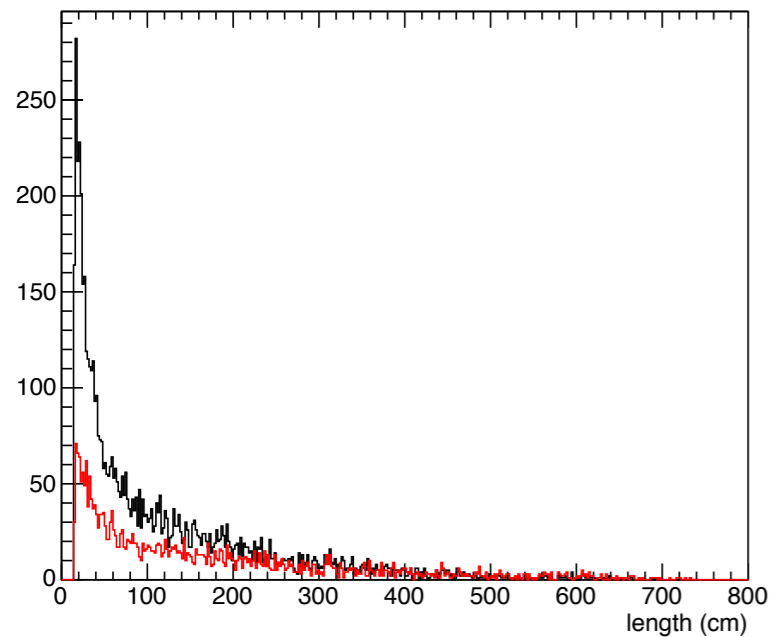
Initial Slope (View0)



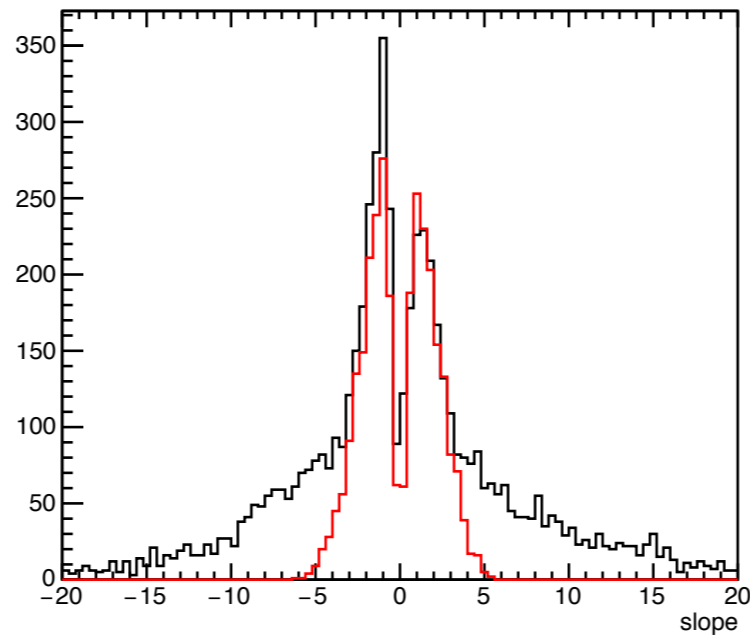
Final Slope (View0)



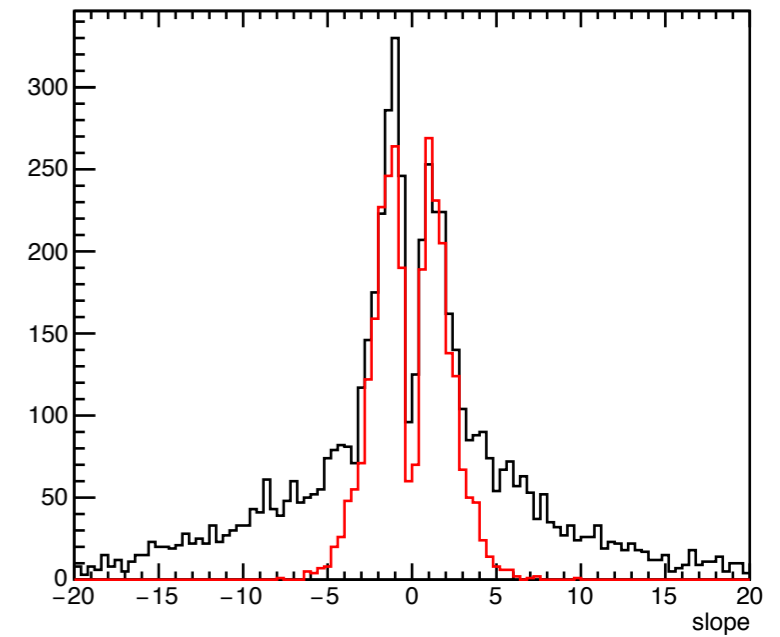
Track Length (View1)



Initial Slope (View1)



Final Slope (View1)

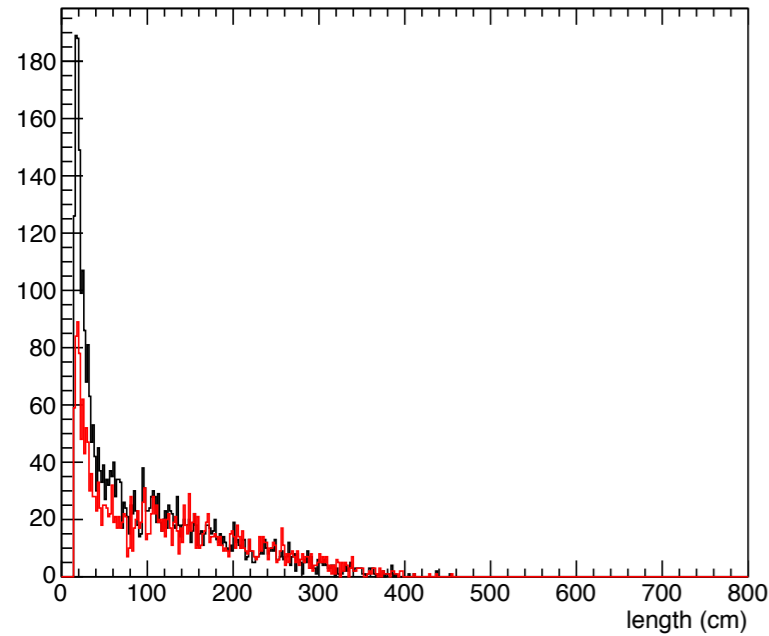


For clarity, only tracks longer than 15cm are considered here

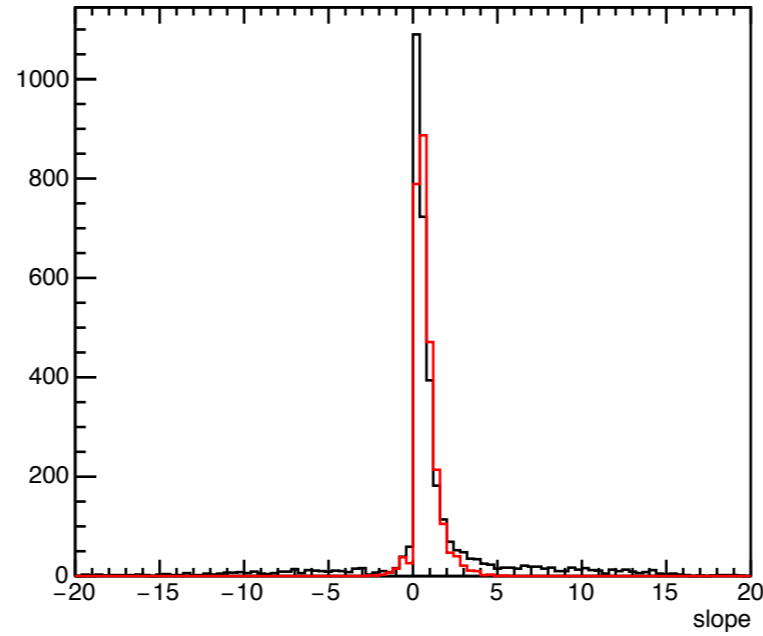
Comparison of track reconstruction - beam halo

TrackBuilderMTC
ClusFiltering

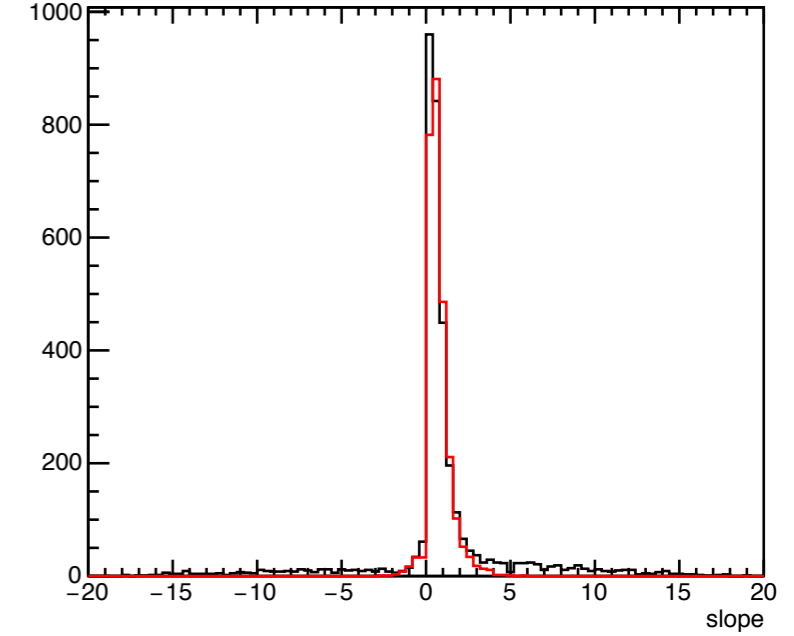
Track Length (View0)



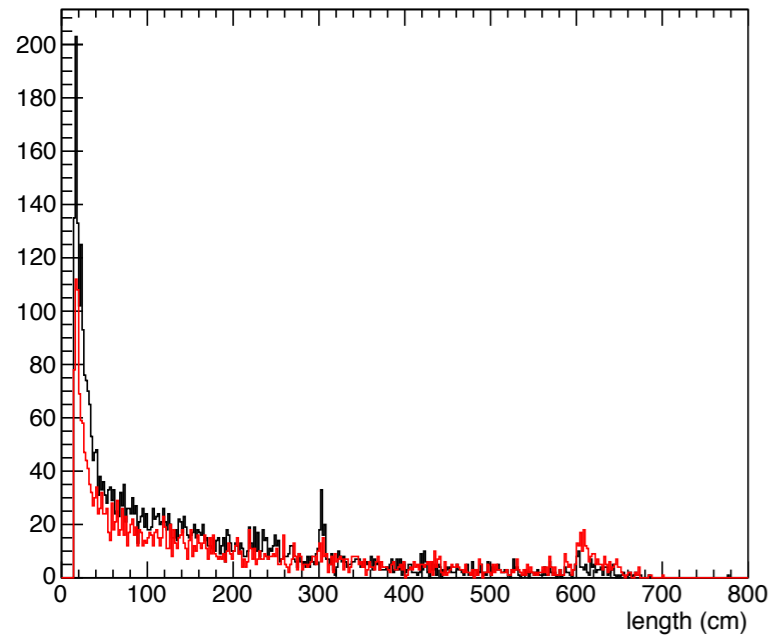
Initial Slope (View0)



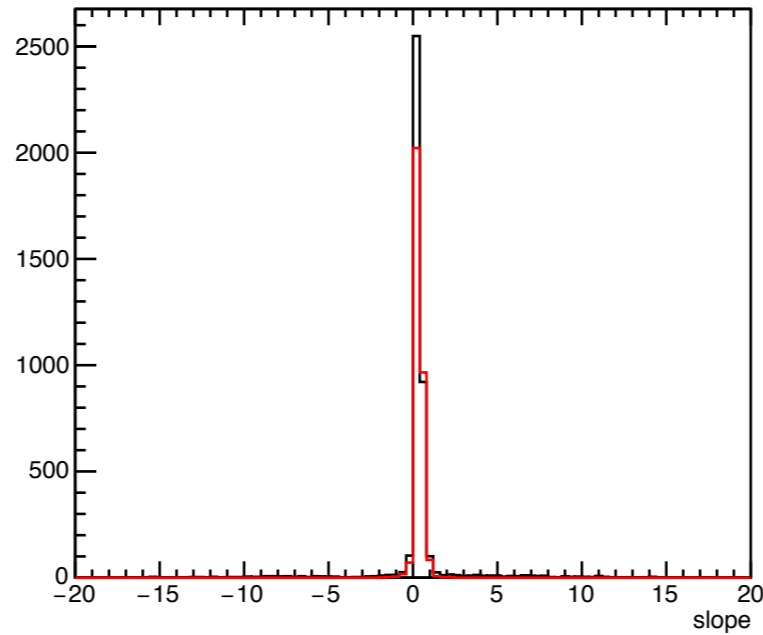
Final Slope (View0)



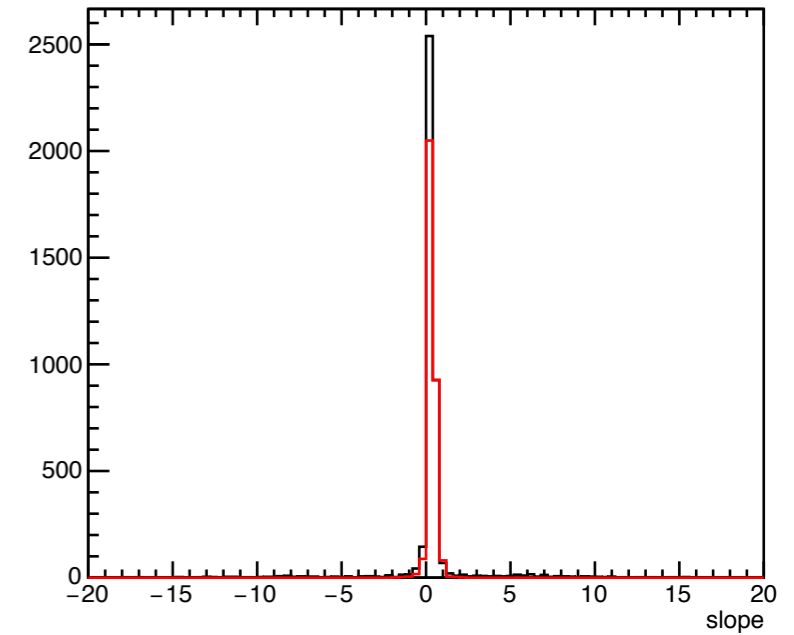
Track Length (View1)



Initial Slope (View1)



Final Slope (View1)

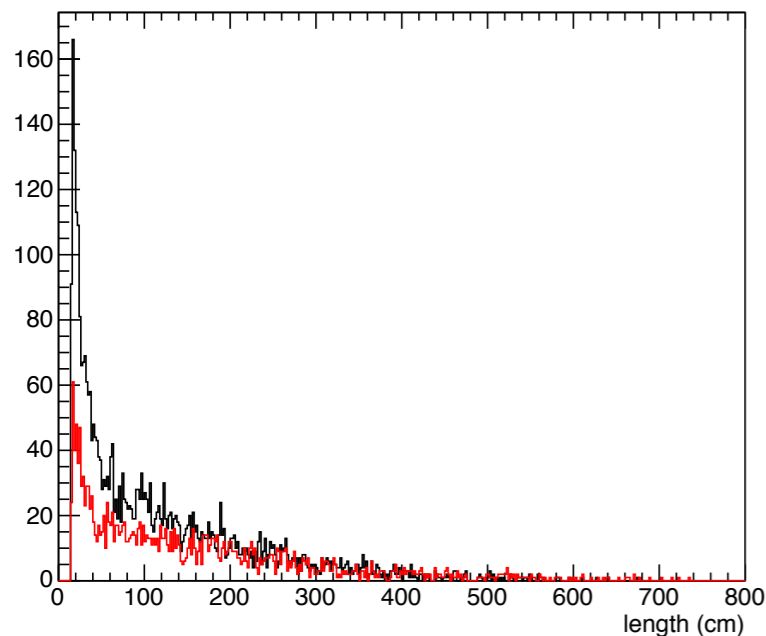


For clarity, only tracks longer than 15cm are considered here

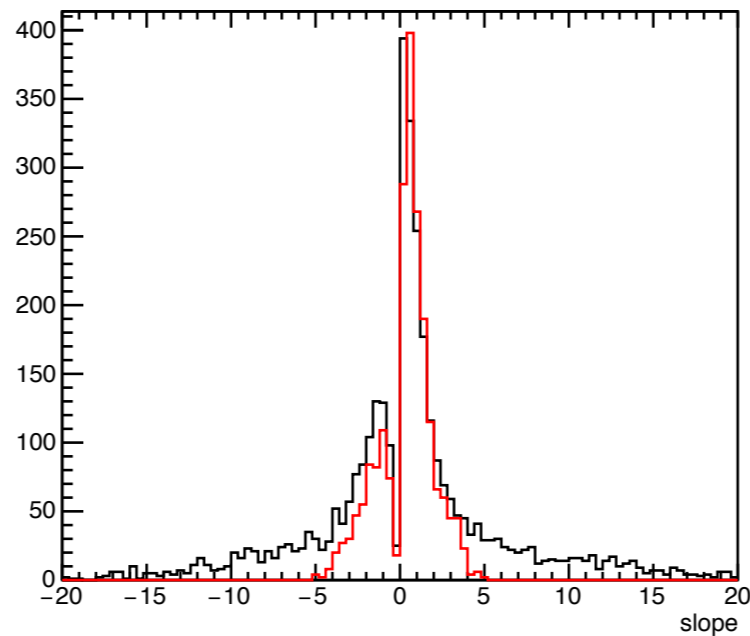
Comparison of track reconstruction - 8ms CR + halo

TrackBuilderMTC
ClusFiltering

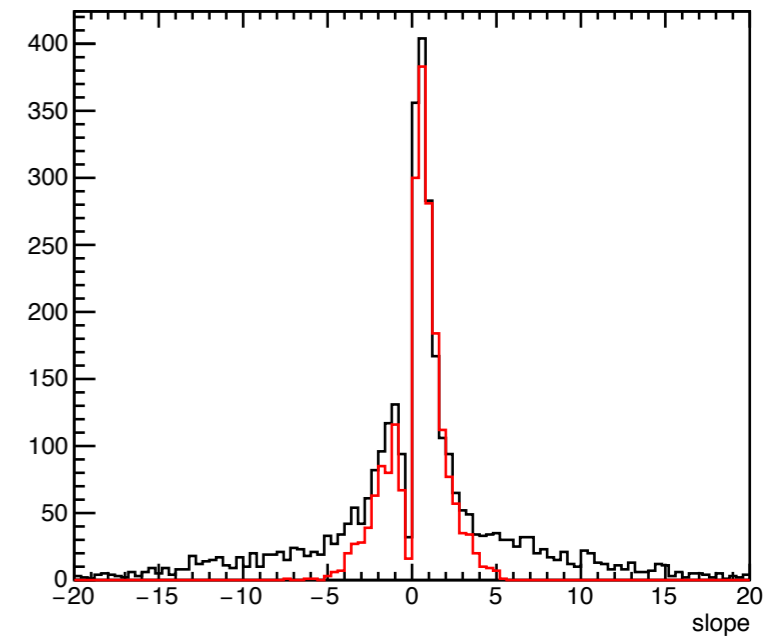
Track Length (View0)



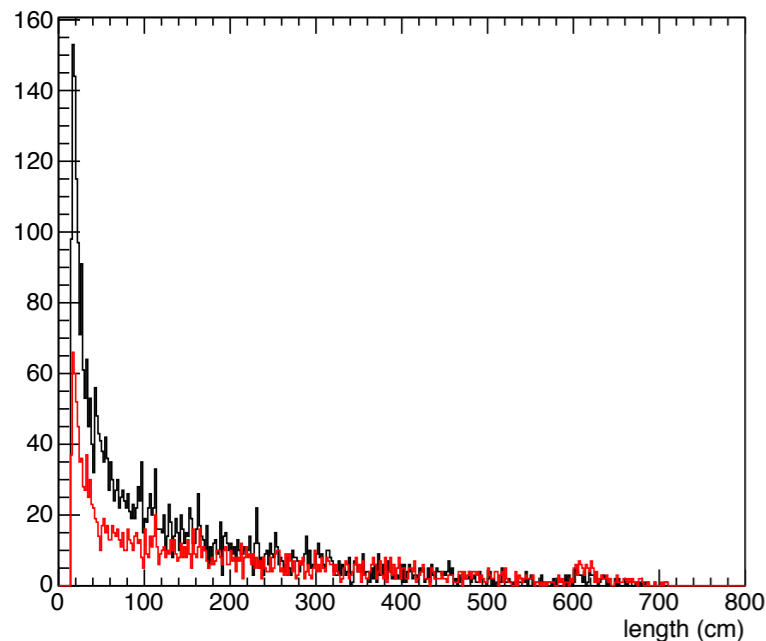
Initial Slope (View0)



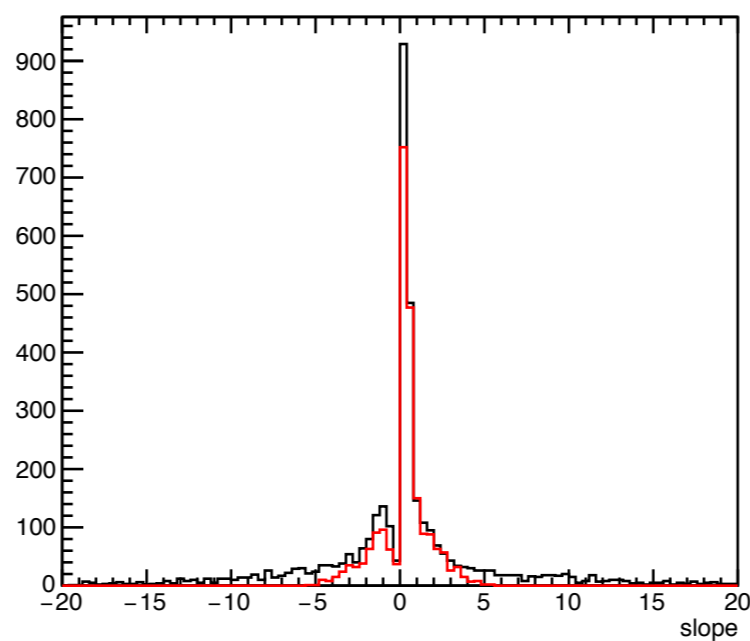
Final Slope (View0)



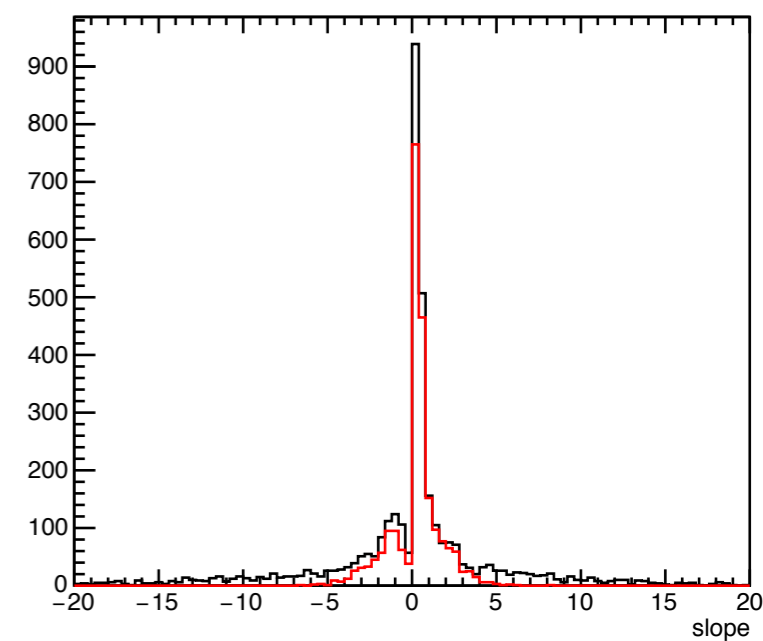
Track Length (View1)



Initial Slope (View1)



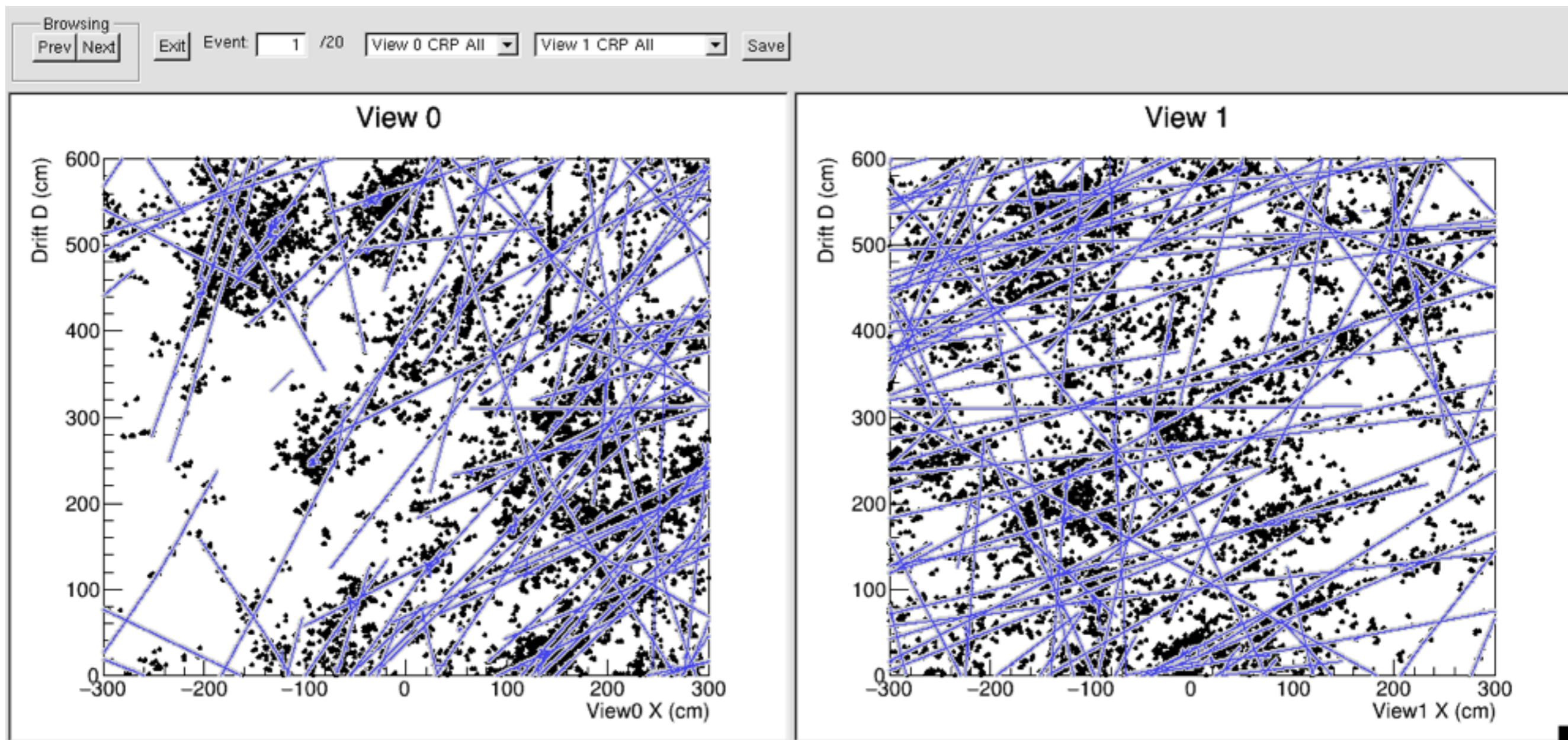
Final Slope (View1)



For clarity, only tracks longer than 15cm are considered here

Reconstruction comparison - CR + beam Halo

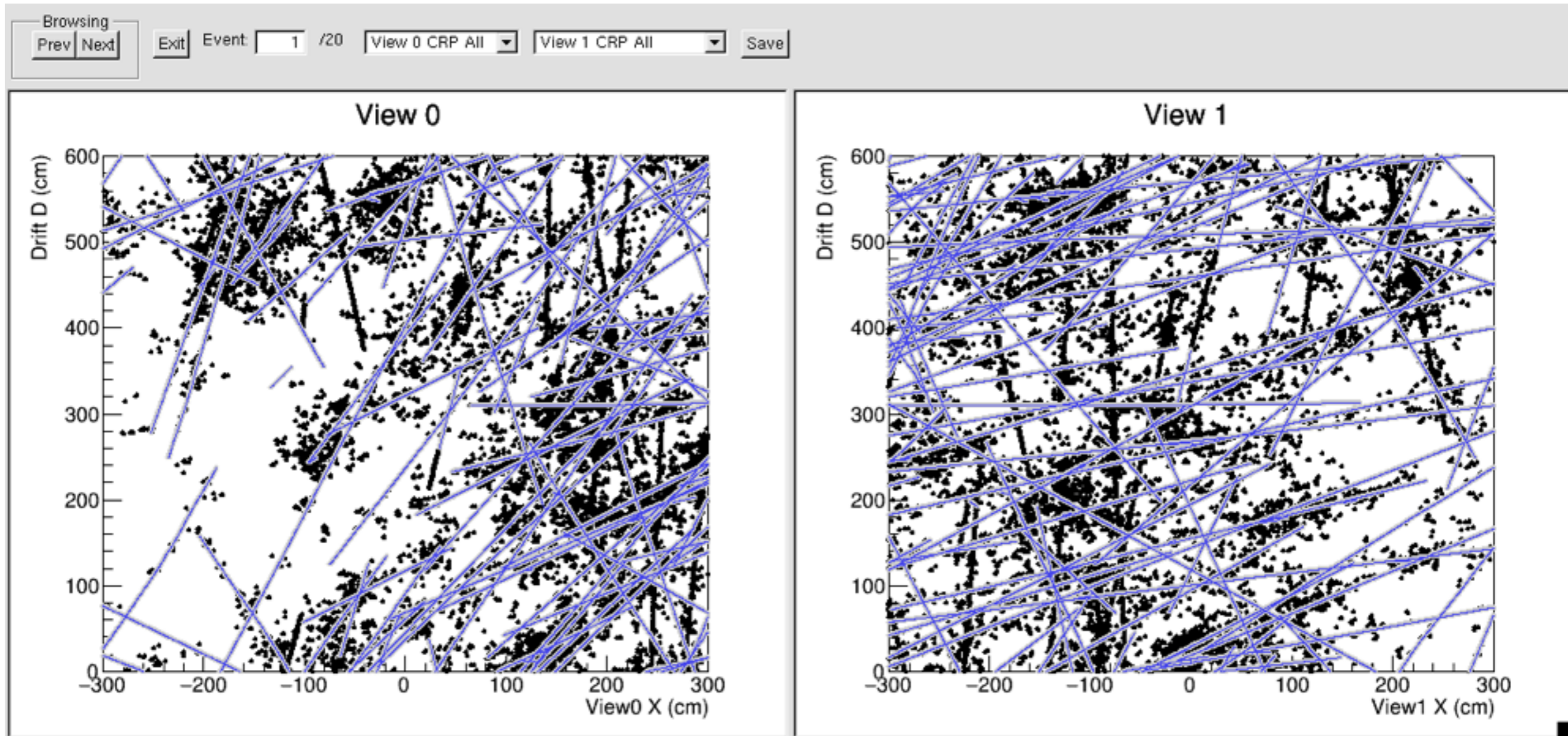
TrackBuilderMTC



Hits attached to a track
Track Path
Unmatched hits

Reconstruction comparison - CR + beam Halo

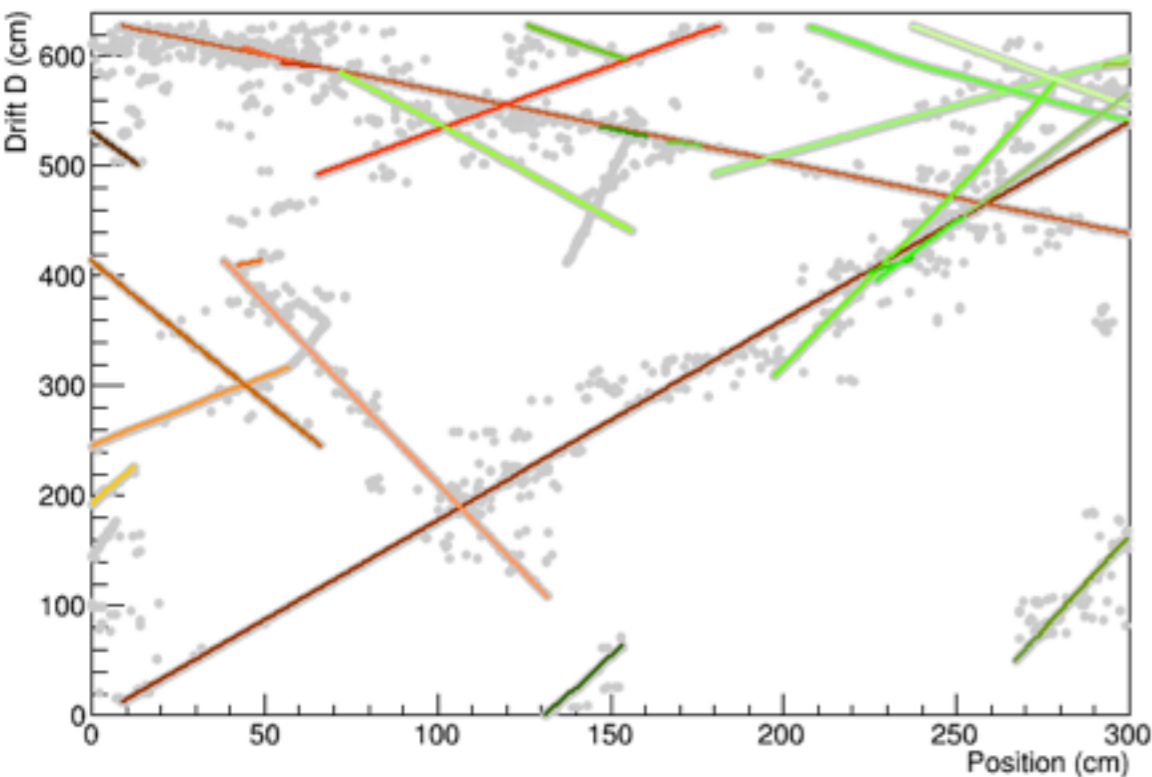
ClusFilter



Hits attached to a track
Track Path
Unmatched hits

Alternative CR Track finder - δ rays and vertical tracks

Tracks found (one CRM)

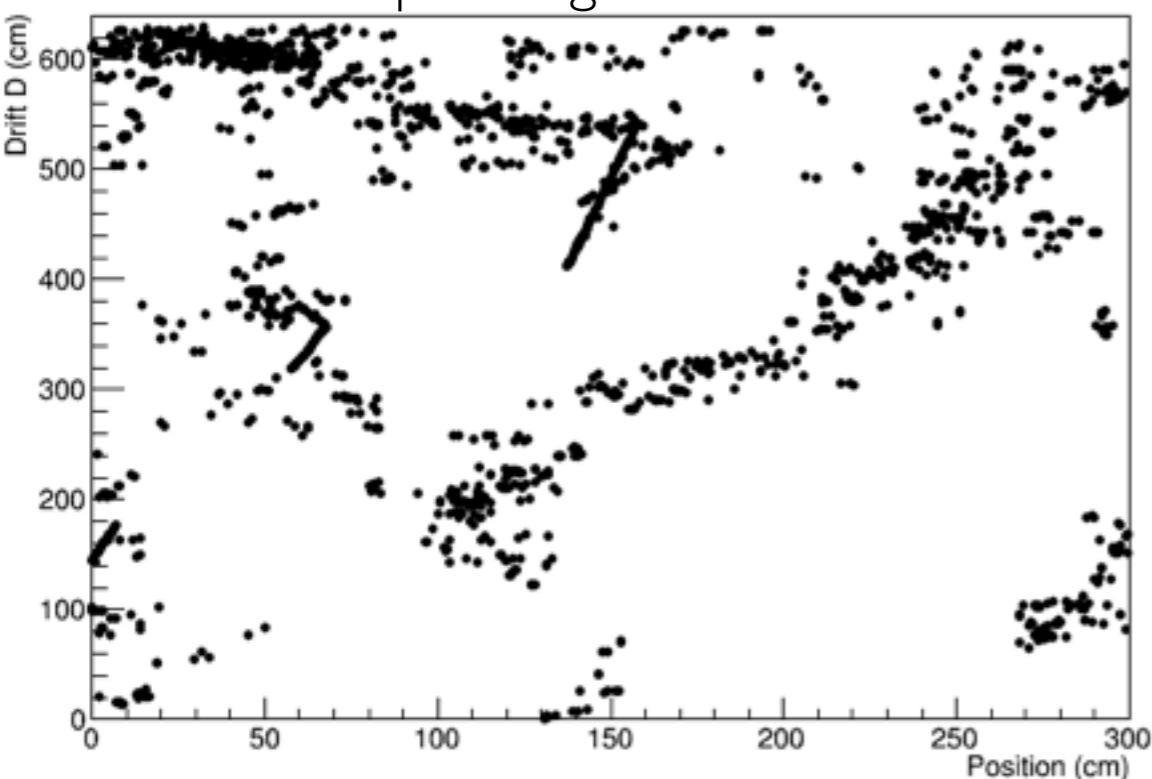


δ -rays finder :

Loop on hits attached to tracks and search for nearby unmatched hits using (very tight) NNClustering method.

If the build cluster is small enough [default is less than 15 hits] consider it as a delta ray belonging to the track

Corresponding unmatched hits




Vertical track finder :

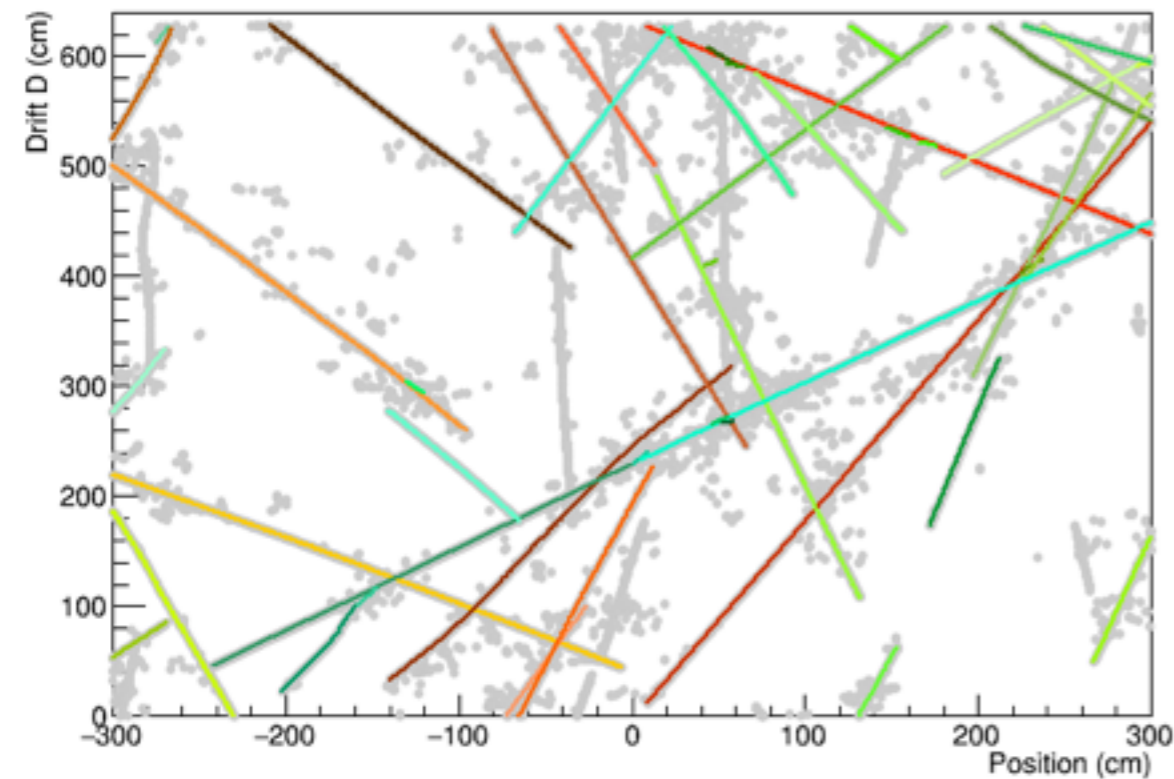
From the unmatched set of hits, try to build clusters by rotating the image ($x \rightarrow y$)

Then if a large cluster is found, try to filter it using the same algorithm, keeping in mind that the image is rotated. (i.e. try to find a line with equation $x = a*y + x_0$)

Alternative CR Track finder - δ rays and vertical tracks

 : new tracks

46 tracks found after merging



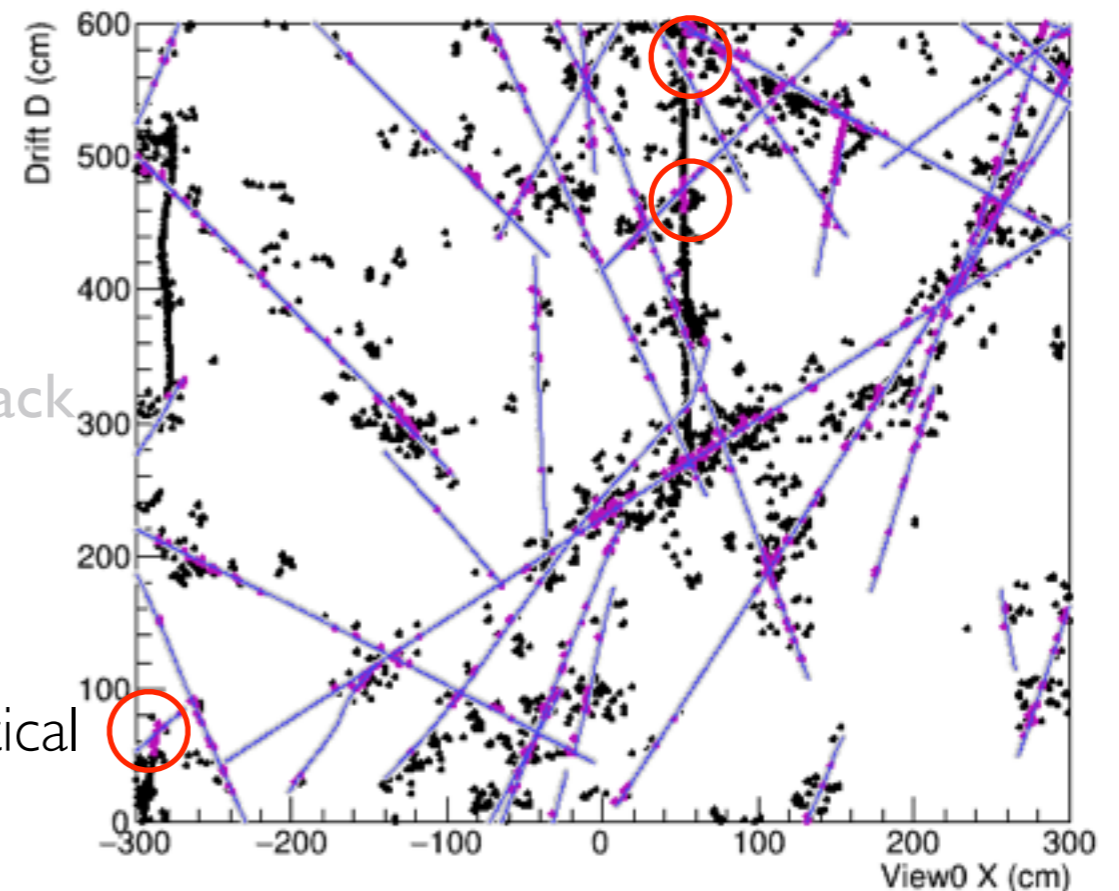
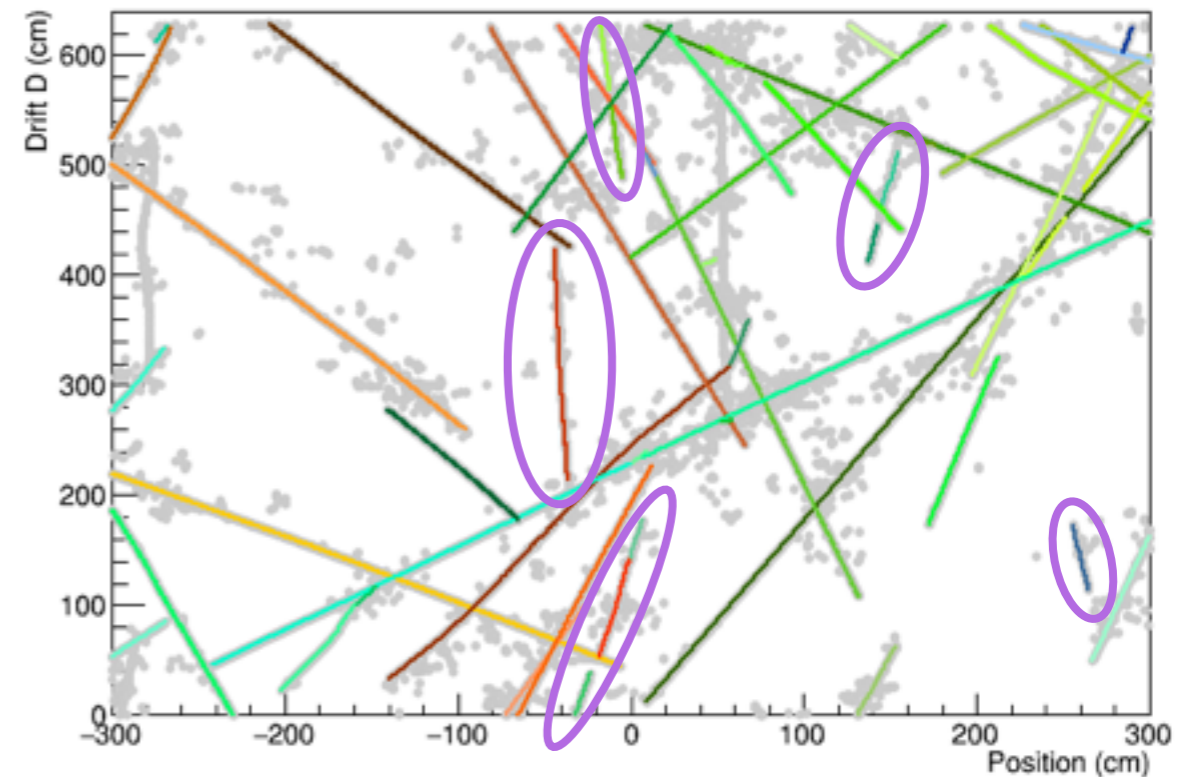
Hits attached to a track


Track Path

Delta Rays

Unmatched hits

55 tracks found when searching for vertical tracks

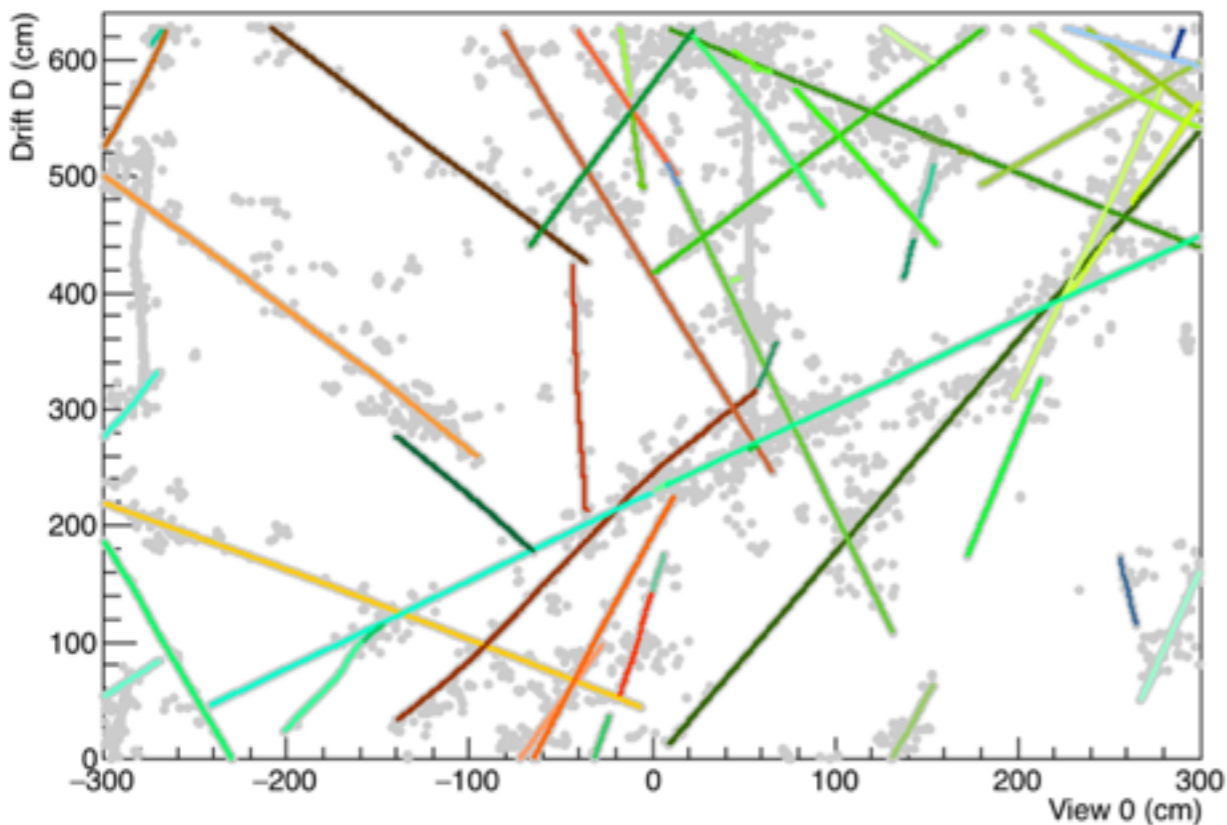


 : Problematic case, when hits belonging to a vertical tracks are misaligned as a delta-rays

Reconstruction with Space Charge Effect

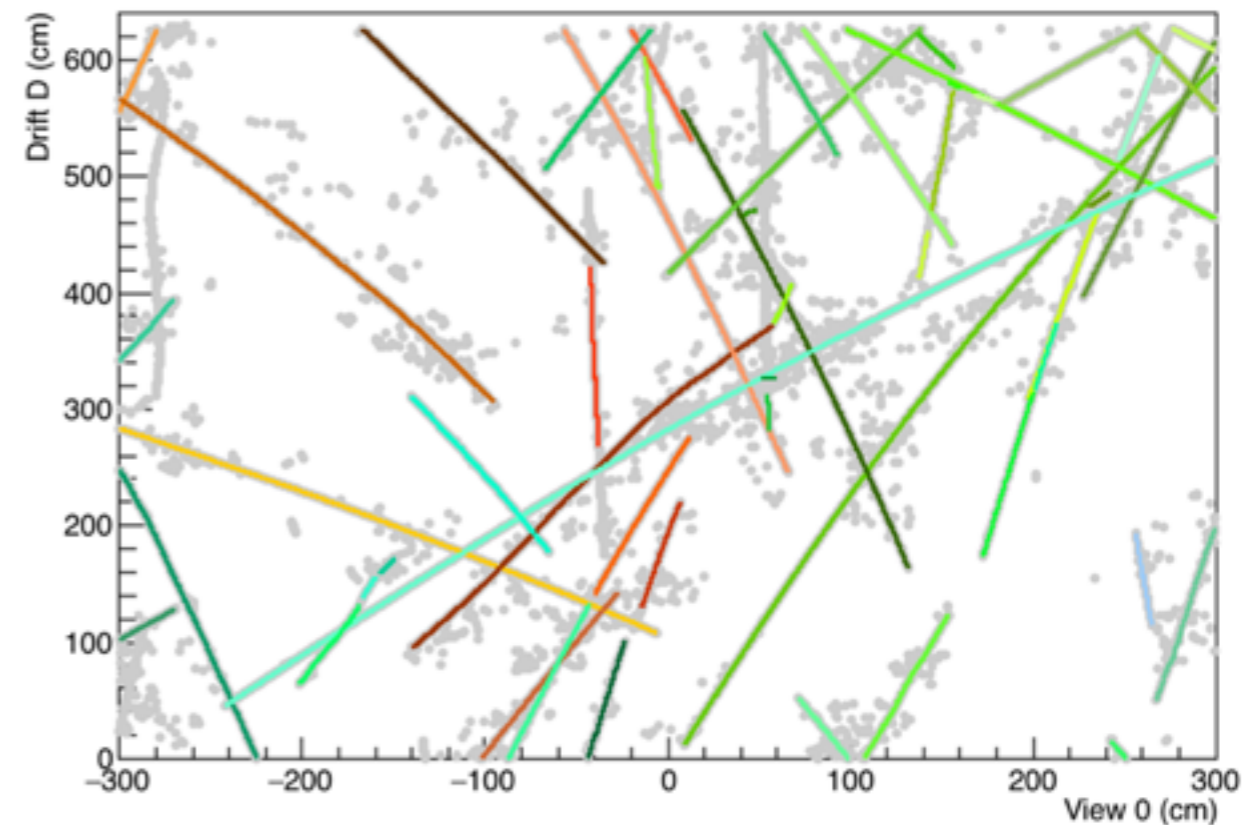
One line color = one track found

8 ms CR, uniform field



55 tracks found

8 ms CR, field map with 10% IBF



52 tracks found

→ The code seems to handle track distortions due to space charge effect

Recotask parameters for ClusFilter

To be added in the recotask config file

```
PARA_CLUSFILTER [20, 3., 2., 10, 0.256, 1., 15] ← Parameters for ClusFilter code
```

```
#To use the ClusFilter algorithm:
```

```
#parameters:
```

```
# 1: Min nb of point to make a track
```

```
# 2: Distance in x or y to search for new hit (in cm)
```

```
# 3: Distance in z to search for new hit (in cm)
```

```
# 4: Chi2 cut
```

```
# 5: uncertainty on hit position along drift (in cm)
```

```
# 6: pbeta guess to compute MS error (the lower the more conservative) (in GeV)
```

```
# 7: Nb of point for d-ray
```

```
# -----
```

```
# Parameter for joining tracks between 3x3m2 modules in 6x6x6
```

```
# - max distance in cm between endpoints
```

```
# - number of std on slope error to look for collinear tracks
```

```
PARA_TRKJOIN [5.0, 5.0]
```

← Parameters for Track Merging (common for all algorithms)

[ClusFilter code is not yet committed though!]

Ideas for future improvements

For example, for a CR events :

Hit Finding + ClusFilter : ~20 s

Hit Finding + ClusFilter + δ rays + vertical tracks finders : ~64 s

↳ Huge amount of time spent to find δ -rays and vertical track (due to multiple loops on the unmatched hits) - to be improved !

For track merging within CRM, one could add some spatial constrains to merge only "mergable" tracks. This can become a problem when there is beam halo, as many collinear tracks crosses the detector.

Develop 3D tracks, by merging in between views. Simple criteria is comparing end points (in time) of tracks with CRM constraints, but more sophisticated technique can be found.

Conclusion and to-do list

- An alternative track reconstruction code has been developed, and show faster CPU performances wrt to other algorithms for comparable results.
- ClusFilter code is not good at finding vertical tracks, a first simple implementation tries to fix it although it needs further improvements.
- A detailed code reconstruction efficiency is needed (like nb of tracks found, slopes, track charge reconstructed, ...)
- 3D tracking is the next step.
- 3x1x1 track reconstruction performance is also needed
- Although ClusFilter is faster, ~30 s to reconstruct a single CR+beam halo event is still too long - to be further improved !