Charged Pions and Taus in MAIA

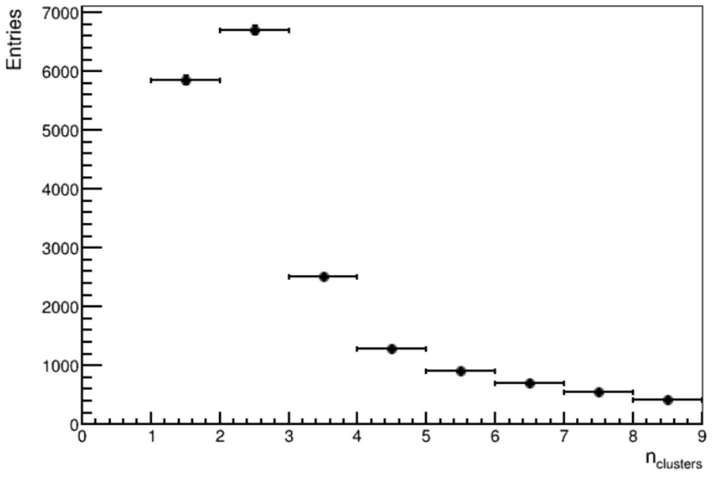
Sarah Demers, Ethan Martinez, and Gregory Penn

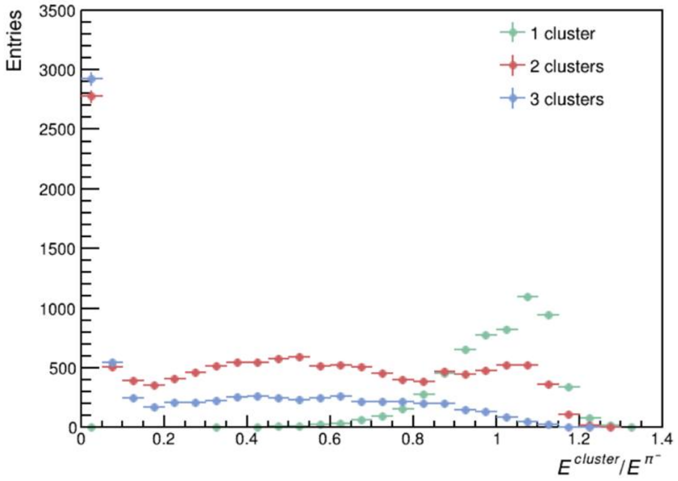
Yale University

Quick Reminder

- W/ pion gun: found that track “refit” greatly improved pion reconstruction efficiency (two presentations ago)

Si Track

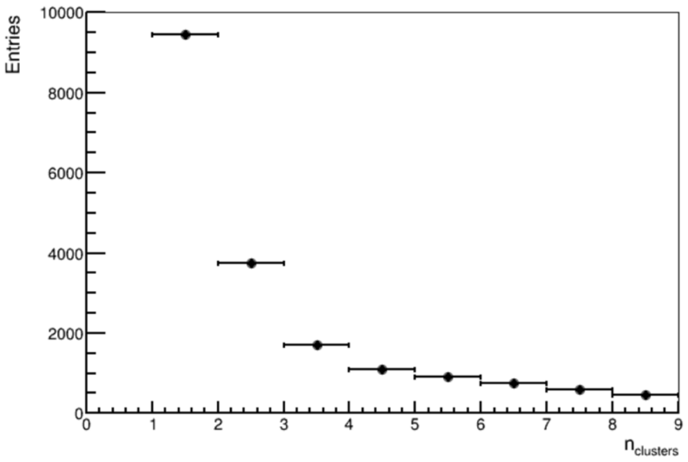


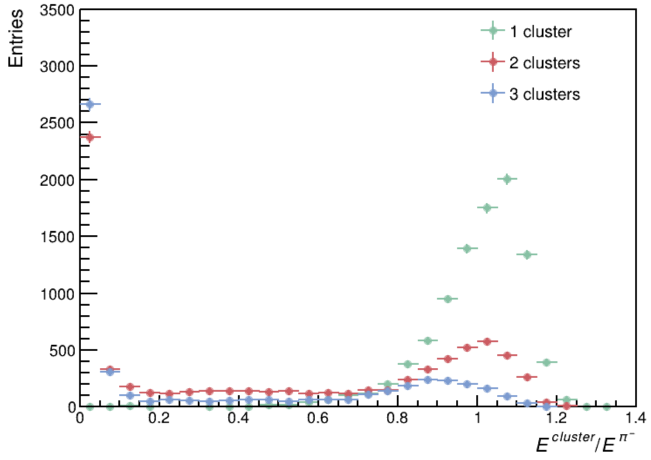


clusters less split up

one cluster more often captures all $\pi^{\pm}$ E

Si Track _Refitted

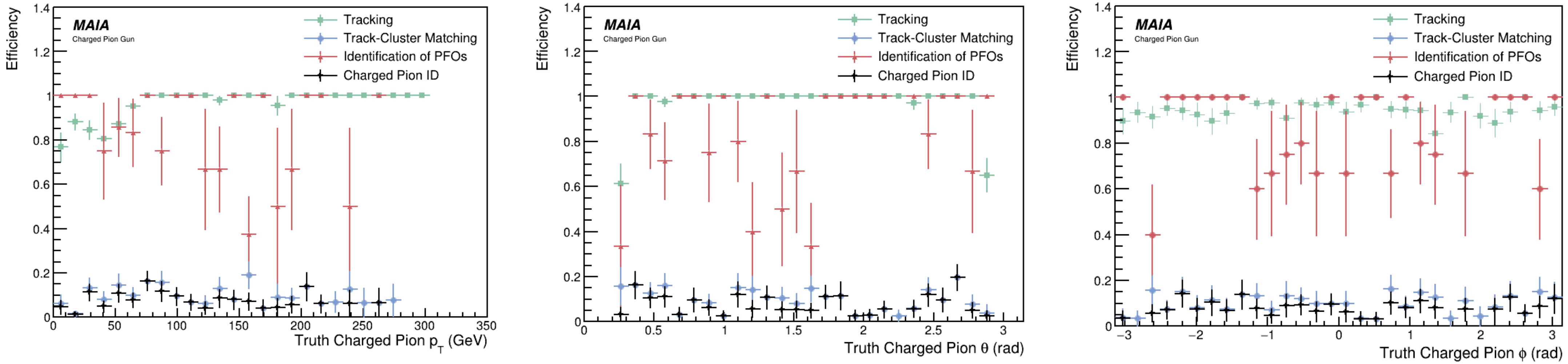




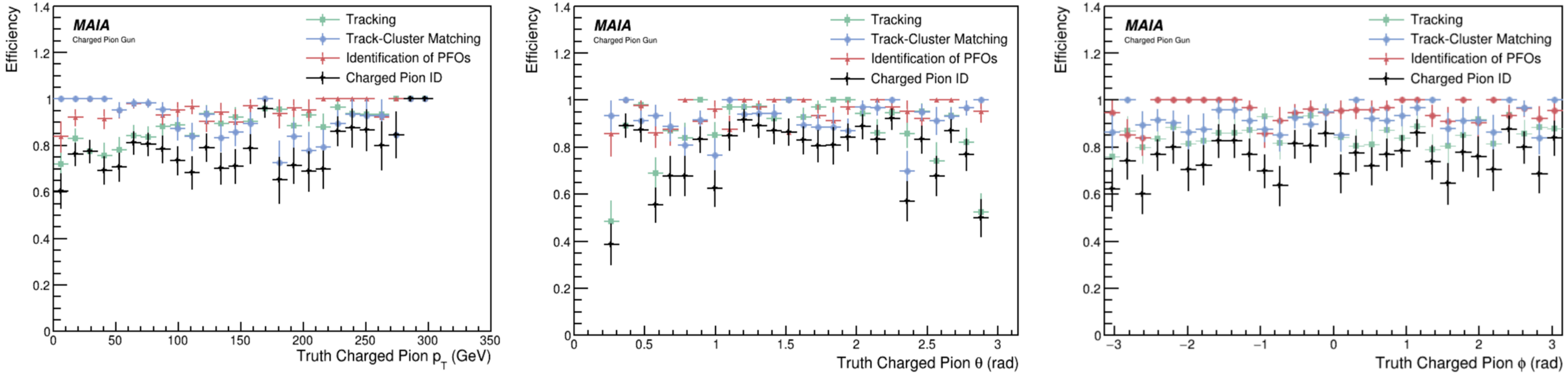
Quick Reminder

- This propagates down reconstruction in the form of drastically different **track-cluster matching** efficiency:

Si Track



Si Track _Refitted



Quick Recap

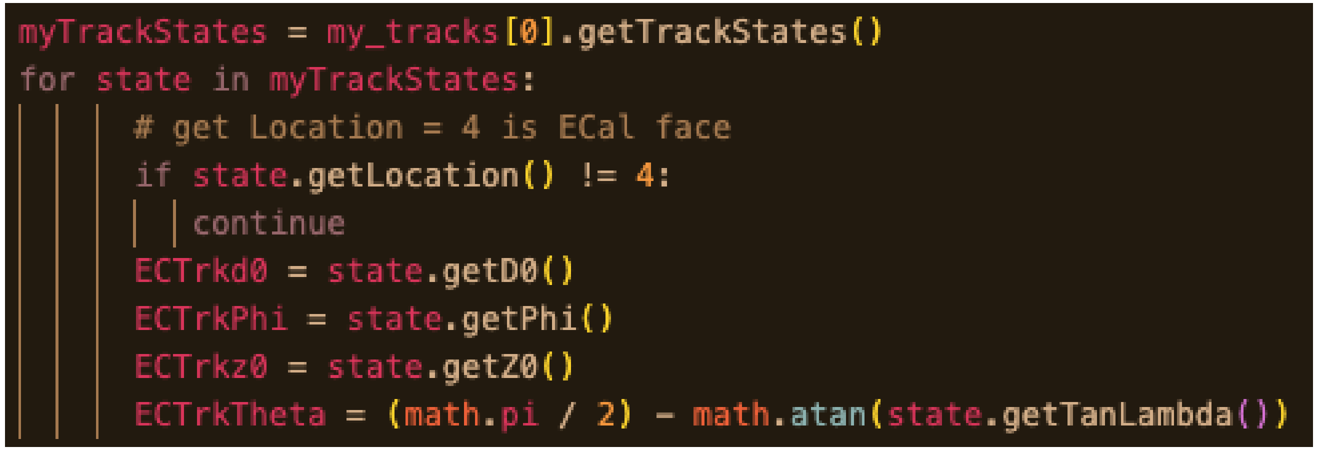
- Question we are still pursuing:
    - *“Why does one track container lead to better charged pion efficiency than the other?”*

- Last presentation covered some ground:
    - Understanding that tracking influences clustering
        - Clusters are seeded with track
        - Branching paths in clustering algorithm dependent on track and cluster are “consistency”
    - SiTracks and SiTracks_Refitted verified to be similar (without extrapolation)
    - Simplification in generated charged pions:
        - $E \subset (5, 300)$ GeV → constant pT = 50 GeV
    - Validation of tracking performance w/ paper results
        - Using muon gun samples w/ pT = 50 GeV
        - Looks like SiTracks container used in paper results (?)

- Questions covered in this presentation:
    - Are the track states @ ECal face similar between SiTracks and SiTracksRefitted?
    - Are there track requirements before it can be considered for track-cluster matching?

Track States @ ECal Face

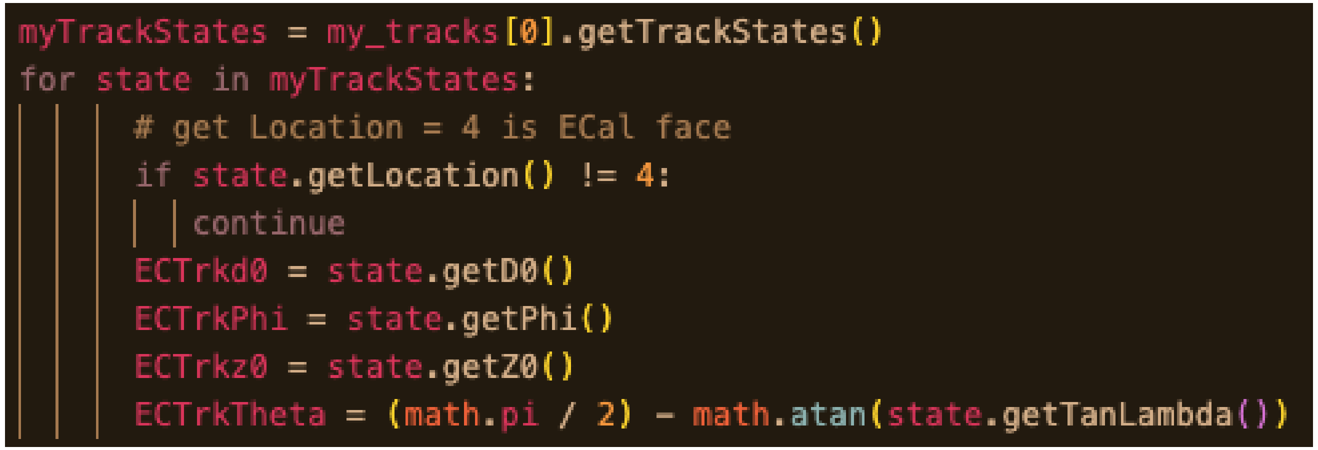
- Previous studies have shown that SiTracks and SiTracks_Refitted have similar states @ Ecal face
- Confirming this is crucial – different states @ Ecal face could explain our clustering differences
- Track states accessed according to class reference



```
myTrackStates = my_tracks[0].getTrackStates()
for state in myTrackStates:
      # get Location = 4 is ECal face
      if state.getLocation() != 4:
         continue
      ECTrkd0 = state.getD0()
      ECTrkPhi = state.getPhi()
      ECTrkz0 = state.getZ0()
      ECTrkTheta = (math.pi / 2) - math.atan(state.getTanLambda())
```

Track States @ ECal Face

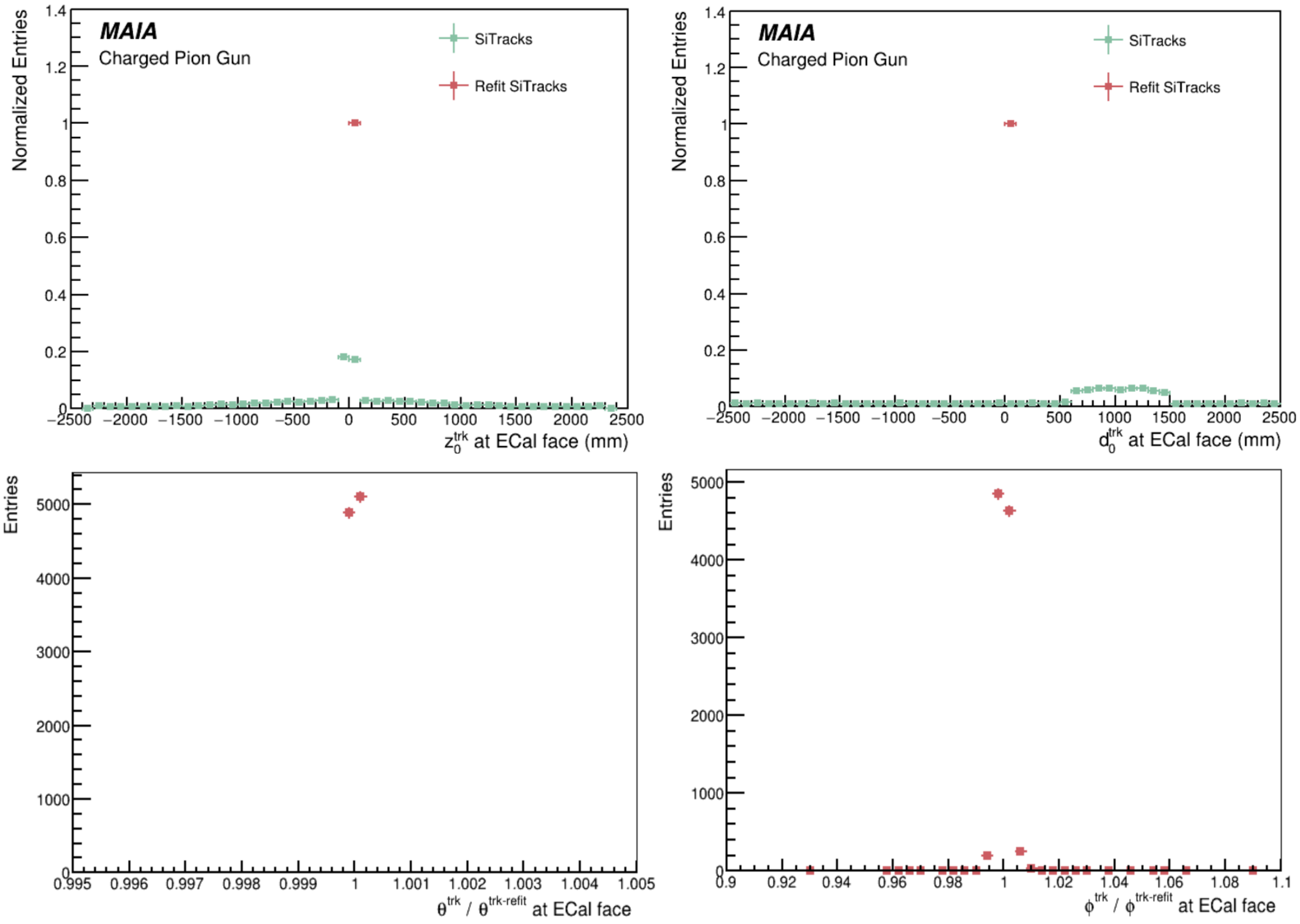
- Previous studies have shown that SiTracks and SiTracks_Refitted have similar states @ Ecal face
- Confirming this is crucial – different states @ Ecal face could explain our clustering differences
- Track states accessed according to class reference



```
myTrackStates = my_tracks[0].getTrackStates()
for state in myTrackStates:
      # get Location = 4 is ECal face
      if state.getLocation() != 4:
         continue
      ECTrkd0 = state.getD0()
      ECTrkPhi = state.getPhi()
      ECTrkz0 = state.getZ0()
      ECTrkTheta = (math.pi / 2) - math.atan(state.getTanLambda())
```

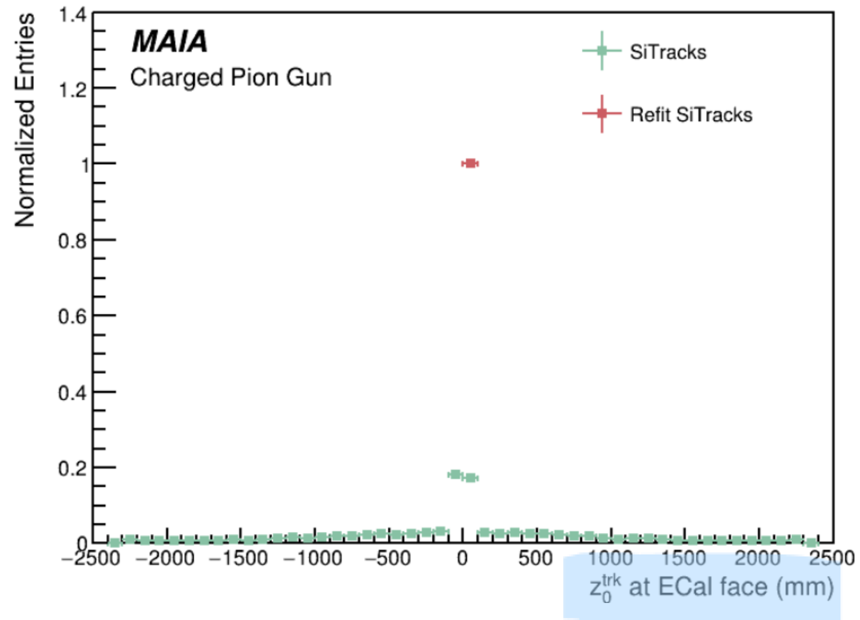
basically all of the information I have!

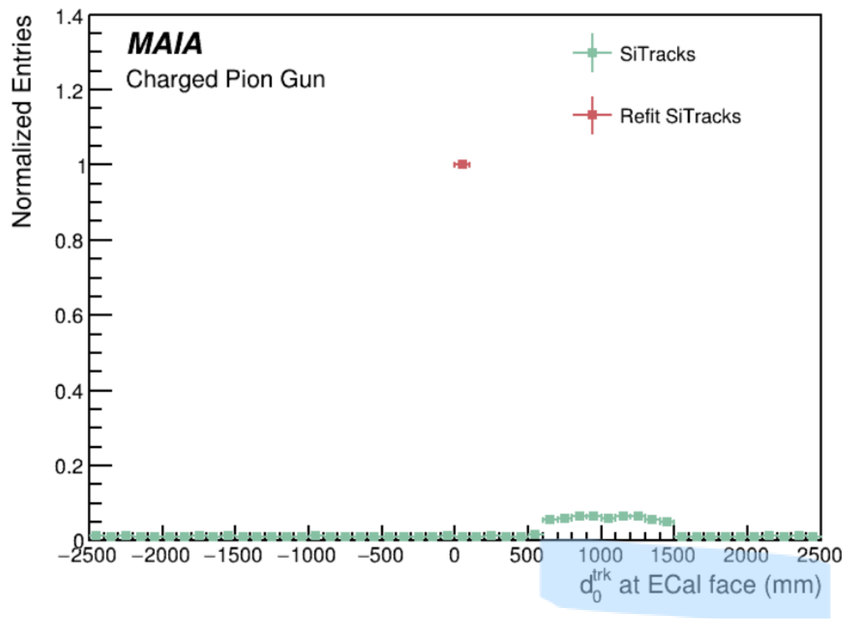
Track States @ ECal Face



So much to break down here...

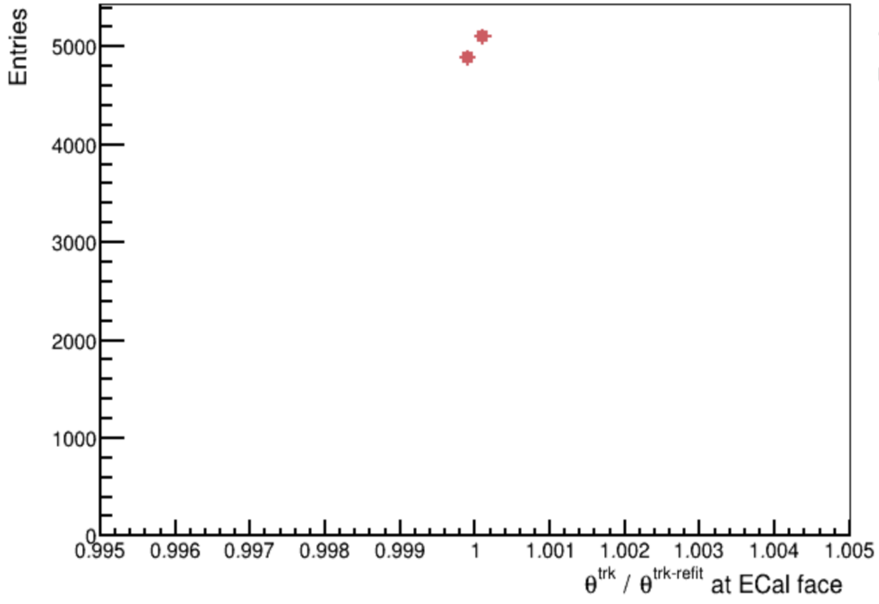
Track States @ ECal Face

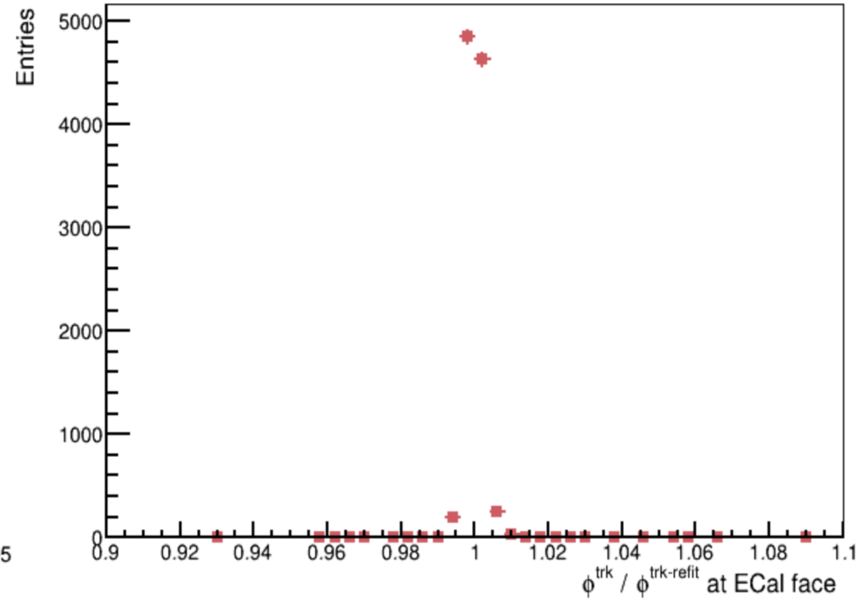




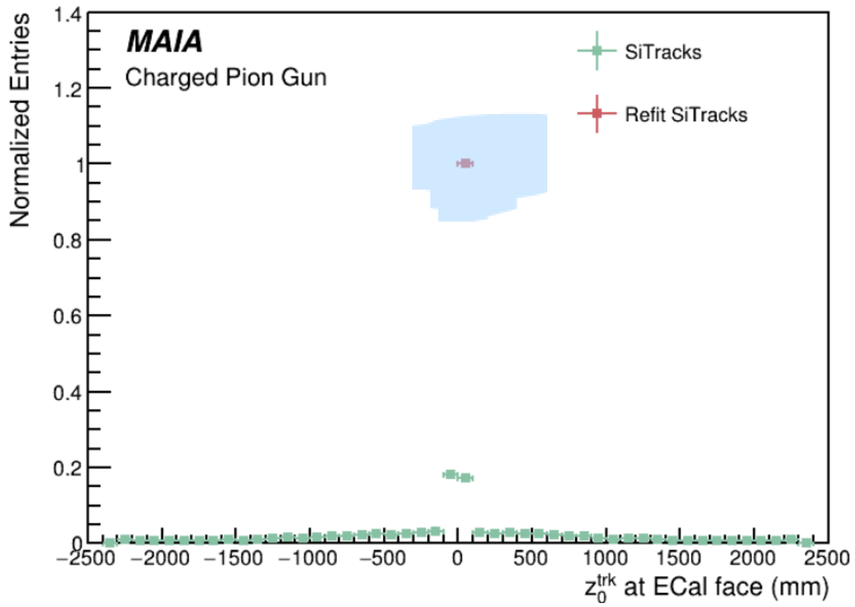
subtlety:

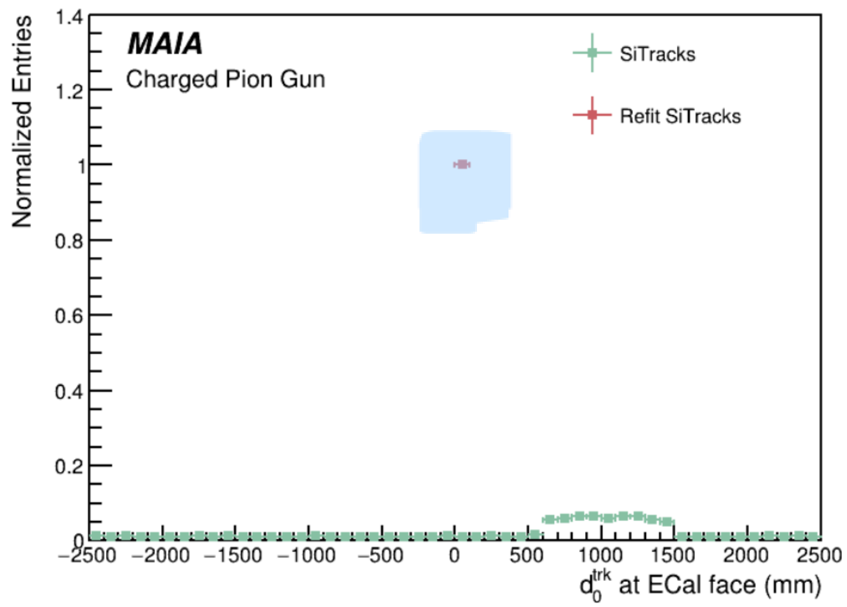
what does an impact parameter @ face of ECal mean?

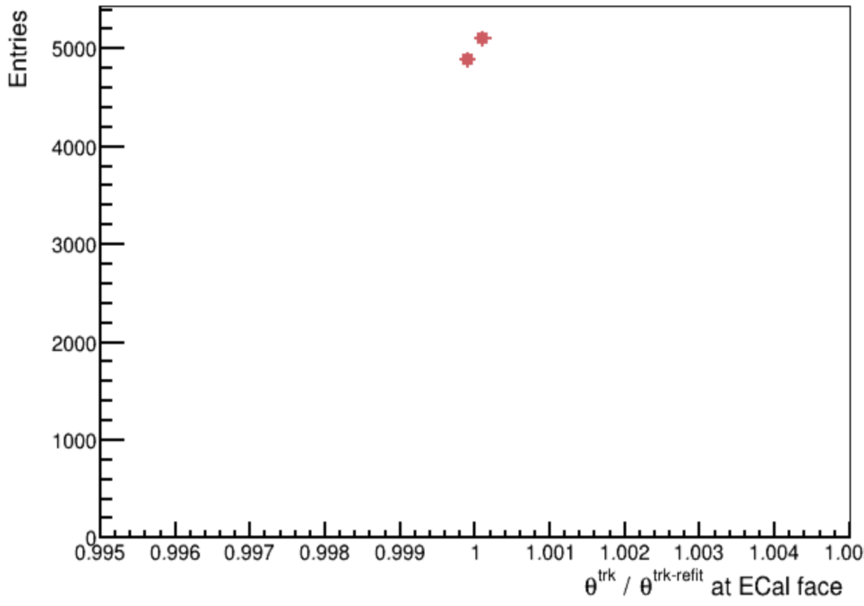


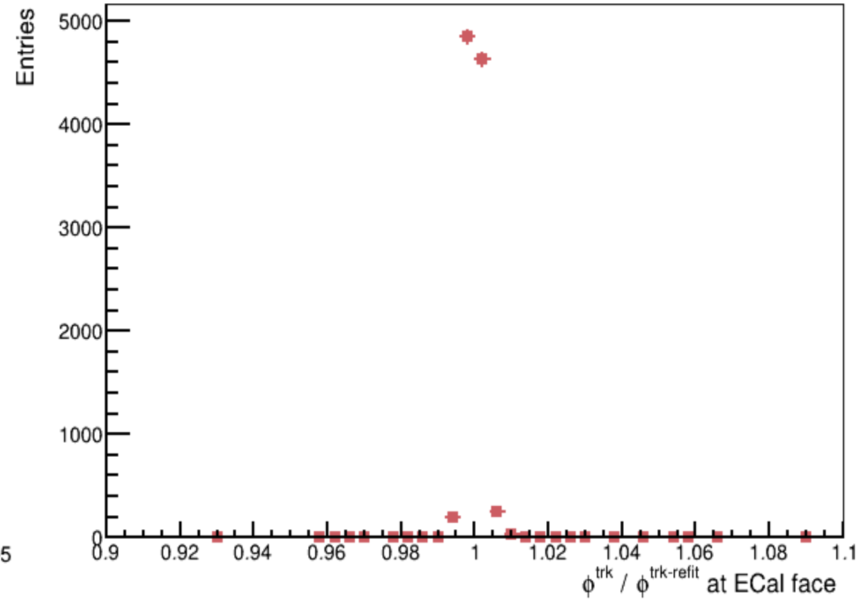


Track States @ ECal Face



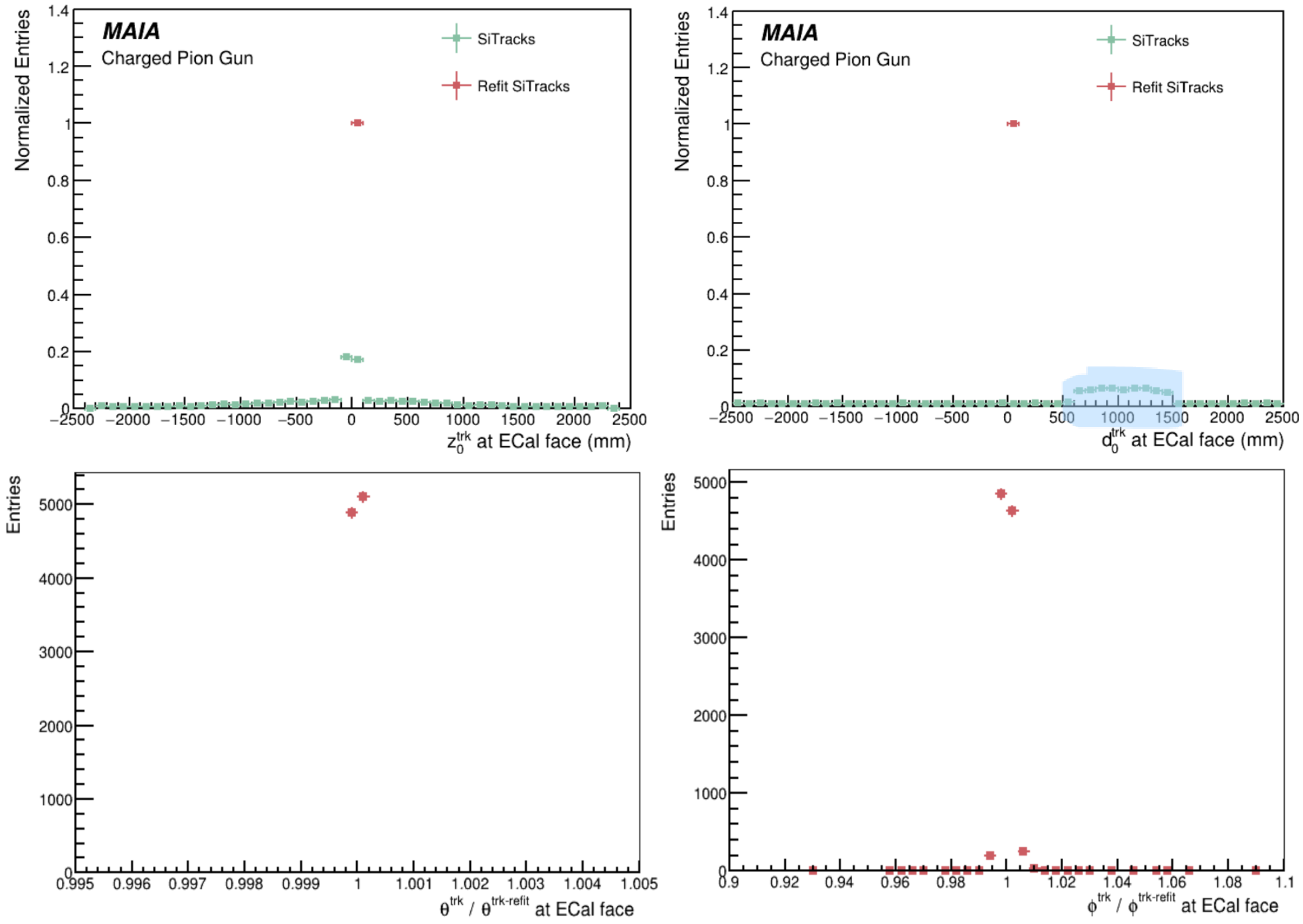






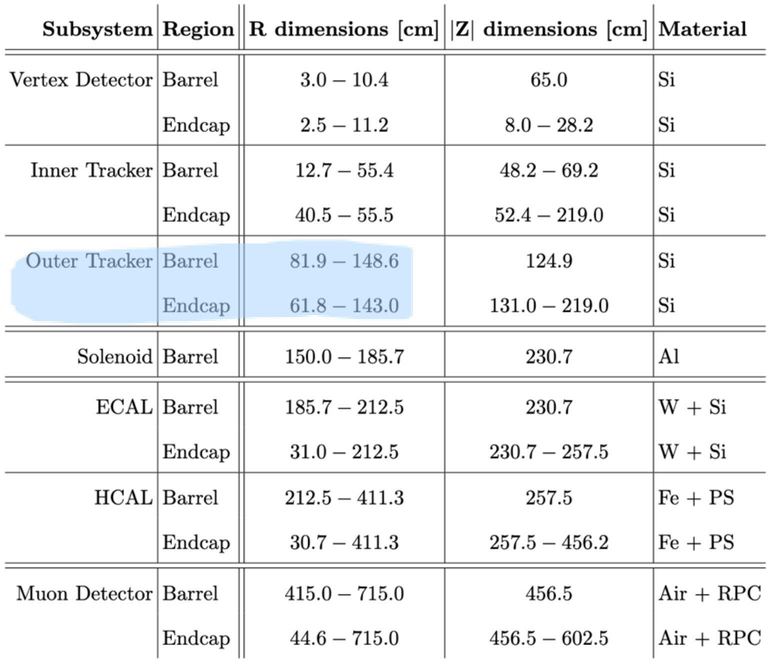
These are all exactly zero! Is it not being filled?

Track States @ ECal Face



What's going on here?

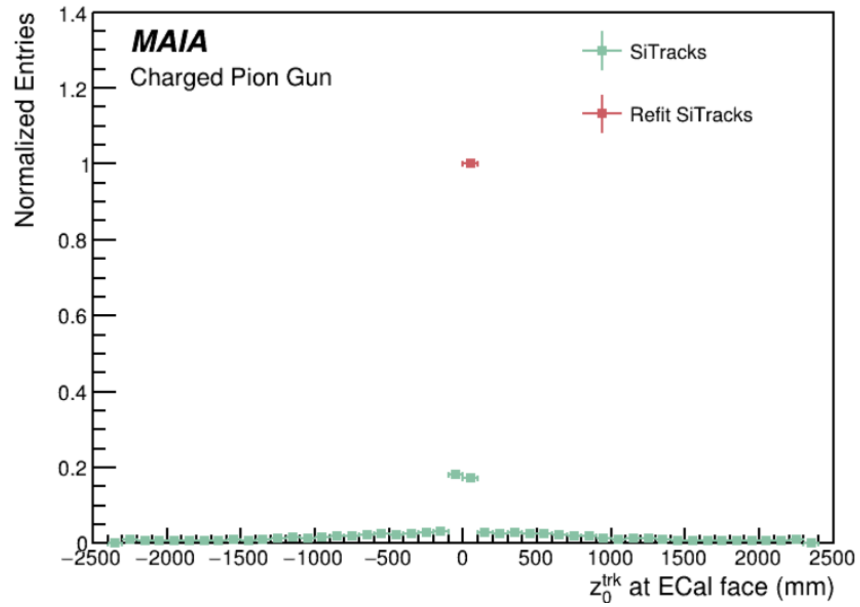
Impact parameter in outer tracker?

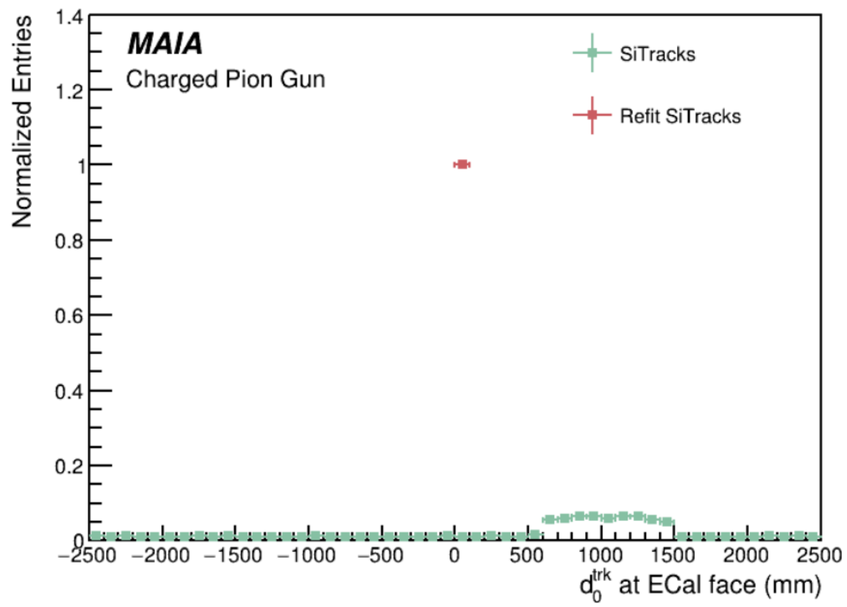


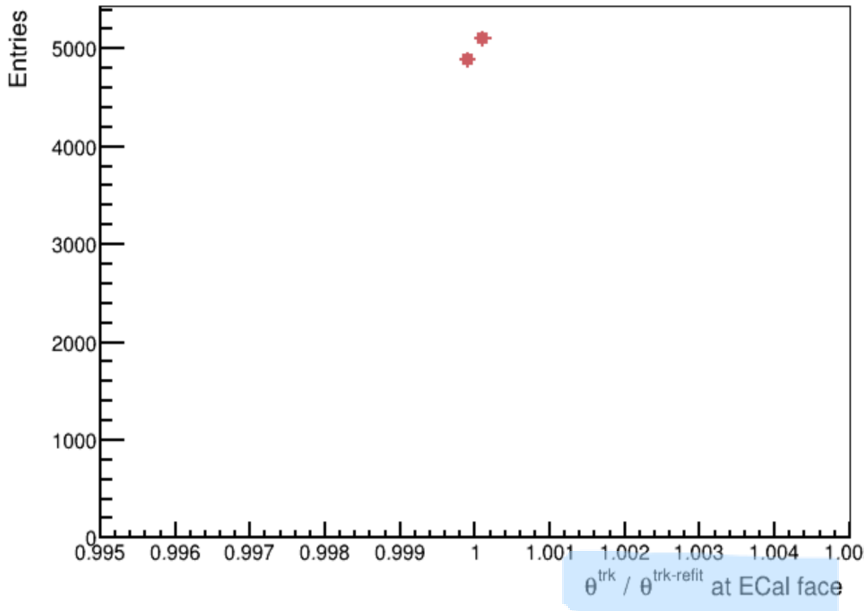
Subsystem	Region	R dimensions [cm]	|Z| dimensions [cm]	Material
Vertex Detector	Barrel	3.0 – 10.4	65.0	Si
	Endcap	2.5 – 11.2	8.0 – 28.2	Si
Inner Tracker	Barrel	12.7 – 55.4	48.2 – 69.2	Si
	Endcap	40.5 – 55.5	52.4 – 219.0	Si
Outer Tracker	Barrel	81.9 – 148.6	124.9	Si
	Endcap	61.8 – 143.0	131.0 – 219.0	Si
Solenoid	Barrel	150.0 – 185.7	230.7	Al
ECAL	Barrel	185.7 – 212.5	230.7	W + Si
	Endcap	31.0 – 212.5	230.7 – 257.5	W + Si
HCAL	Barrel	212.5 – 411.3	257.5	Fe + PS
	Endcap	30.7 – 411.3	257.5 – 456.2	Fe + PS
Muon Detector	Barrel	415.0 – 715.0	456.5	Air + RPC
	Endcap	44.6 – 715.0	456.5 – 602.5	Air + RPC

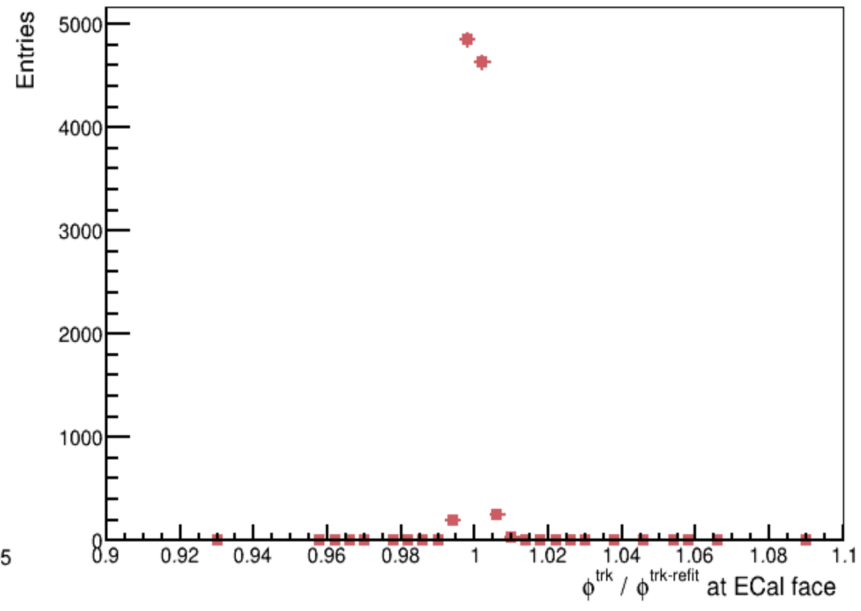
Table 1: Boundaries and materials of individual subdetectors.

Track States @ ECal Face



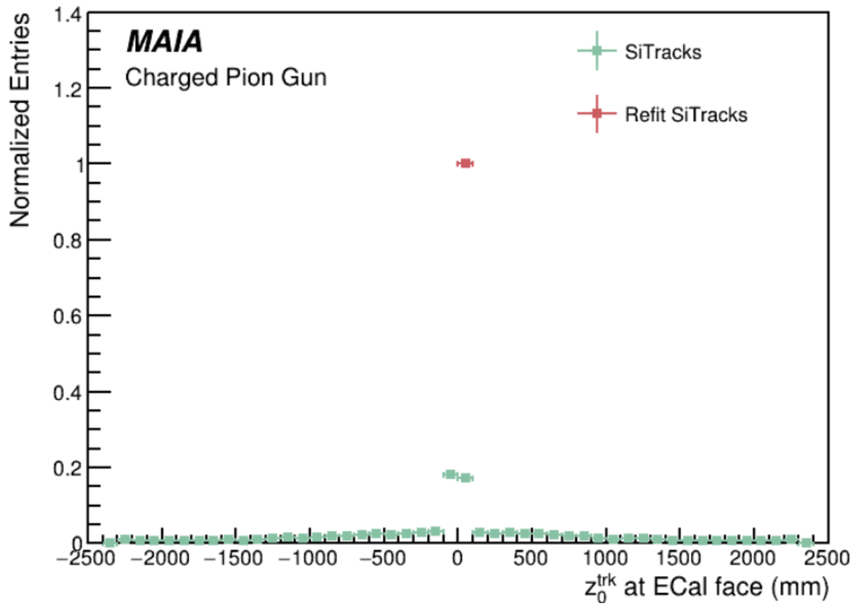


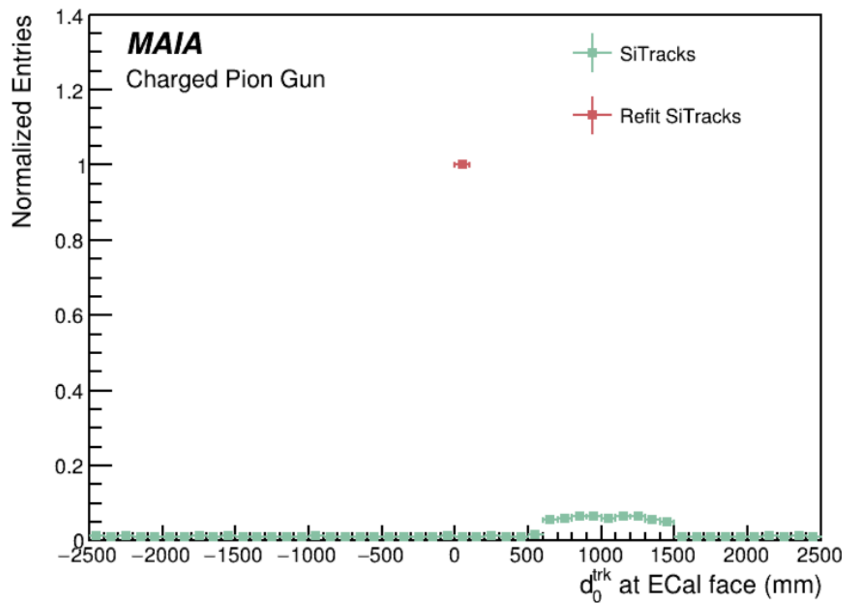


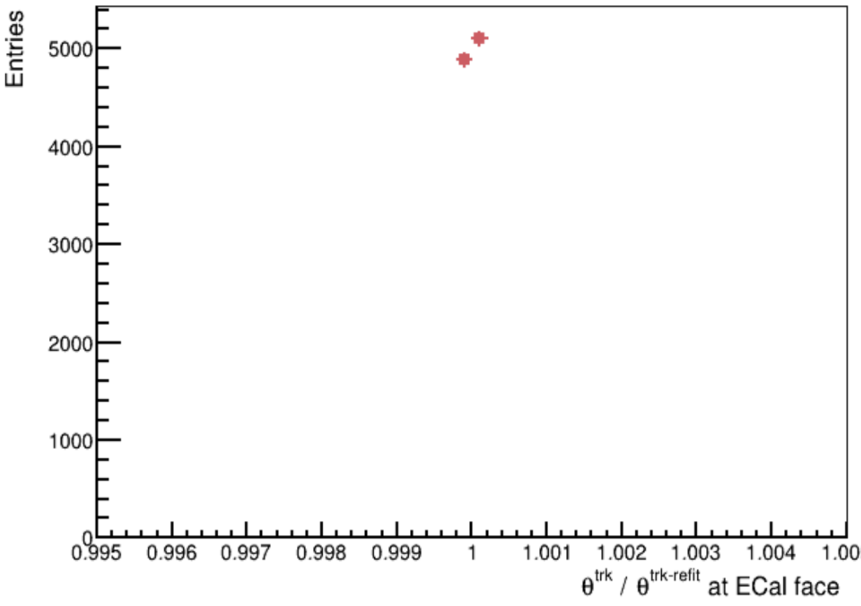


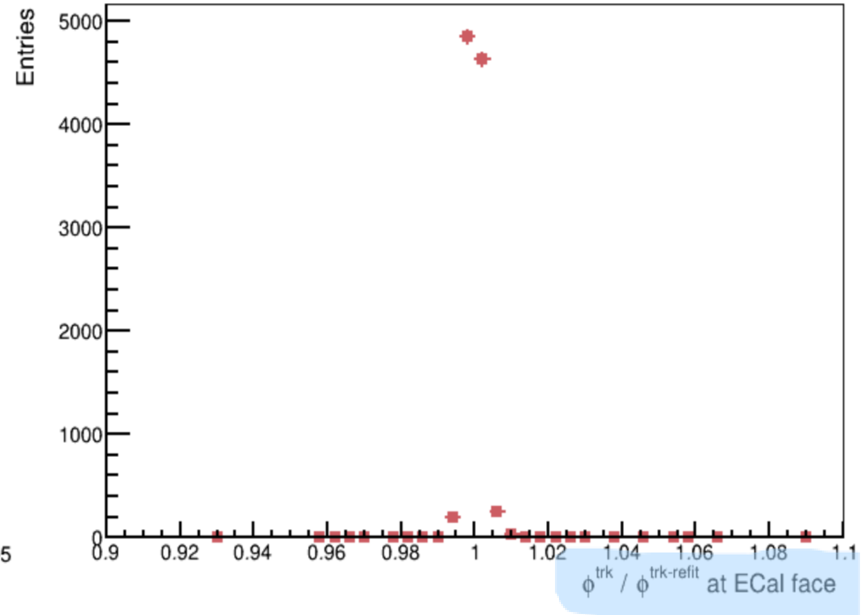
θ @ ECal face for SiTracks and SiTracks_Refitted almost exactly the same?

Track States @ ECal Face



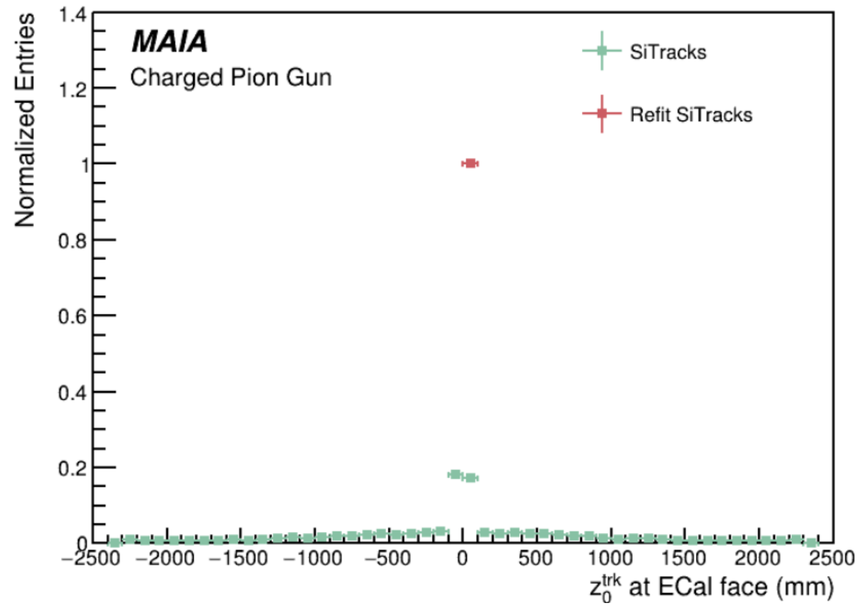


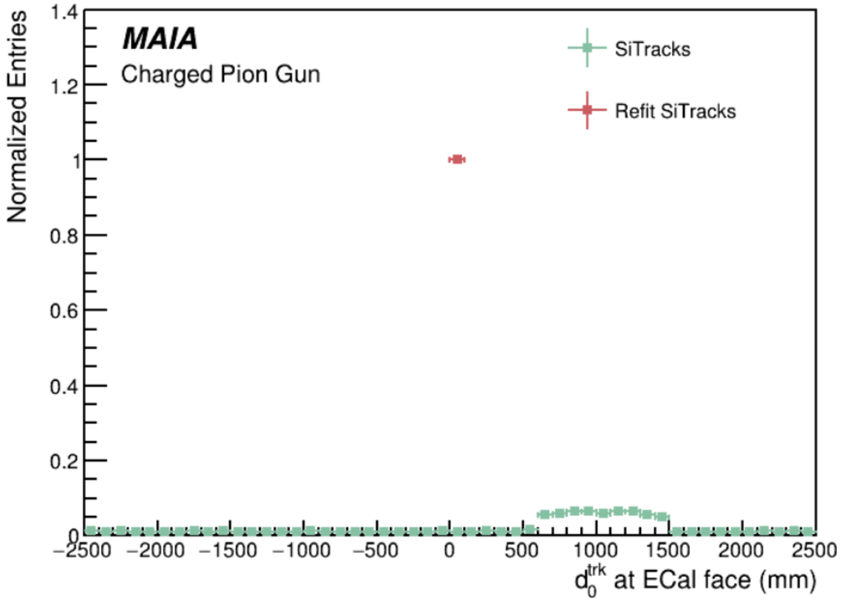


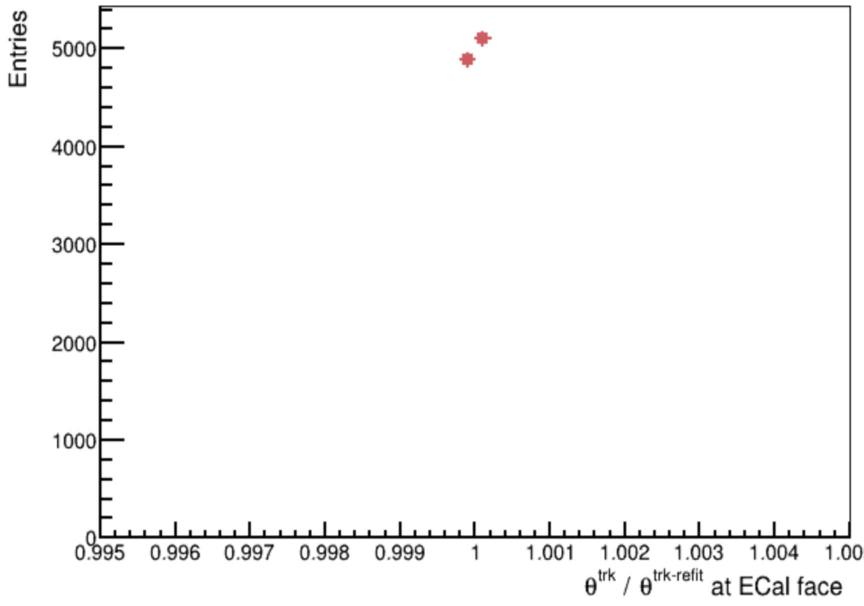


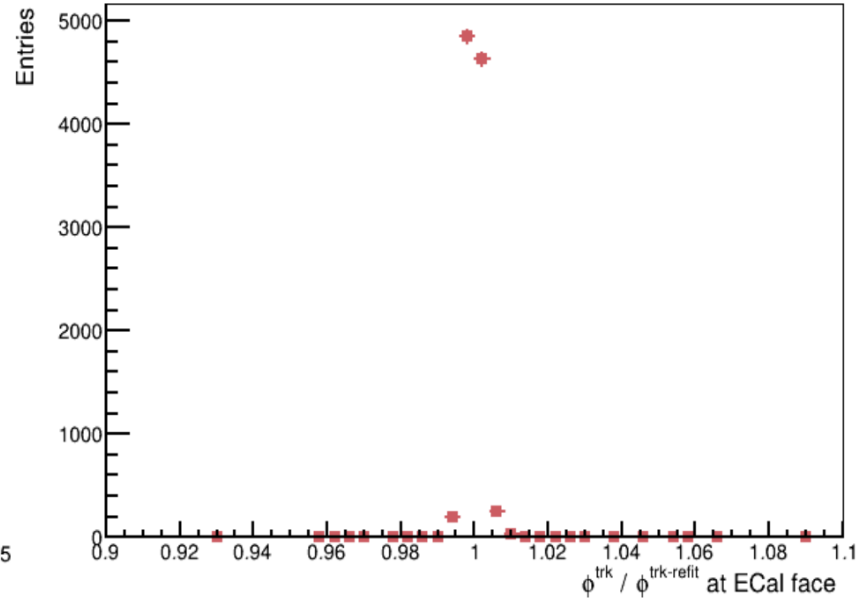
At least this is reasonable!

Track States @ ECal Face









Overall difficult to understand these plots... any ideas?

Otherwise, we will continue by following a track through its cutflow

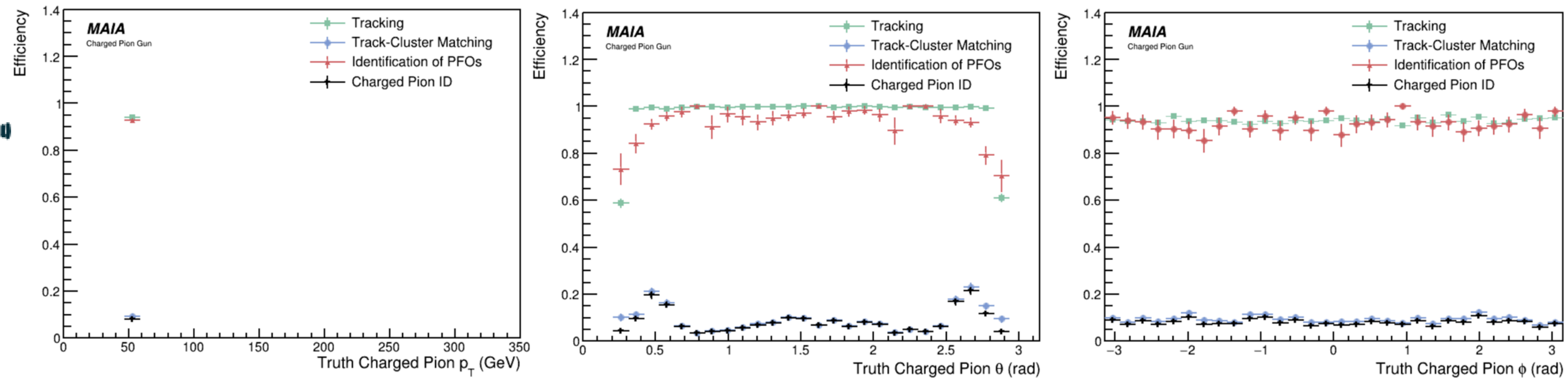
Track-Cluster Matching “Cutflow”

- There are **five** places where tracks can be lost for track-cluster matching:
  - Track **PassesQualityChecks() (link)**
    - *See backup for details!*
  - Track **CanFormPFO() (link)**
    - *See backup for details!*
  - Track **reaches ECal face**:
    - Technically a part of CanFormPFO(), but difficult to isolate (not a controllable cut)
    - Previous slides potentially show signs of problems here
  - Various references to tracks in **Cluster Seeding (link)**
    - See backup for clustering details
    - Still haven’t gone through all of this in detail
  - **Track-Cluster Matching (link)**
    - Still haven’t gone through all of this in detail

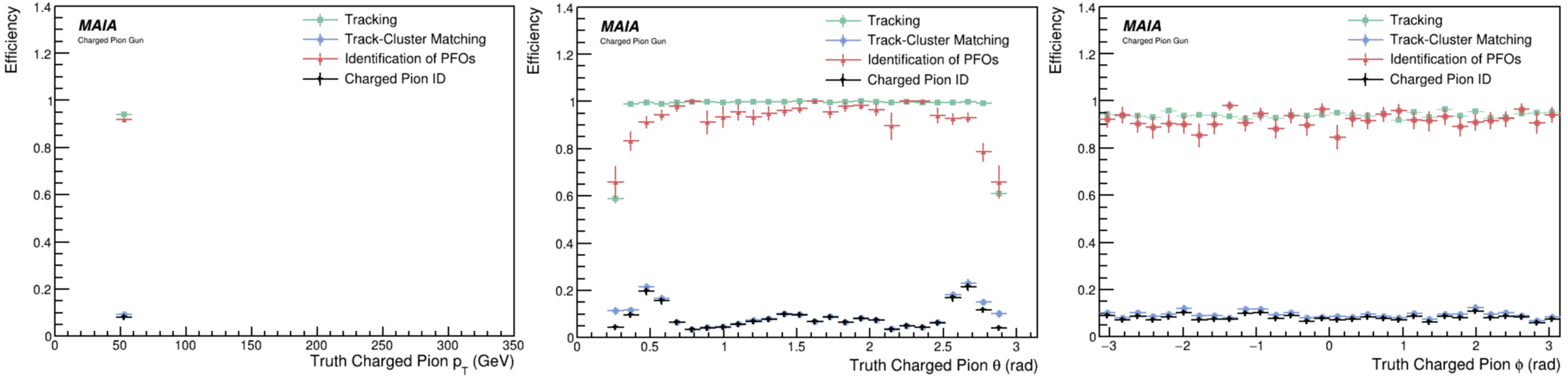
- *The idea going forward: systematically tune each cut to be loose, isolating the inefficiency*

Impact of “PassesQualityChecks” and “CanFormPFO”

Default
" PassesQualityChecks "
and
" CanFormPFO "



Infinitely loosened
" PassesQualityChecks "
and
" CanFormPFO "



failure point must be later – to be continued..

Summary

- Not sure if track extrapolation to ECal face is making sense
- Some of the track "cutflow" has been worked through
  - **Up next:** We'll continue to requirements in the clustering algorithm!
    - Particle interactions w/ solenoid being upstream of Ecal may motivate looser track selections in clustering

- I'm likely to submit a poster abstract on charged pion (maybe taus also?) to the IMCC annual meeting
  - Any objections?

Pandora Clustering Steps

Loose summary: parameters and exact details get to be understood (by me) and checked against code

- This could make sense given how Pandora clustering / PFO creation works :
  1. Clusters are seeded with tracks @ ECal face
  2. "Typical" clustering, beginning from the ECal face to the HCal end
     1. Pandora will tend to split clusters
  3. Clusters are then merged, according to several algorithms
  4. Attempt to match cluster to track. If cluster energy inconsistent w/ track pT, *do not associate*. Instead, try to combine that cluster with another to see if the energy becomes consistent w/ trk pT.
  5. Run fragment removal, i.e., merging neutral clusters nearby charged clusters
  6. Form PFOs
  7. Run PFO ID

this could explain previous slides - clusters being split with ~even energy leads to none being consistent with track p_T

→ match failure

Main question from here:

Why is one track container "better" than the other?

Track-Cluster Matching “Cutflow”

- Track **PassesQualityChecks() (**link**)** has requirements on the following parameters:
  1. Position of track state @ Ecal face
     1. Cut is a lower bound, set to 0 mm. Shouldn’t cause any problems.
  2. Track radius of curvature ≠ 0. Should be fine!
  3. $\sigma(p)/p$
     1. Cut @ 0.15 by default

Track-Cluster Matching "Cutflow"

- Track **CanFormPFO() (link)** has requirements on the following parameters:
    1. It must reach the calorimeter! Is this always happening?
        1. **Challenge:** Difficult to study – no print statements and not tunable. ECal propagation only reasonable method, but previous studies in this show that more work is needed to understand track extrapolation.
    2. It is not a parent track. This should be fine.
    3. These three requirements are ***OR*** requirements!
        1. **Requirement 1:**
            1. Track d0 < 200 mm
            2. Track z0 < 200 mm
            3. If the tangential distance of the closest ***hit*** to (0,0) is < trackerInnerR + 200 mm
        2. **Requirement 2:**
            1. usingNonVertexTracks = 1. This is satisfied in the config and by default.
            2. Requirements on the position of the closest hit, which could be different hits for r and z:
                1. z-distance of closest hit (in z) is < 250 mm
                2. R-distance of closest hit (in r) is < trackerInnerR + 200 mm
        3. **Requirement 3:**
            1. "IsV0" = True OR isDaughter = True. This is whether the parent is a track from "v0" (maybe the "primary vertex" in ATLAS / LHC jargon?) or a daughter track.