CDCTRG 3D fitter

U-Tokyo M2 Hiroto Sudo

Review

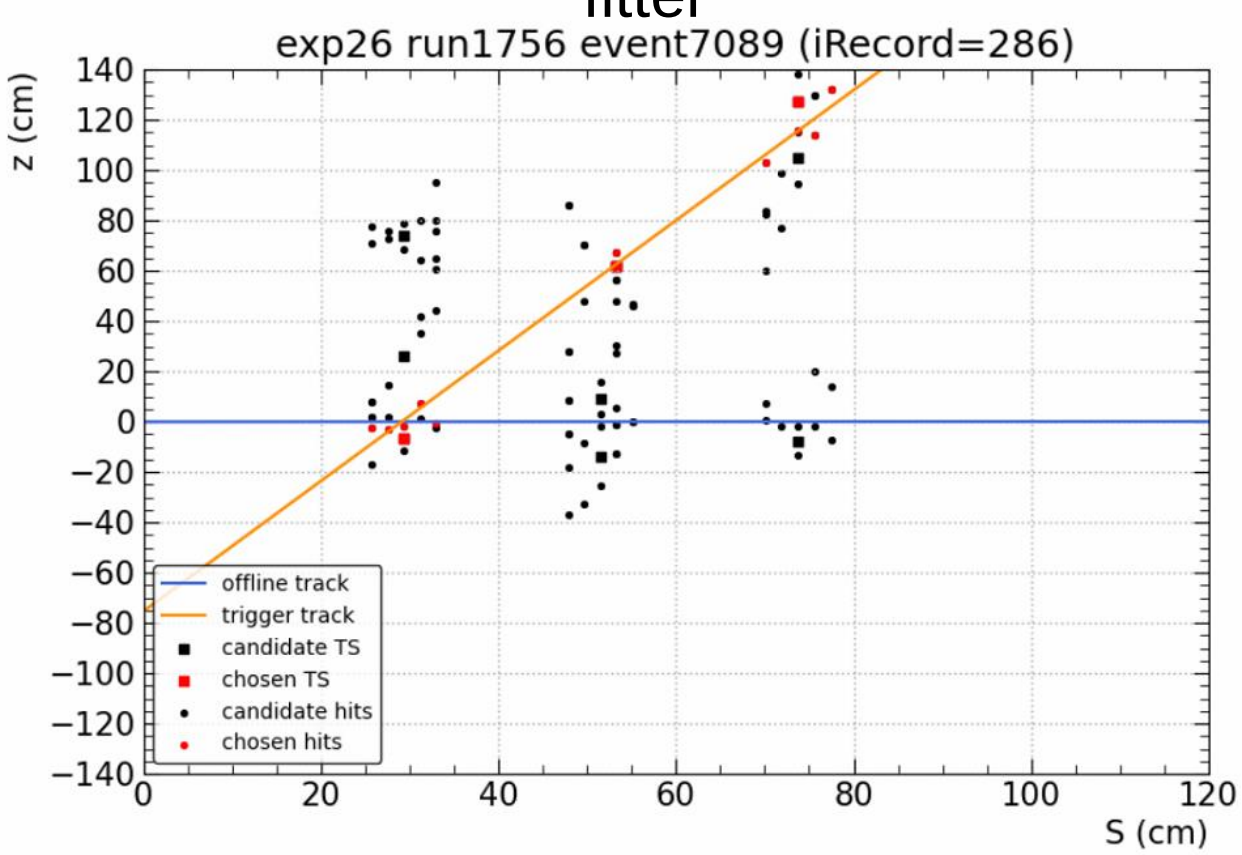
- Motivation
  - To reduce trigger rate, I want to **improve BG rejection rate** while **keeping efficiency.**

- So far I tried
  - Drift correction, ADC, hit selecting
  - **Full hit** (use all wires, not only priority wire)
  - **Voting** (like 2D Hough voting) for selecting hits

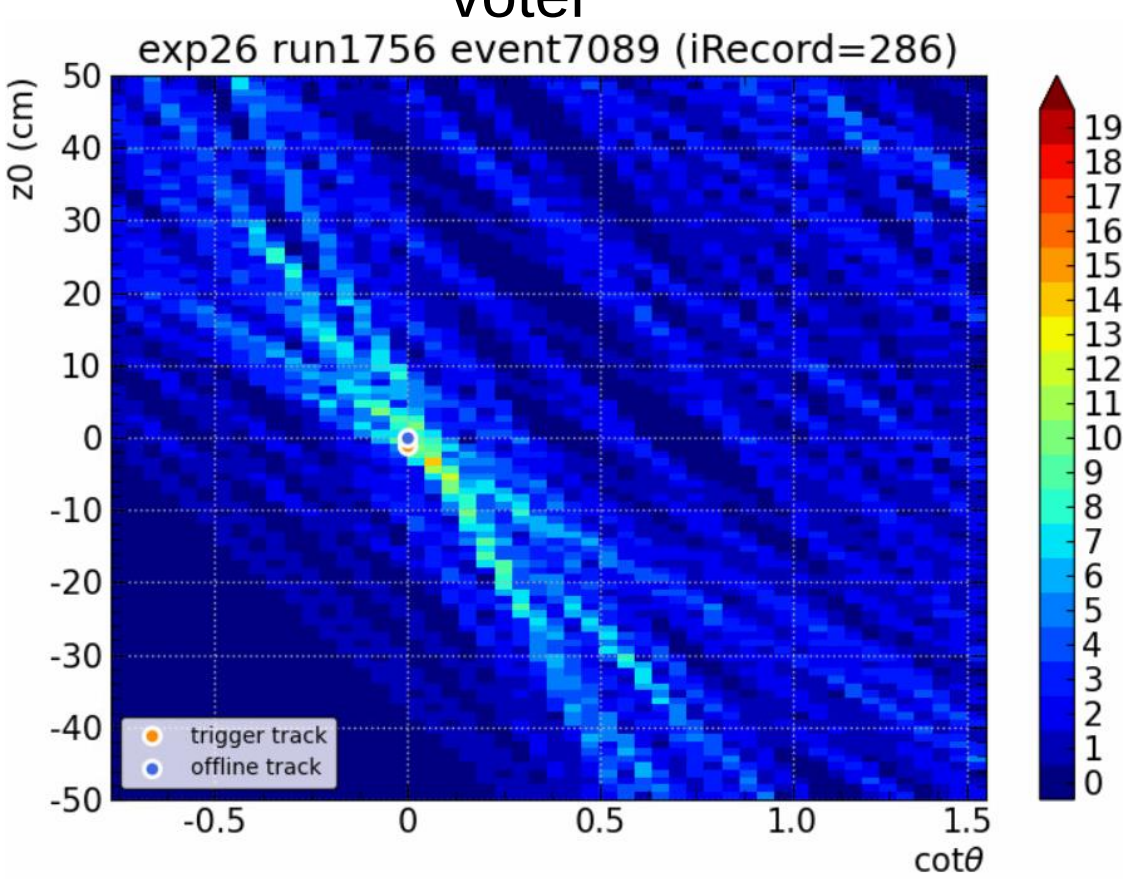
} seems good!

Voting

fitter



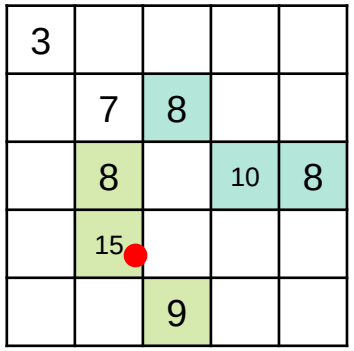
voter



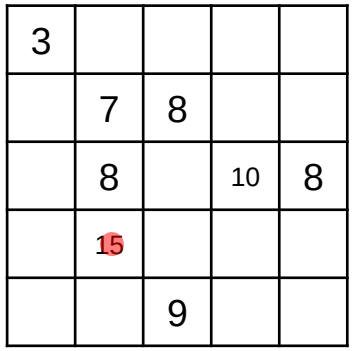
Voter is **noise-robust**

Voting

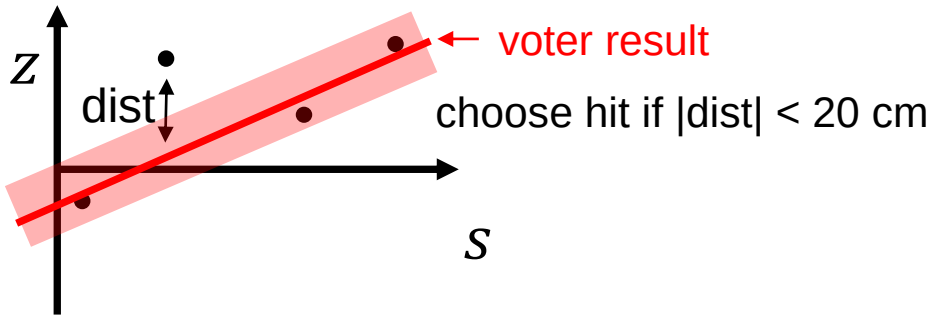
- Cell size = 50 x 40 (for $z_0 \in [-50, 50]\mathrm{cm},\ \cot\theta \in [-0.8, 1.5]$)
  - based on optimization
  - cf. Cell size of 2D Hough ~ 5000
- Peak finding
  - **A) clustering**
    - Threshold for peak candidate = 8
    - Assume ↘ shaped cluster
    - No cluster size limit
    - Result is center of mass of the most voted cluster
  - **B) Maximum**
    - Result is most voted cell
  - **C) Voting(clustering) + fitter**
    - Select hits near voter's result and fit
  - **D) Voting(maximum) + fitter**



clustering

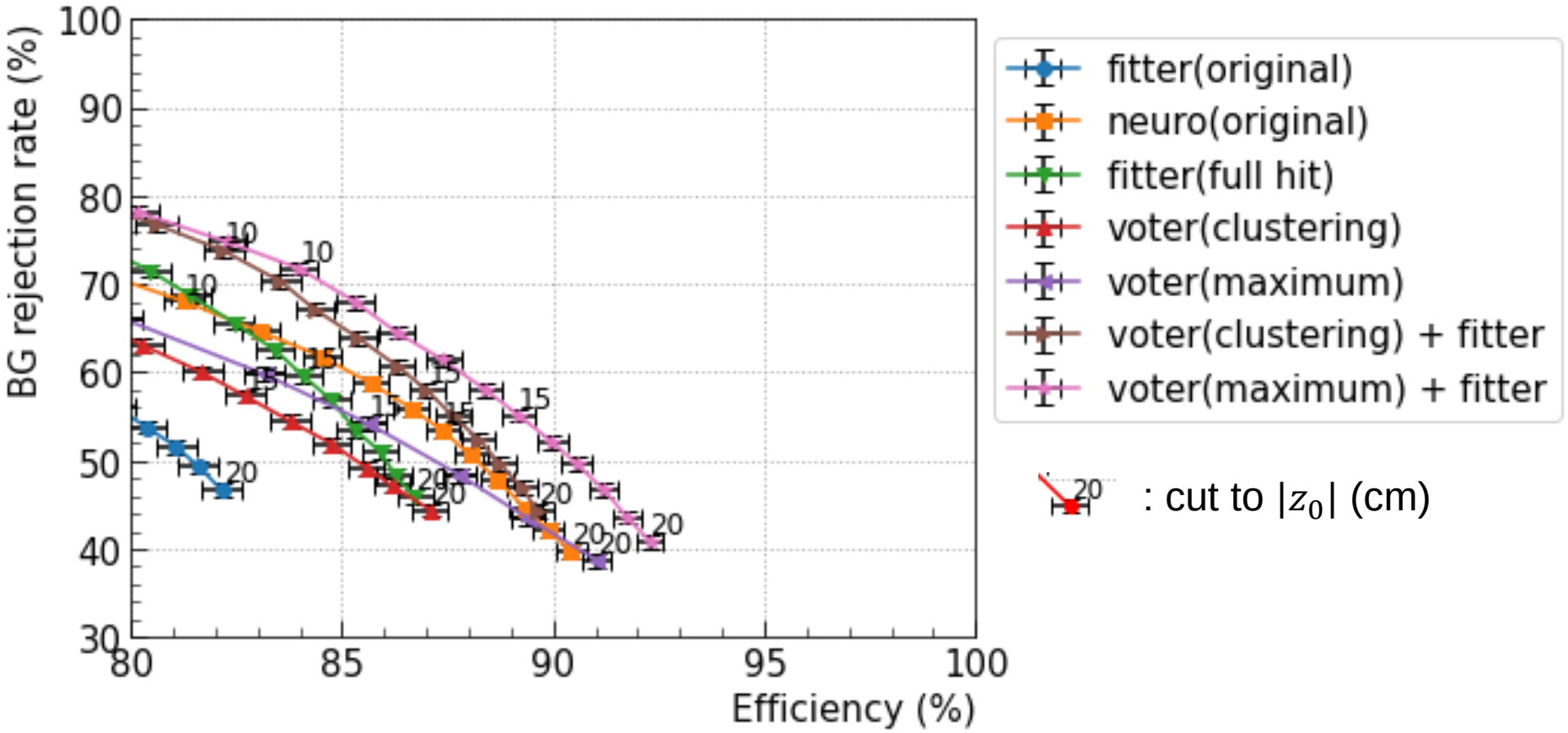


maximum



voting + fitter

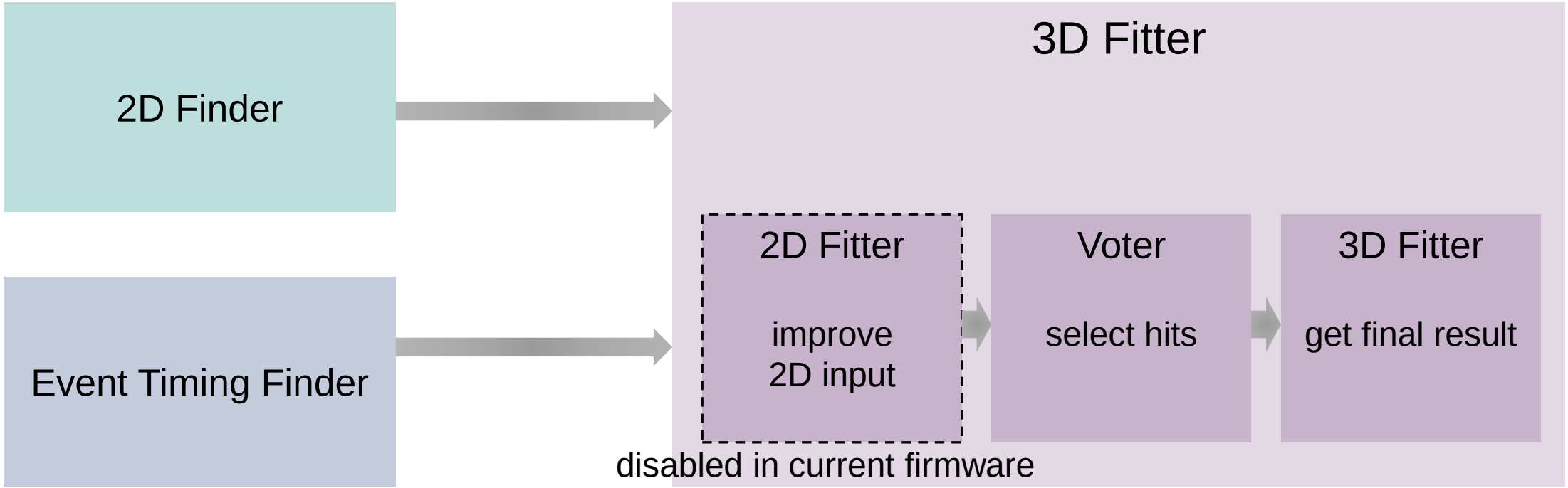
Performance



- voter(maximum) + fitter has best performance

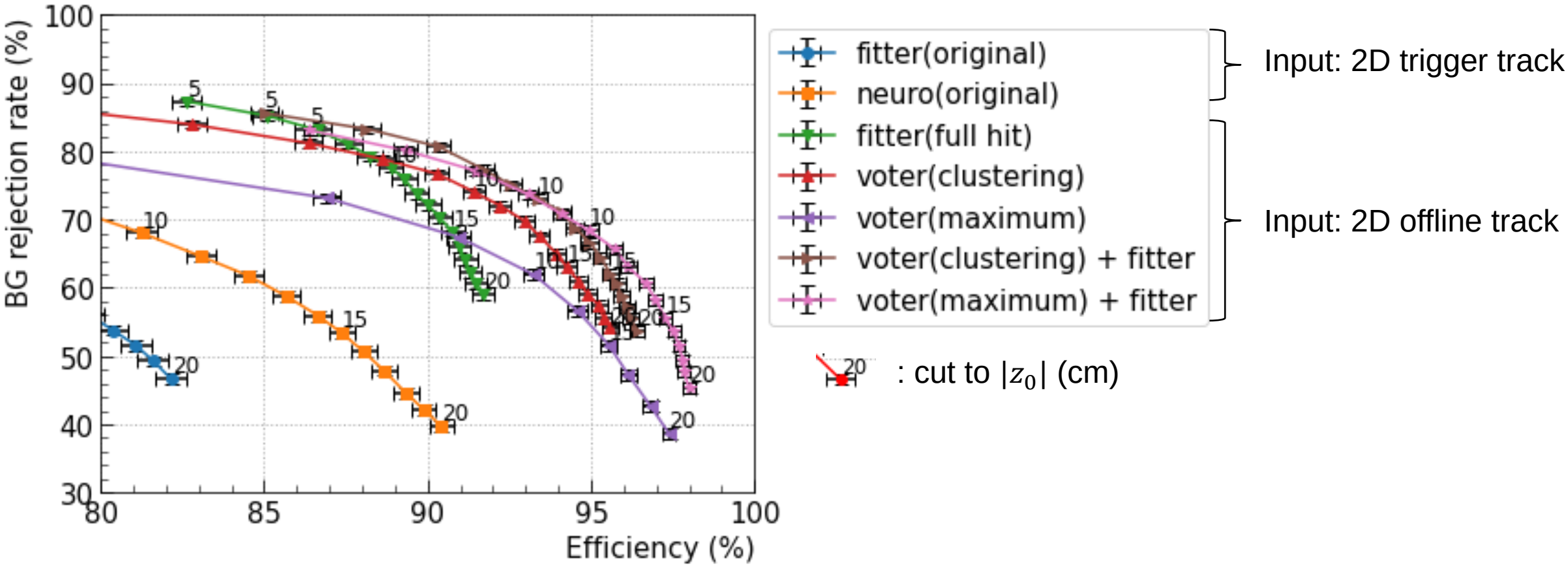
(error bar is based on 68% Clopper-Pearson confidence interval)

2D fitter


2D Finder
Event Timing Finder
3D Fitter
2D Fitter
improve
2D input
Voter
select hits
3D Fitter
get final result
disabled in current firmware

Performance (with ideal input)

- Instead of 2D fitter, I used **offline 2D** input.



(error bar is based on 68% Clopper-Pearson confidence interval)

Summary & Future plan

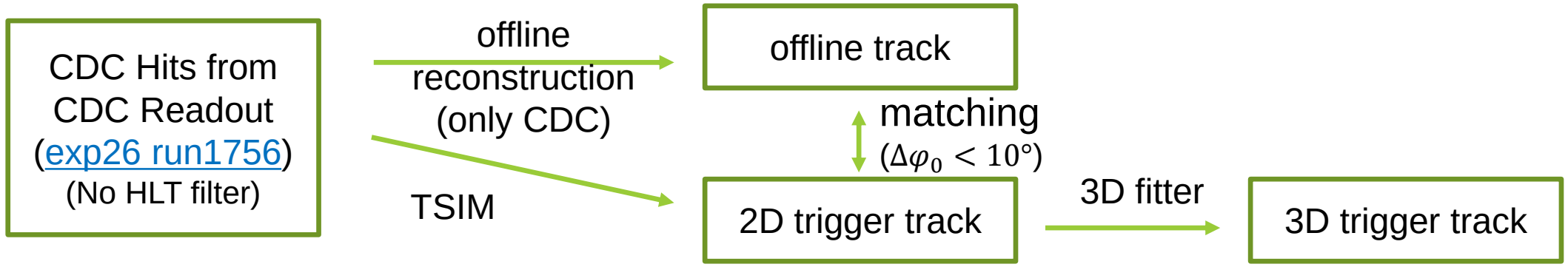
- I tried some voting methods to reduce noise hits.
- **voter + fitter** has best performance, so I want to implement this.

Plan

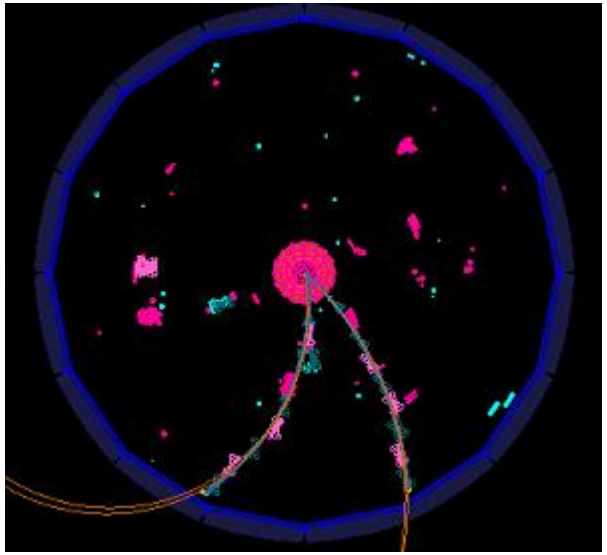
- Transfer original logic UT3 -> UT4 <- working
- add my new logic

Backup

Reconstruction and Selection



- Track Level Selection (after matching):
  - Exclude unreconstructed offline tracks(no 2D case)
  - Exclude short tracks (2D and offline)
    - short track := !(offlineTrack.getTransverseMomentum() > 0.3
      && offlineTrack.getHitPatternCDC().getLastLayer() > 50
      && offlineTrack.getHitPatternCDC().getFirstLayer() < 5)
  - Exclude overcounting 2D tracks



Performance index

Categorization for matched tracks

		Trigger Track		
		2D + 3D $\lvert z_0 \rvert < 20$	2D + 3D $\lvert z_0 \rvert > 20$	Only 2D (failed 3D tracking)
Offline track	$\lvert z_0 \rvert < 1$ (signal)	signal	loss	loss
	$\lvert z_0 \rvert > 1$ (BG)	BG	rejected BG	rejected BG
	no track	fake	rejected fake	rejected fake

Performance index

3D efficiency := #(signal) / #(signal offline track)

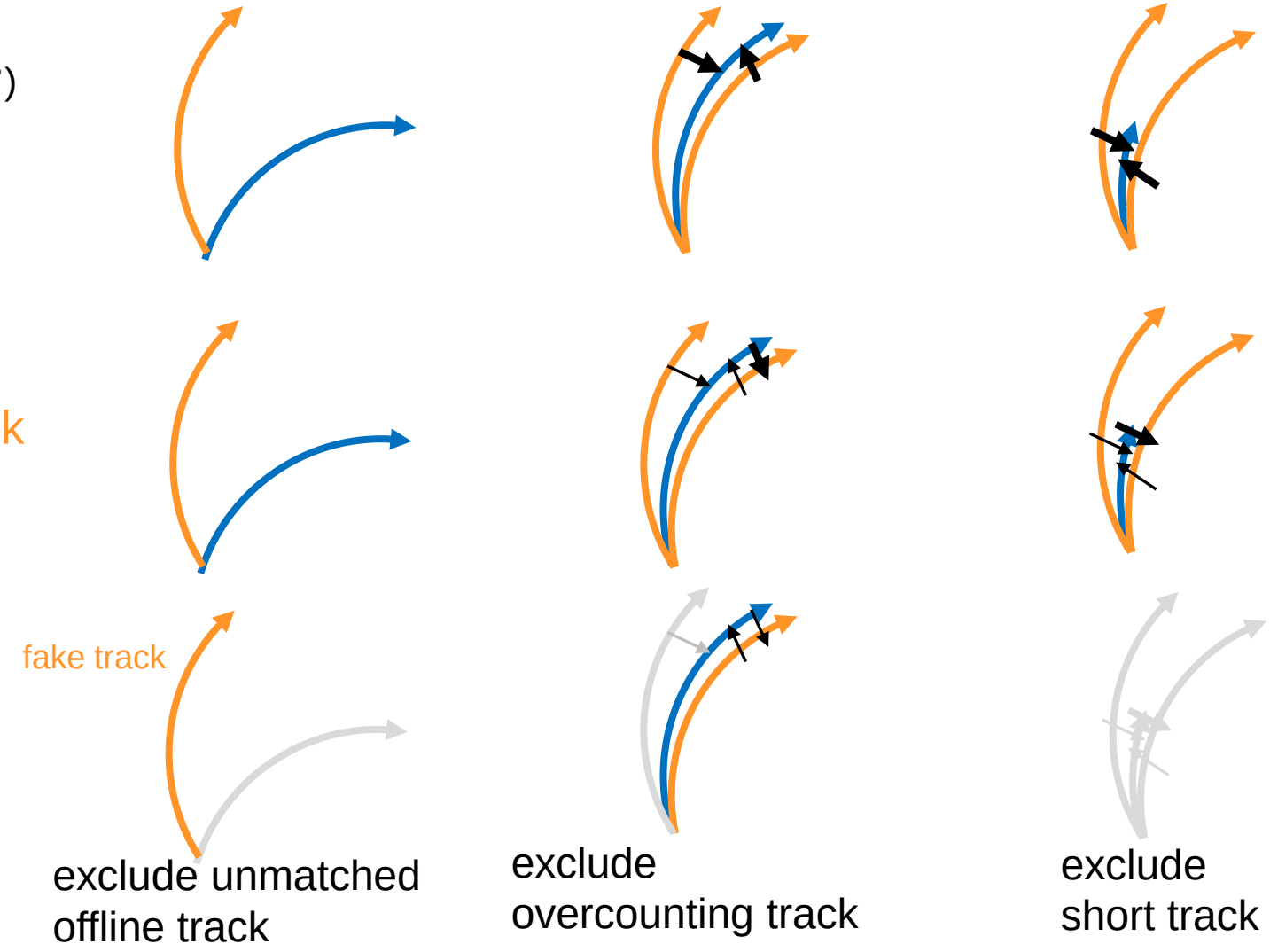
BG rejection rate := #(rejected BG) / #(BG offline track)

Matching based on 2D track

1. matching from trigger track
to nearest offline track ($\Delta\varphi_0 < 10°$)
※"nearest" means $\Delta\varphi_0$ is the smallest.

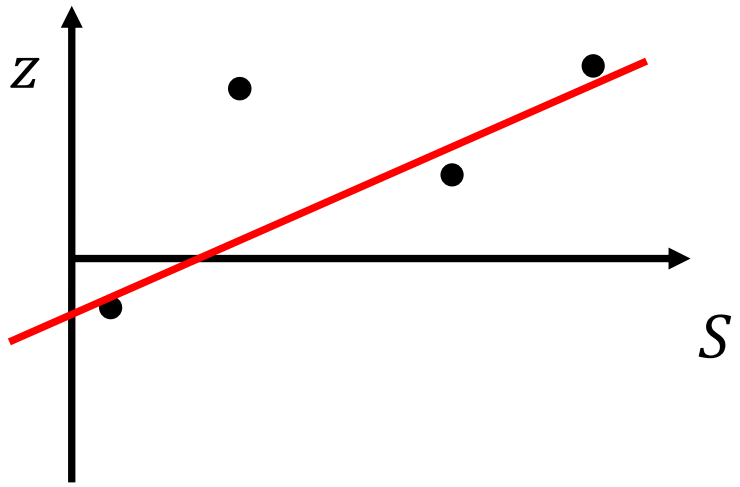
2. matching from offline track
to nearest matched trigger track

3. selection



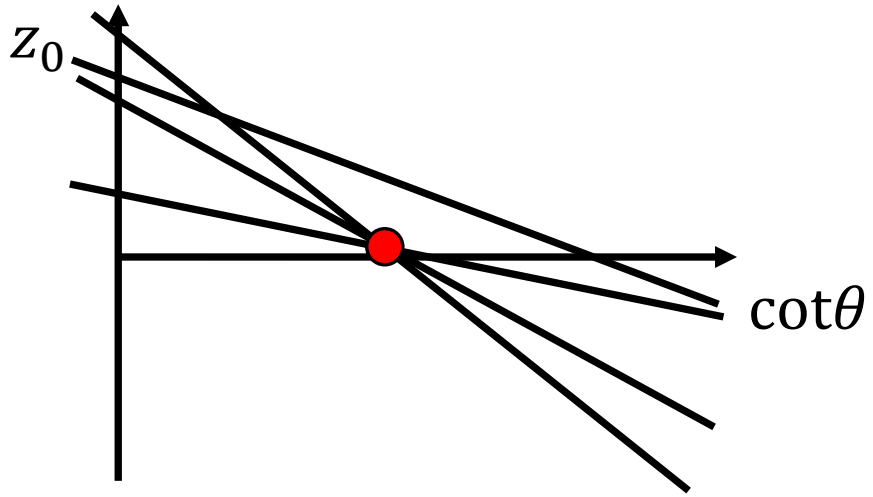
Voting

Fitting



$$z_{hit} = +\cot\theta \cdot S_{hit} + z_0$$

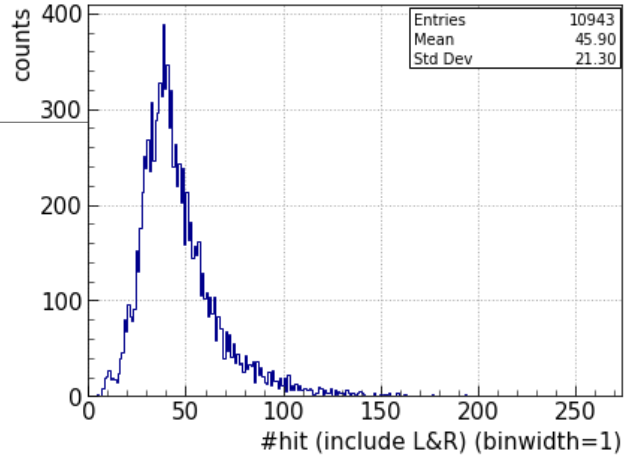
Voting



$$z_0 = -S_{hit} \cdot \cot\theta + z_{hit}$$

Merit: Noise doesn't affect the correct point

Algorithm of new 3D fitter



get $z_{0,voter}, \cot\theta_{voter}$ from voter using all candidate hits (< ~100hits)

↓

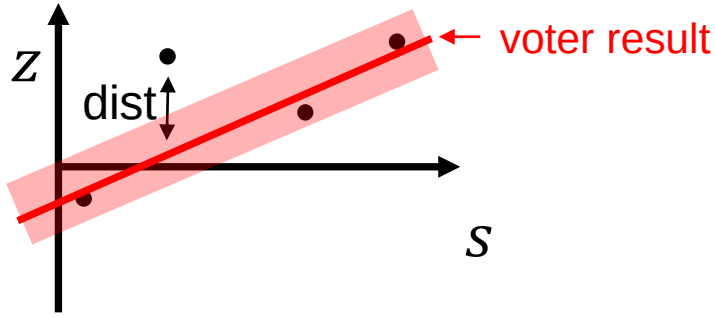
choose 4TS if |dist| < 20 cm
(choose nearest one in 1 Super Layer)

↓

choose hits in chosen TSs
(choose nearest one in 1 layer)
(choose nearer one about L/R)

↓

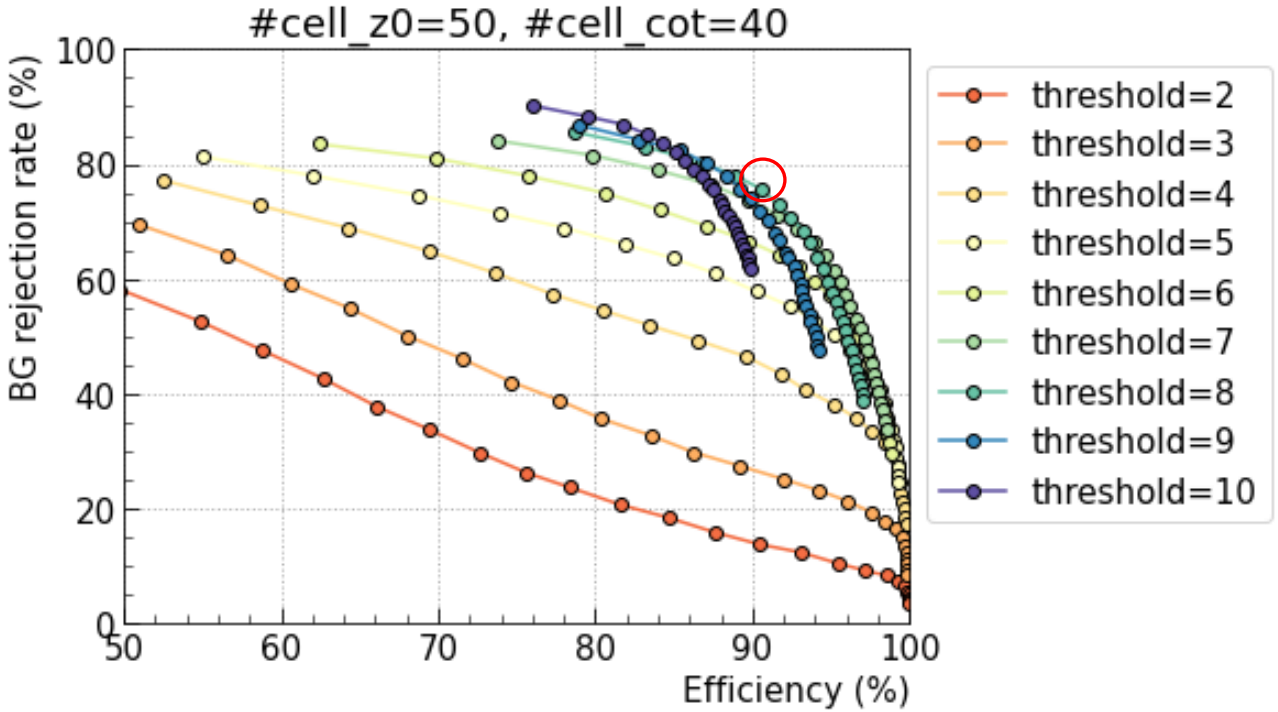
fit



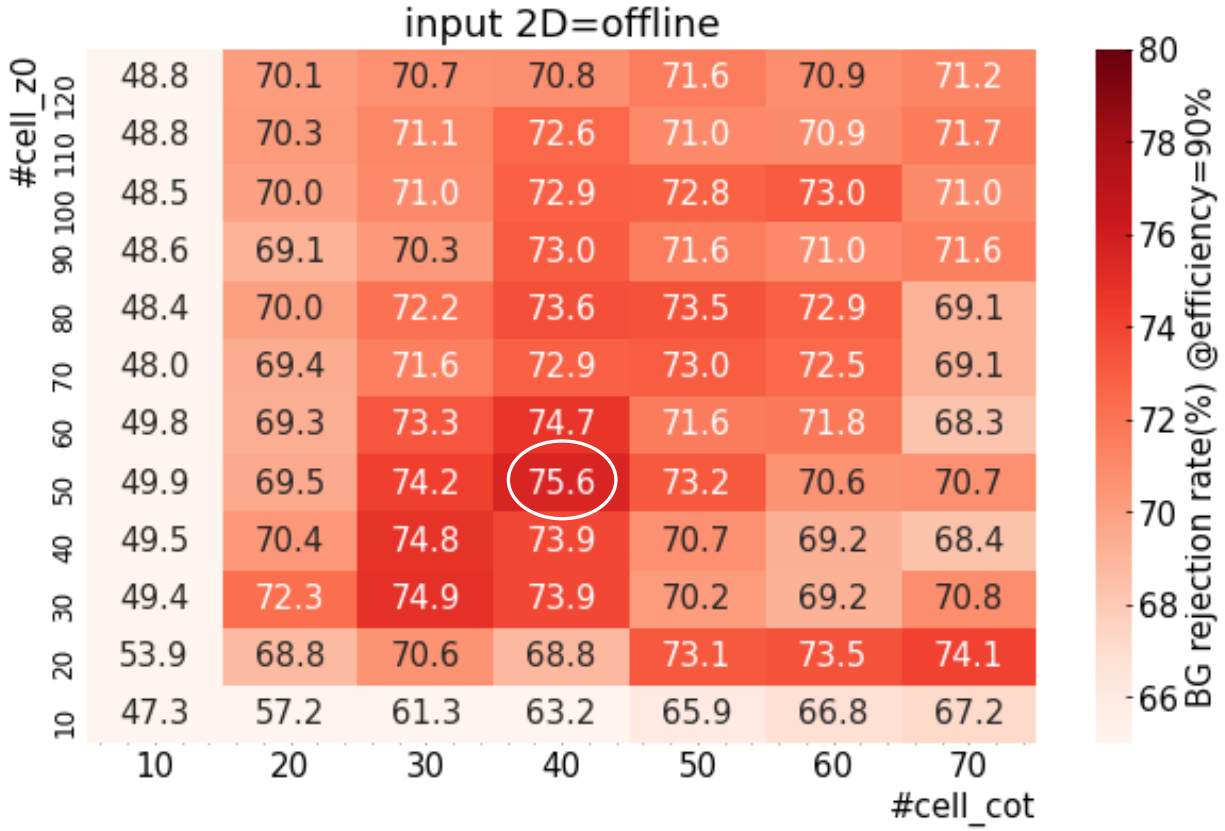
at most 20 hits are chosen
(4 Super Layer x 5 layer in SL)

Optimization of voter

- Voter
  - with clustering (no fitter)
  - input2D = offline
- Objective value
  - BG rejection rate @ efficiency > 90%
- Cell size: grid search
  - $\#\text{cell\_z0} \in [30, 40, \ldots, 120]$
  - $\#\text{cell\_cot}\theta \in [30, 40, 50, 60, 70]$
- Peak threshold
  - scan (2~10) and adopt best one

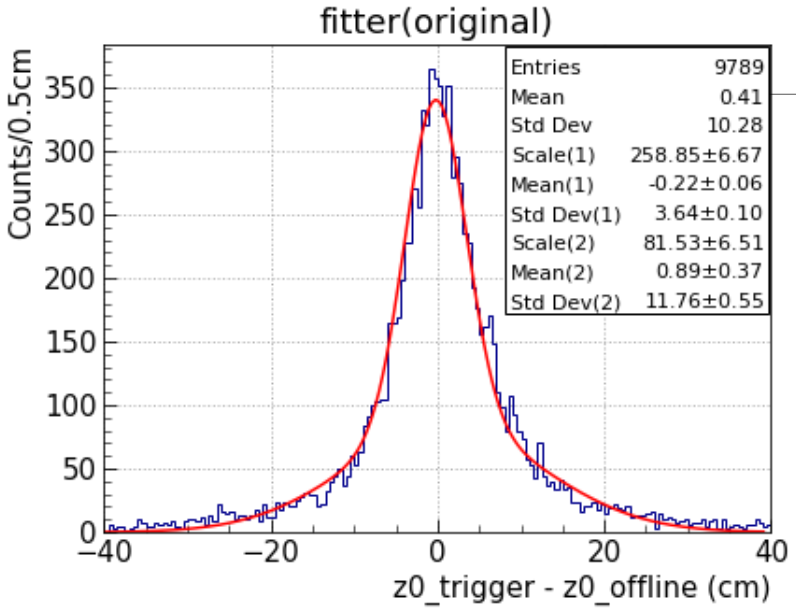


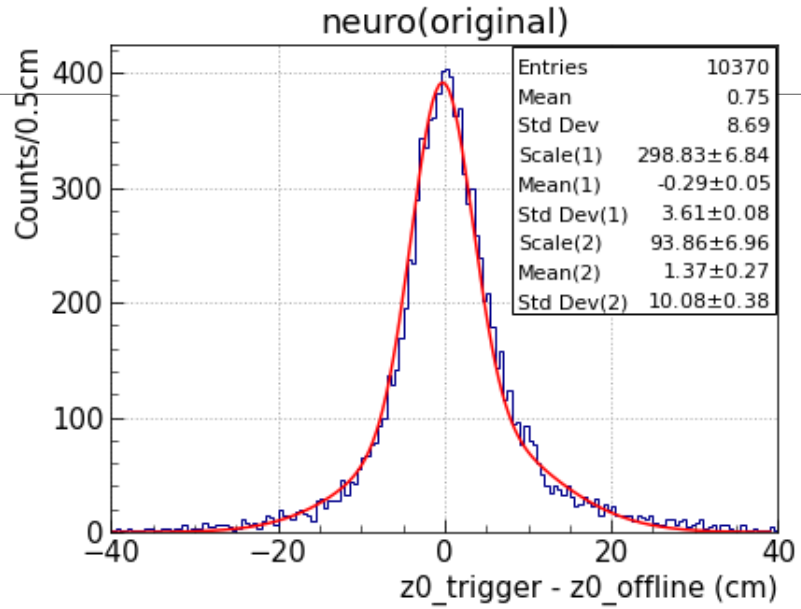
Optimization of voter

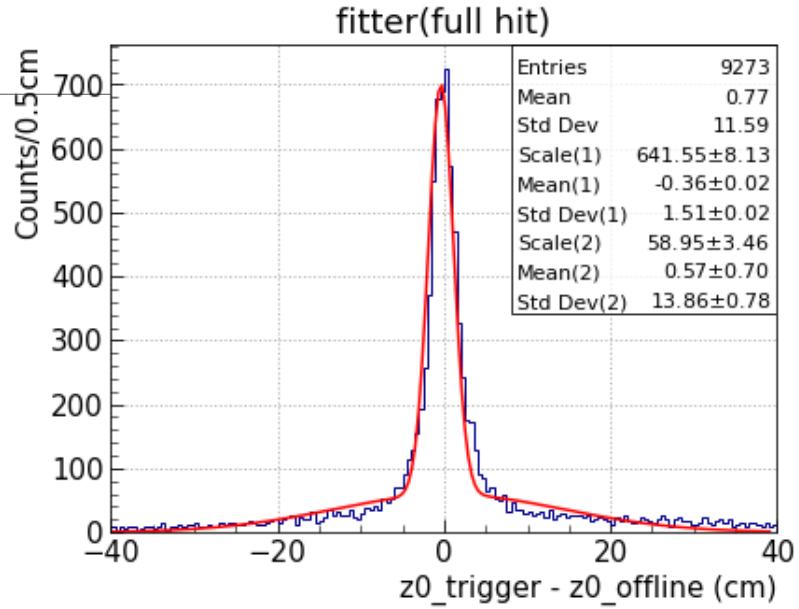


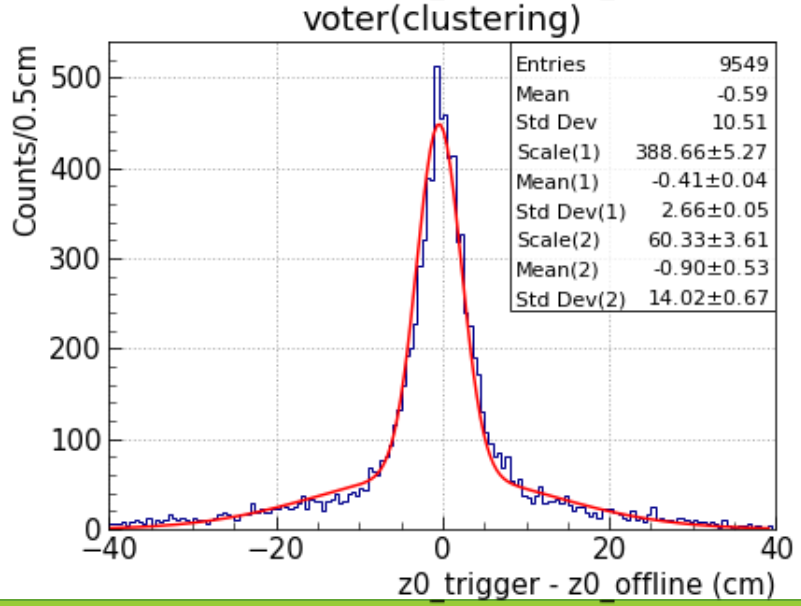
- #cell_z0 = 50, #cell_cot = 40 seems optimal

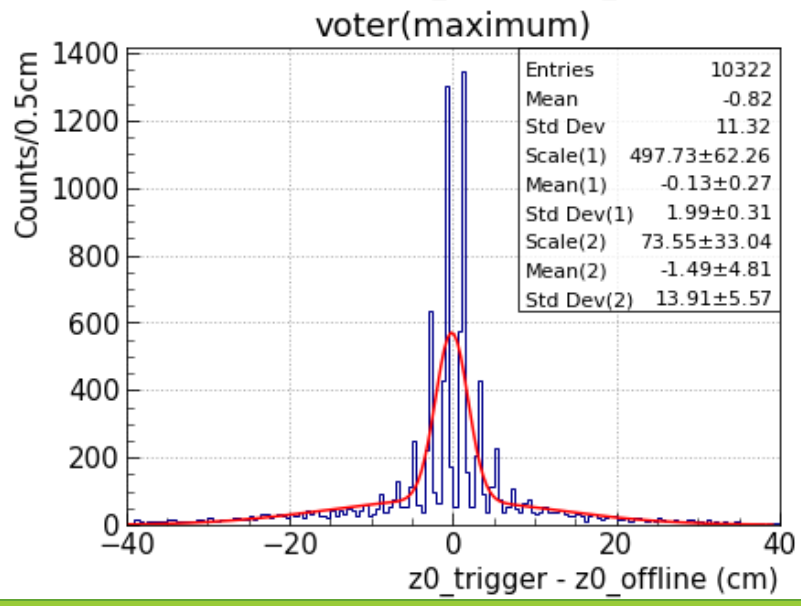
z_0 resolution (input: offline2D)

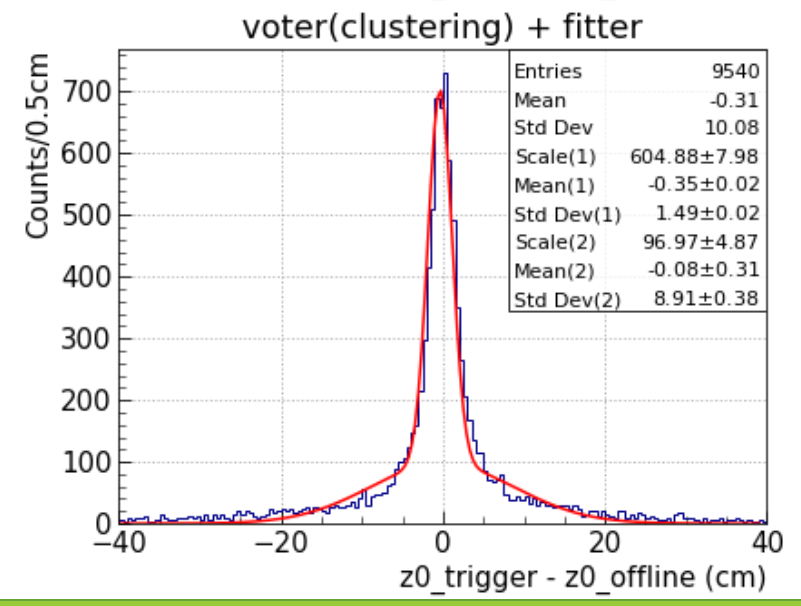




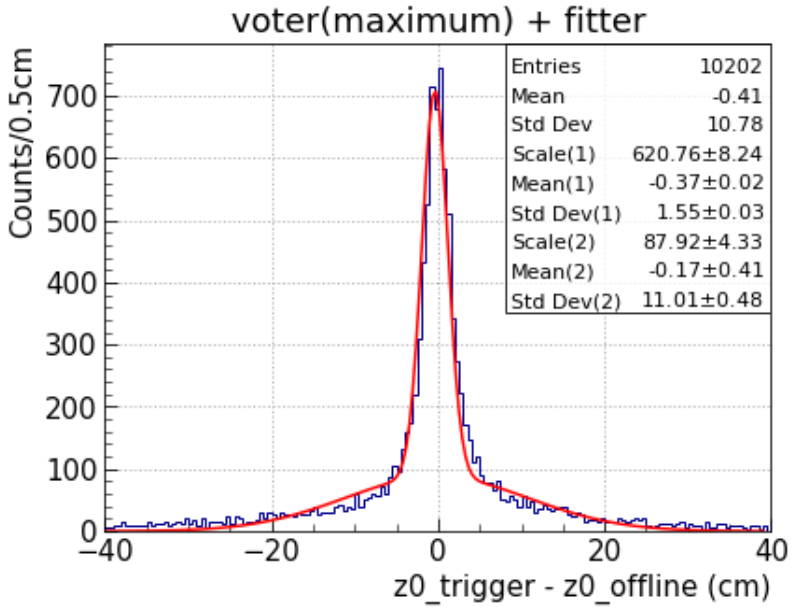


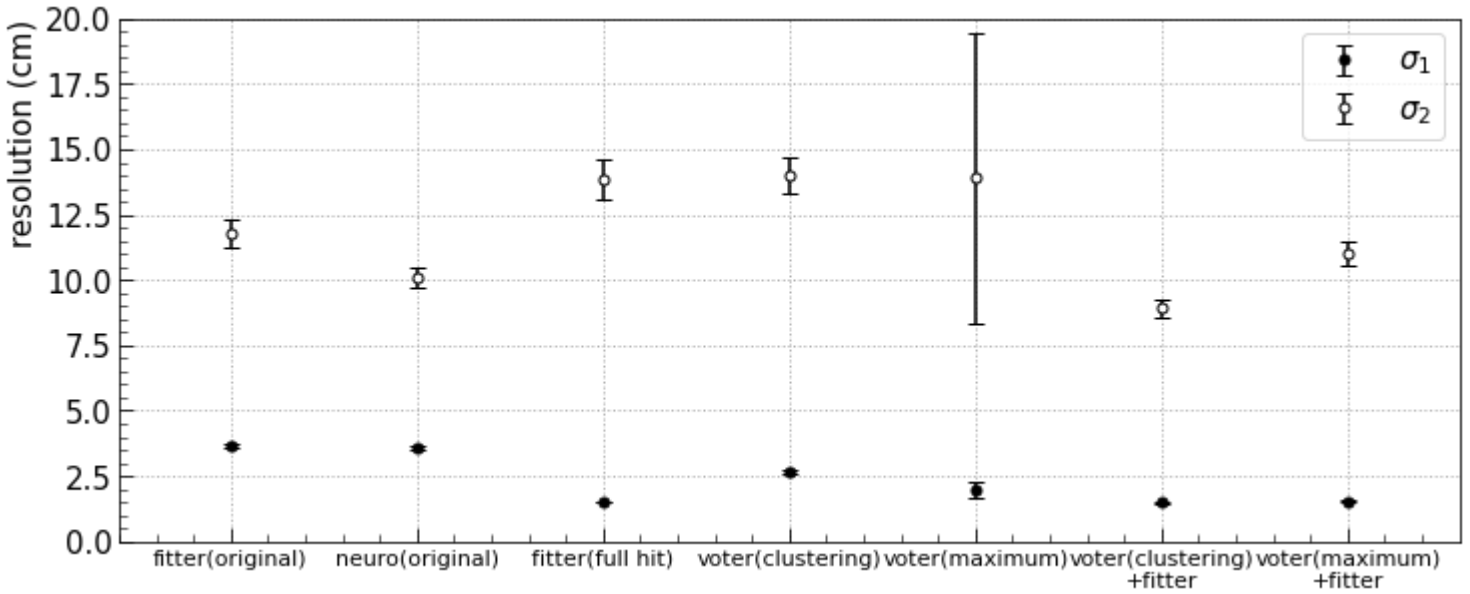




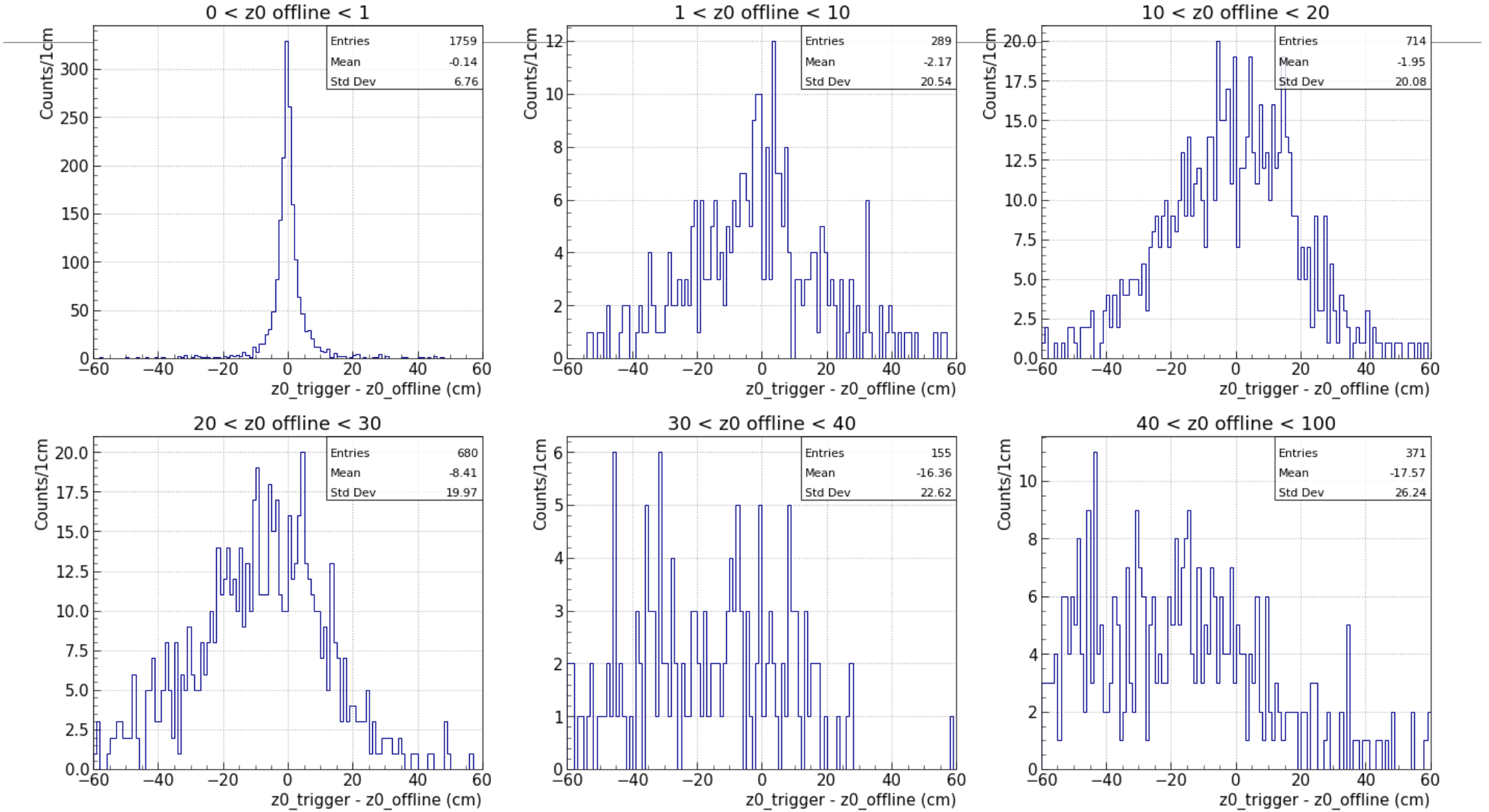


z_0 resolution (input: offline2D)





z_0 resolution for each z region(input: offline2D)



z_0 resolution for each z region(input: offline2D)