ROI FINDING EFFICIENCY UPDATE

Giulia Casarosa

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Alexander von Humboldt Stiftung/Foundation

Virtual Tracking Meeting ~ January 27th 2017

ROI Efficiency Recap

➡ Definition based on the fraction of digits inside an ROI:

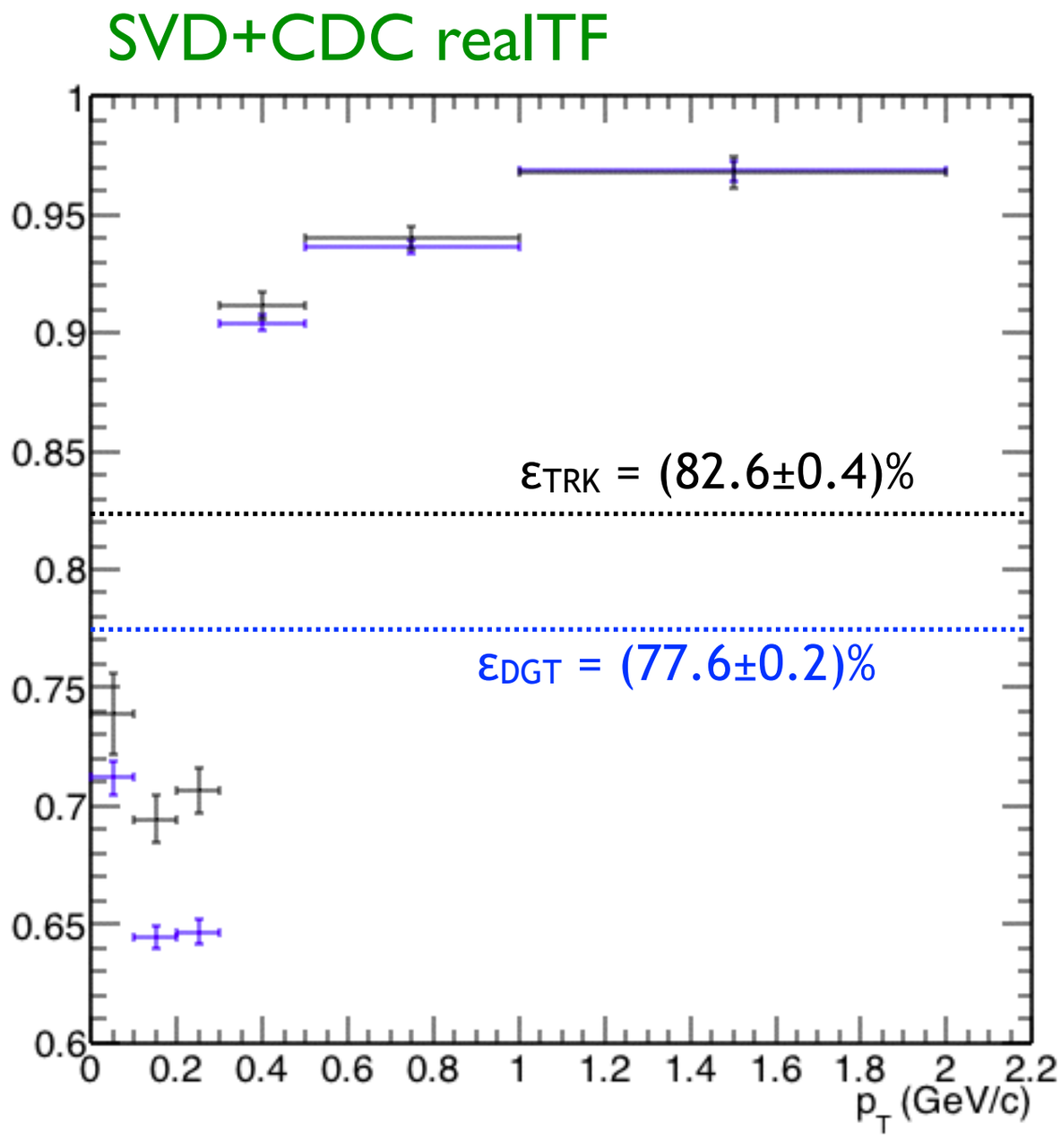
$$\varepsilon_{DGT} = \frac{\text{\# PXDDigits inside a ROI}}{\text{\# PXDDigits of Track}}$$

➡ Introduced ε_{TRK}:

$$\varepsilon_{TRK} = \frac{\text{\# Tracks with at least one related PXD Digit inside a ROI}}{\text{\# Tracks with at least one related PXD Digit}}$$

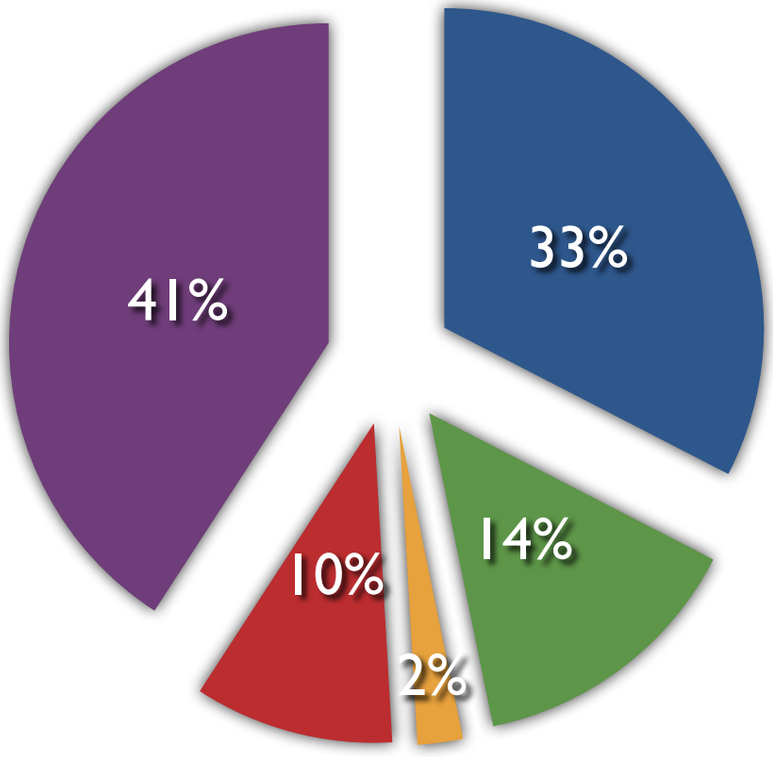
➡ 17% of the fitted tracks misses the intercepts:

- inefficiency in the extrapolation (unlikely)
- SVD track finder performance (extrapolation of CDC tracks to the PXD planes does not have sufficient precision)
- SVD+CDC track merging performance (CDC tracks not correctly merged with SVD tracks count as inefficiency)

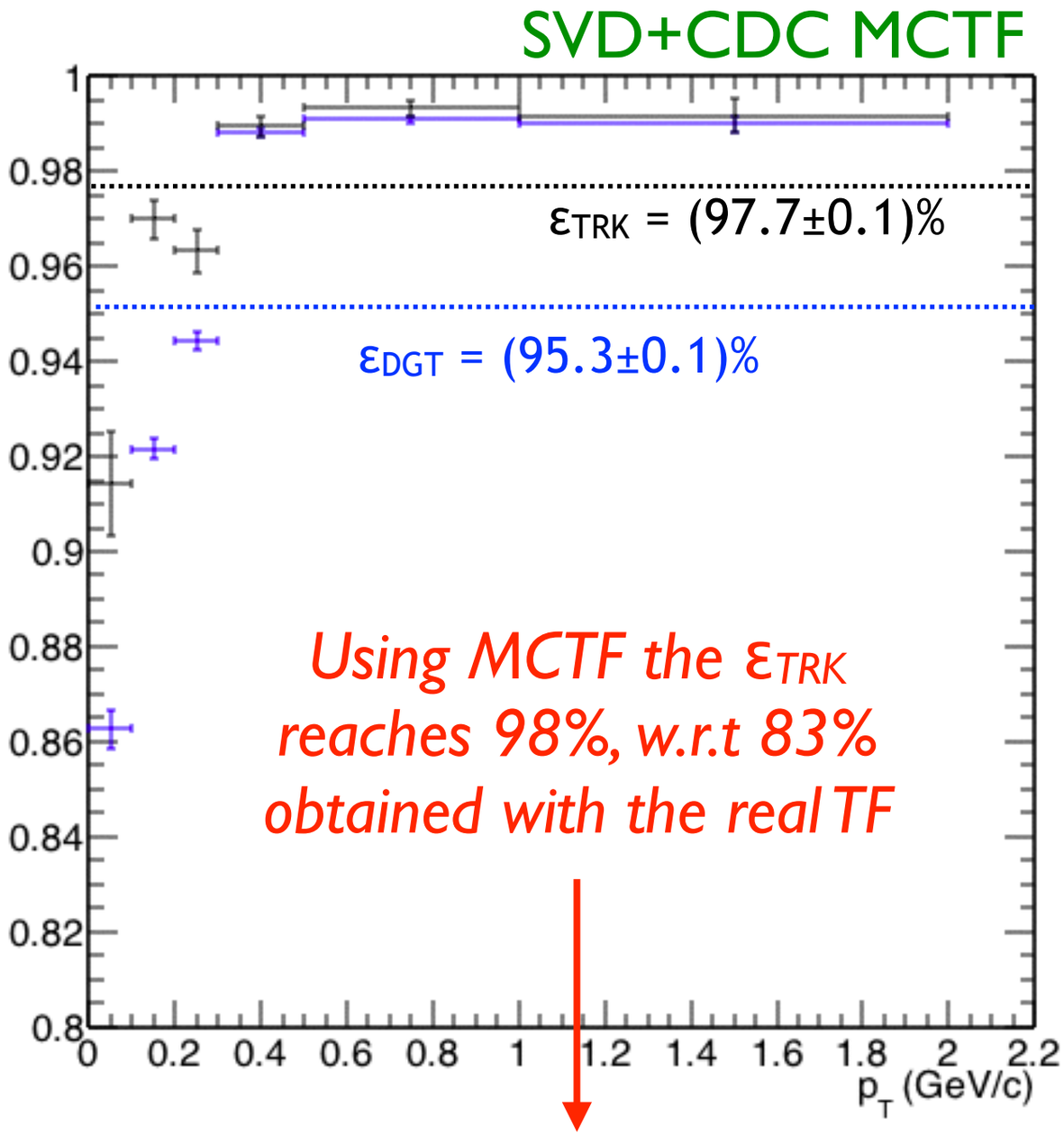


SVD+CDC MC Tracking

- Using the MC track finder, VXDTF and track merging are factored out
- less than 1% of the tracks miss the intercept



- ROI with correct VxdID (71)
- ROI with wrong VxdID (31)
- no ROI, Intercept with correct VxdID (5)
- no ROI, Intercept with wrong VxdID (22)
- no Intercept (89) ← it was 1454 with the real TF



The inefficiency is not due to problems in the extrapolation

Efficiency Definition

➡ In order to investigate the source of the inefficiency a new definition of efficiency is needed because there can be one more than one Track related to one MCParticle

$$\varepsilon_{PTD} = \frac{\text{\# Particles with at least one related RecoTrack and one related PXD Digit inside a ROI}}{\text{\# Particles with at least one related RecoTrack and one related PXD Digit}}$$

PTD = Particle with Track and Digit

➡ The *track merger* inefficiencies do not influence the evaluation of the ROI efficiency ε_{PTD}

➡ The efficiency definition regarding PXDDigits is also modified:

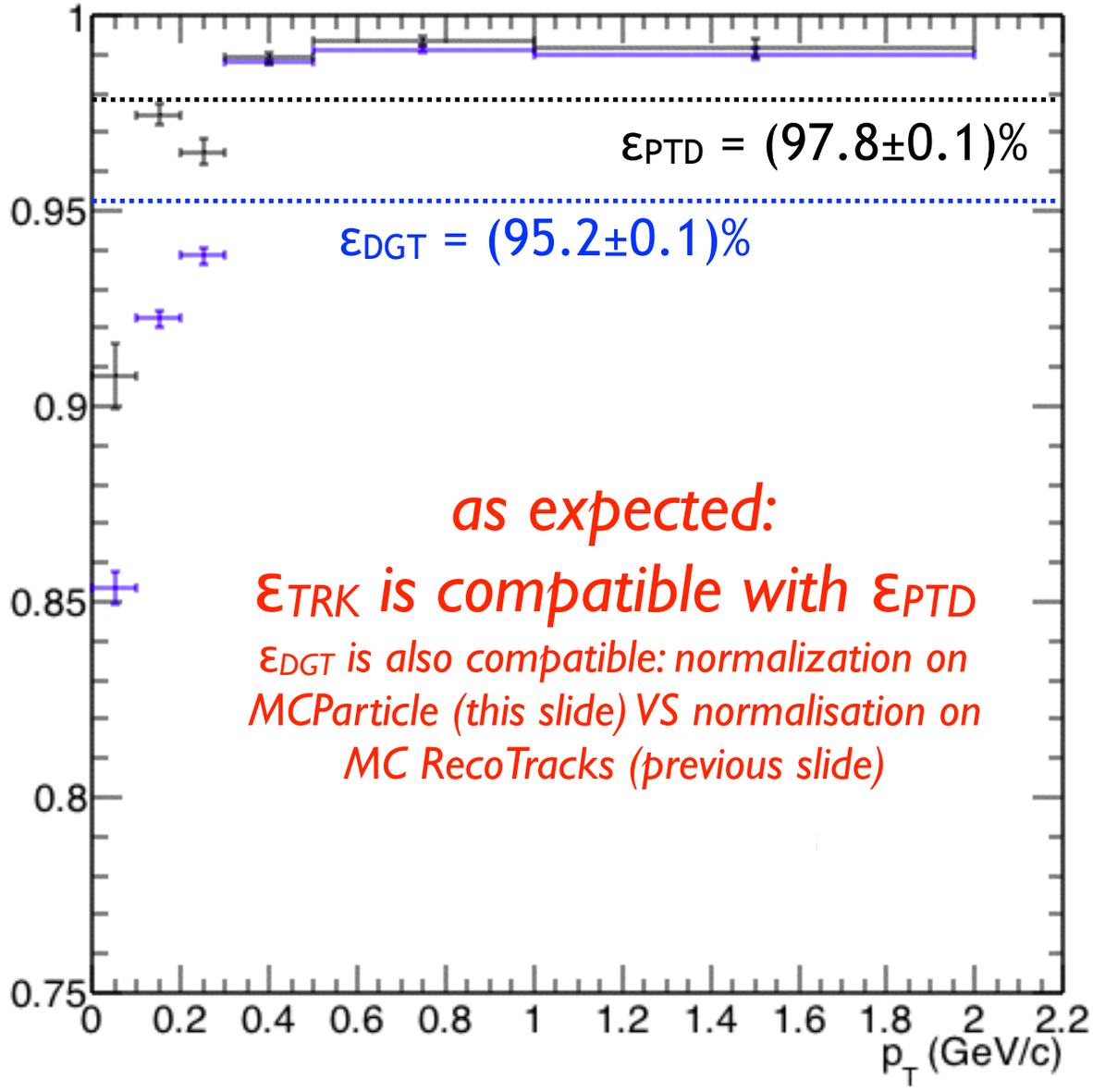
$$\varepsilon_{DGT} = \frac{\text{\# PXDDigits inside a ROI}}{\text{\# PXDDigits of MCParticle}}$$

➡ As a sanity check, compare ε_{PTD} with ε_{TRK} using the **MC track finder**: the results are perfectly compatible

- also ε_{DGT} not differ

➡ a new efficiency definition requires a new inefficiency classification

SVD+CDC MCTF



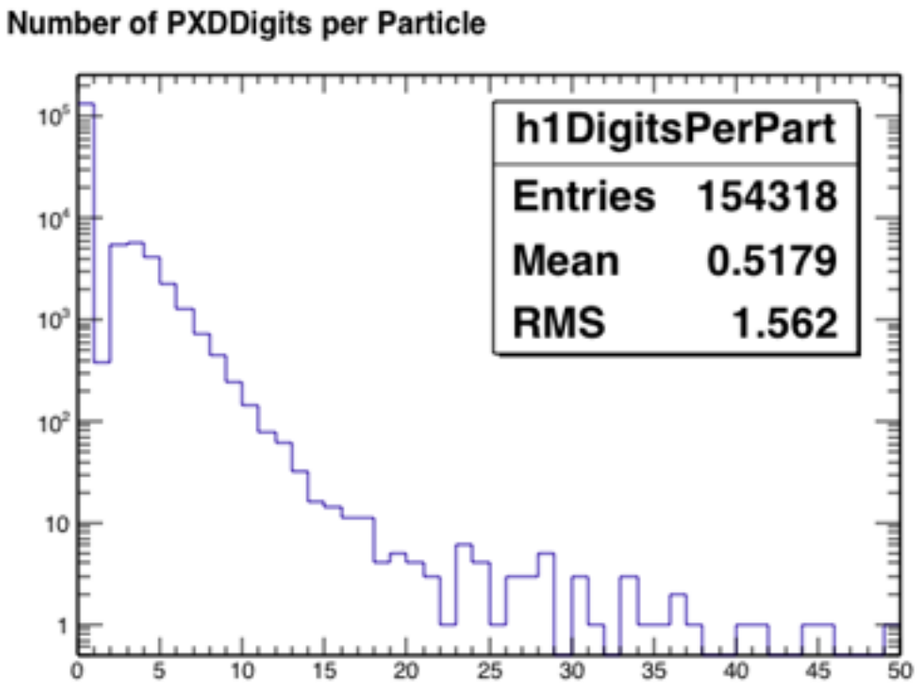
as expected:

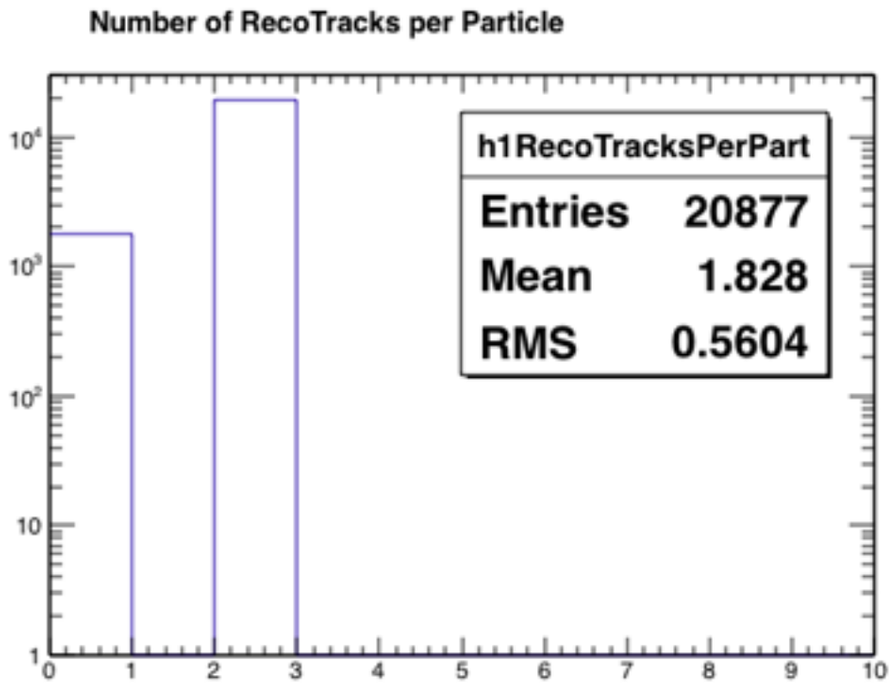
ε_{TRK} is compatible with ε_{PTD}

ε_{DGT} is also compatible: normalization on MCParticle (this slide) VS normalisation on MC RecoTracks (previous slide)

Inefficiency Classification

➡ Consider only MCParticle with at least one related PXDDigit and one related RecoTrack





➡ Loop on the PXDDigits related to the MCParticle

- *loop on all RecoTracks and for each RecoTrack loop on all intercepts*

  = (#intercepts x #recoTracks) cases to judge for each digit: choose the least serious case

➡ Classification of the MCParticle is the least serious case of its PXDDigits

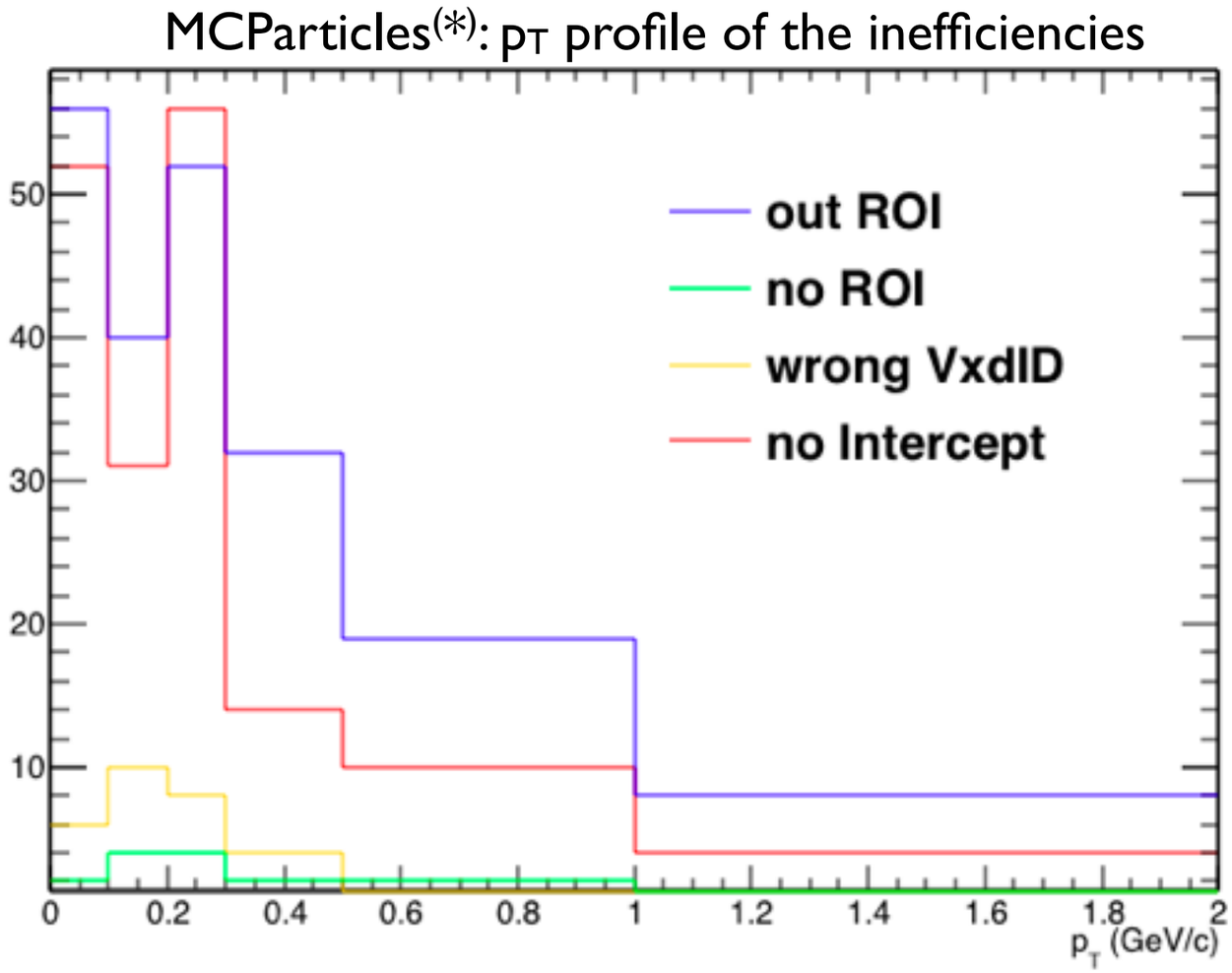
given a PXDDigit these are the possible cases:

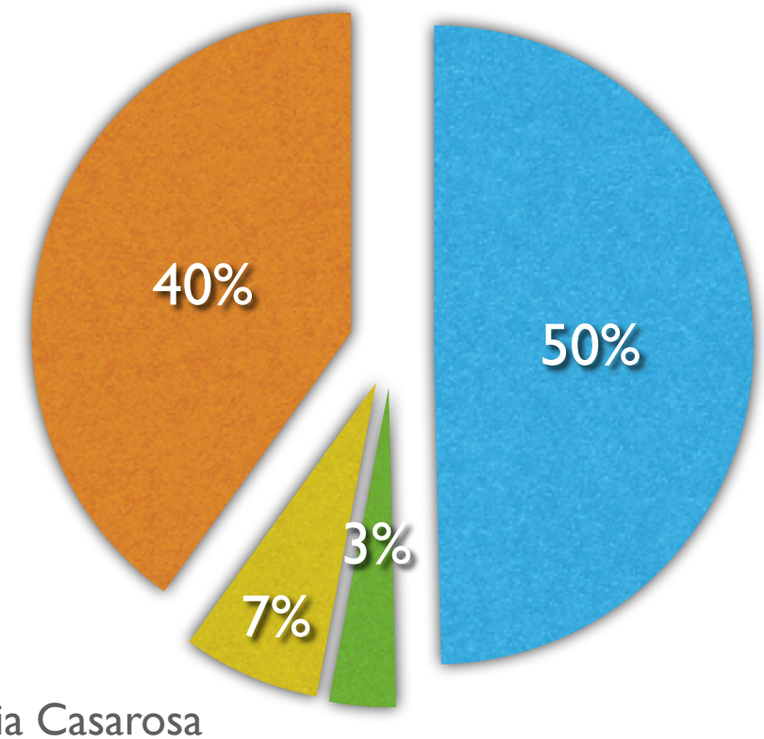
- digit is outside the ROI defined on the digit sensor
- no ROI defined on the digit sensor
- intercepts exists but none on the digit sensor
- intercept does not exist

least serious ↑ *most serious*

Inefficiency with MC TrackFinder

- In 2k events, 19084 MCParticles(*) are found, and **97.8% have at least one PXDDigit contained in an ROI**
- Only 417 suffer from an inefficient ROI finding and miss a PXDDigit
  - in half cases the ROI exist on the correct sensor but it does not include the PXDDigit
  - in 40% of the cases the intercept does not exist (165 MCParticles(*))





MCParticles(*) with No Digits in ROI:

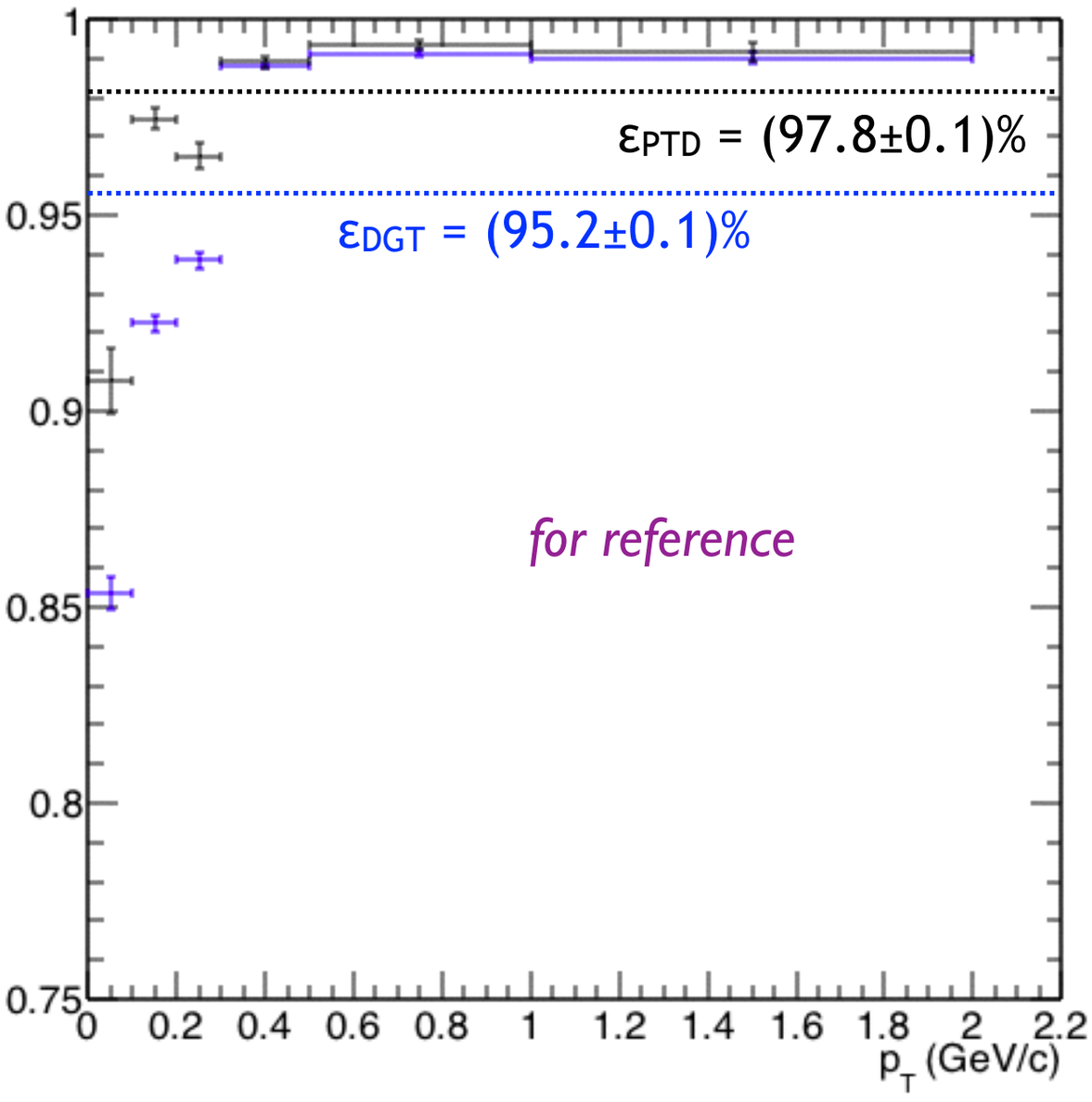
- digit is outside the ROI defined on the digit sensor
- no ROI defined on the digit sensor
- intercepts exists but none on the digit sensor
- intercept does not exist

(*) MCParticles with at least one related RecoTrack and one related PXDDigit

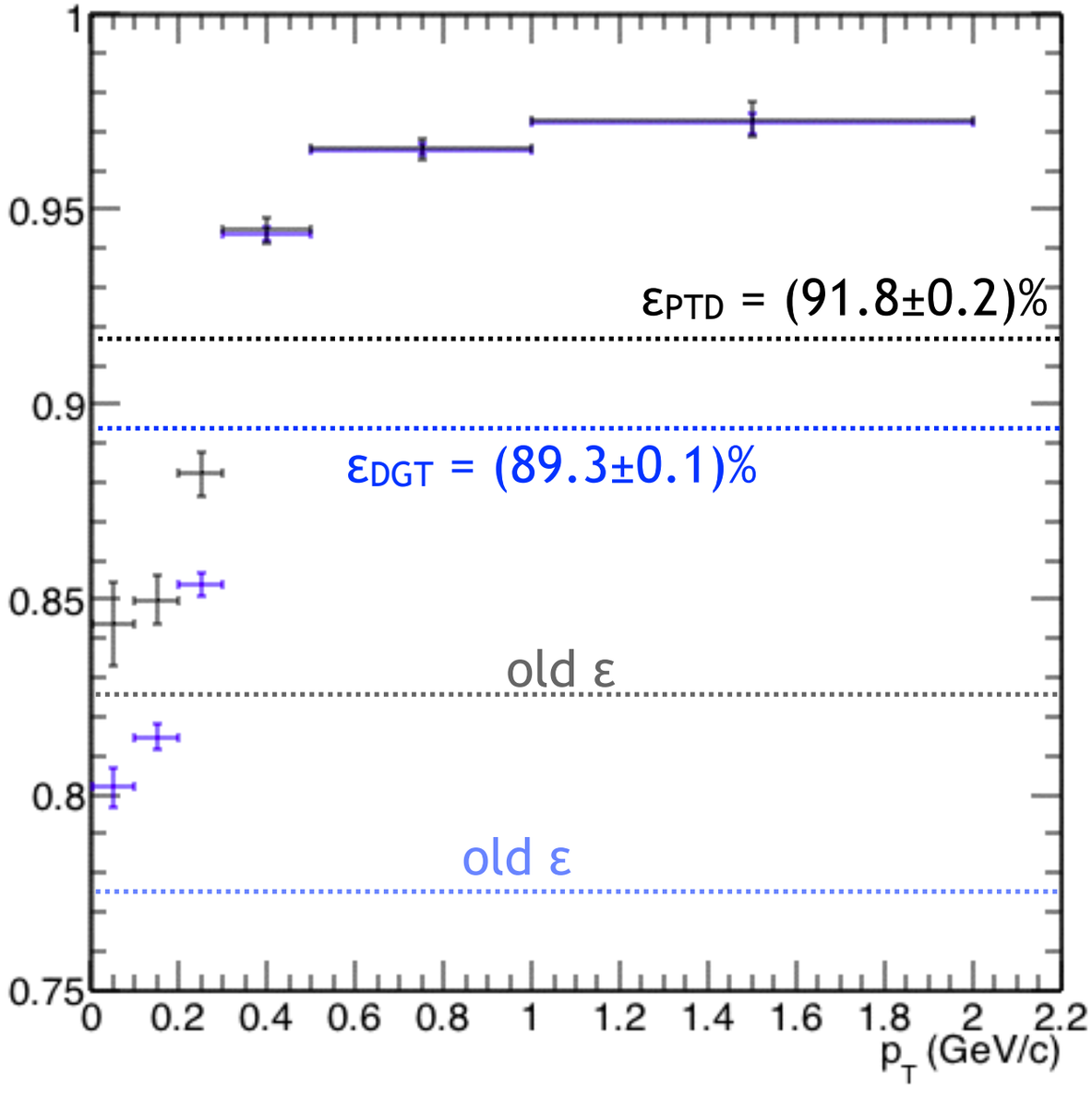
SVD+CDC real Track Finder

- With the MC TrackFinder the difference in definition in efficiency is negligible, as expected
- With the realTF, on the other hand, there is a significant improvement that confirms that **the issues with the merger were affecting the ROI efficiency estimation**

SVD+CDC MCTF

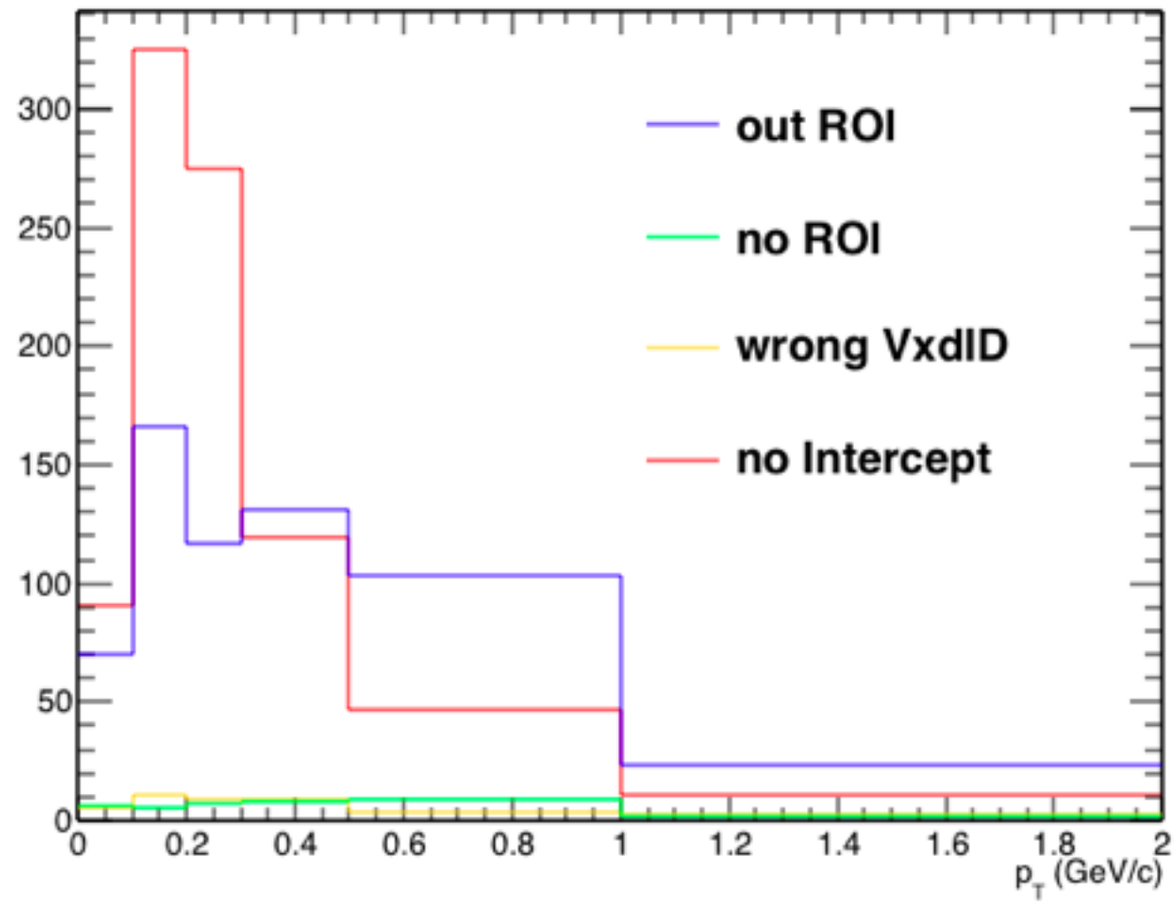


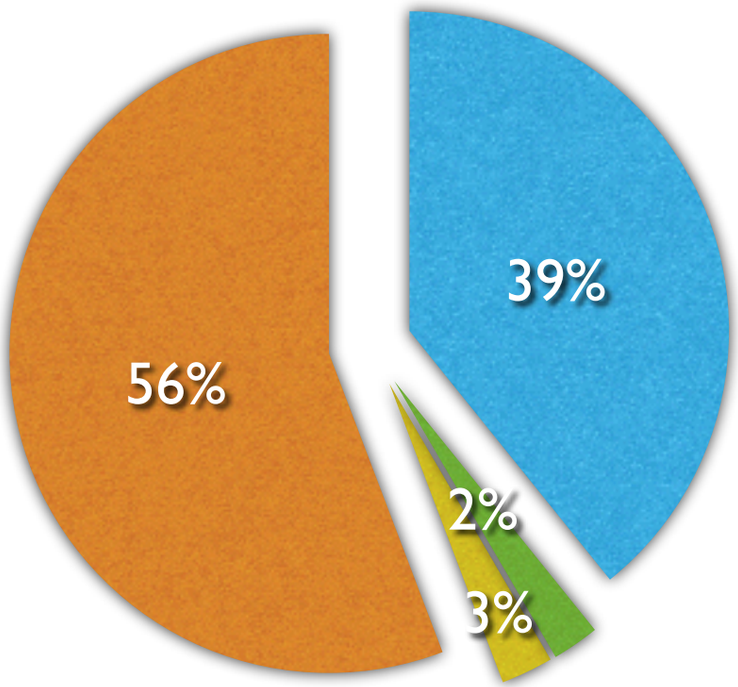
SVD+CDC realTF



Inefficiency with real TF

- In 2k events, 18963 MCParticles(*) are found, and 91.2% have at least one PXDDigit contained in an ROI
- Only 1556 suffer from an inefficient ROI finding and miss a PXDDigit
  - in 40% of the cases the ROI exist on the correct sensor but it does not include the PXDDigit
  - in 55% of the cases the intercept does not exist (868 MCParticles(*))





MCParticles(*) with No Digits in ROI:

- digit is outside the ROI defined on the digit sensor
- no ROI defined on the digit sensor
- intercepts exists but none on the digit sensor
- intercept does not exist

(*) MCParticles with at least one related RecoTrack and one related PXDDigit

Half - Presentation Conclusions

- The efficiency of 82.6% reported in the first slide was not correctly estimated: half of the inefficiency was due to CDC tracks not correctly merged to existing SVD tracks
- After correcting for this with the new definition of ε_{PTD}, the ROI finding efficiency with *real TrackFinding* and *not-optimized ROI settings* are:

$$\varepsilon_{PTD} = (91.8 \pm 0.2)\% \qquad \varepsilon_{DGT} = (89.3 \pm 0.1)\%$$

- The ROI efficiency with the MC TrackFinder is $\varepsilon_{PTD} = (97.7 \pm 0.1)\%$
- Roughly half of the inefficiency is due to **missing intercepts** related to MCParticles
  - the fraction of MCParticles with no intercept is reduced to 1/5 using the MCTF instead of the realTF
  - with realTF an increase of missing intercepts is observed for $100 < p_T < 300$ MeV/c
- The other half of the inefficiency is due to the **size of the ROI** since there is an ROI on the correct sensor, but the digit is not inside the ROI
  - changing the width of the ROIs the inefficiency should be significantly reduced

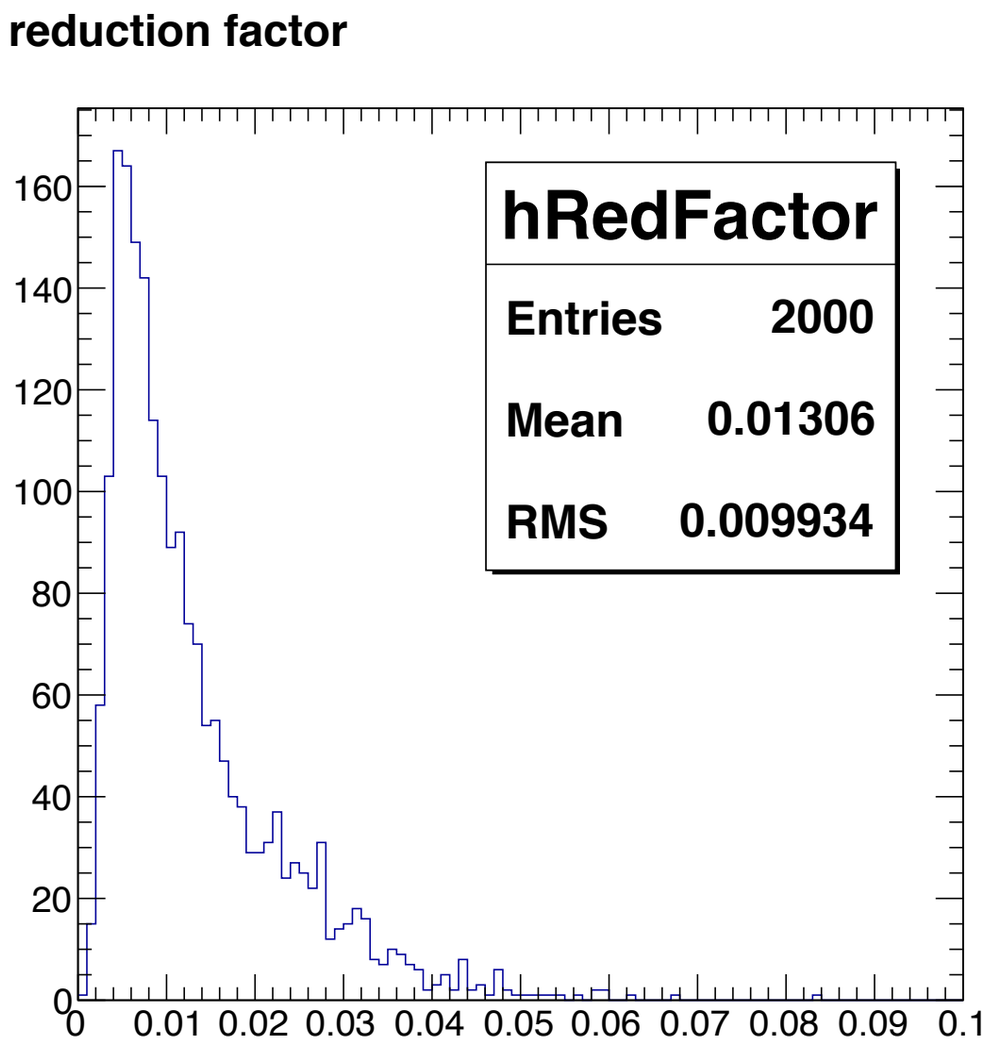
Reduction Factor

- The fraction of PXD sensors inside the ROIs (reduction factor) is well below the 10%
- An increase of the ROI size is possible and should improve the efficiency

- We can afford roughly a factor 10 increase of the area, which is not so much since the ROI are two-dimensional objects

- ROIs are rectangular regions, the standard size of the U and V sides are computed as:
  - 10 x $\sqrt{(syst^2 + stat^2)}$
  - stat = statistical error of the extrapolation
  - syst = 0.2 mm
  - max width U = 5 mm
  - max width V = 15 mm

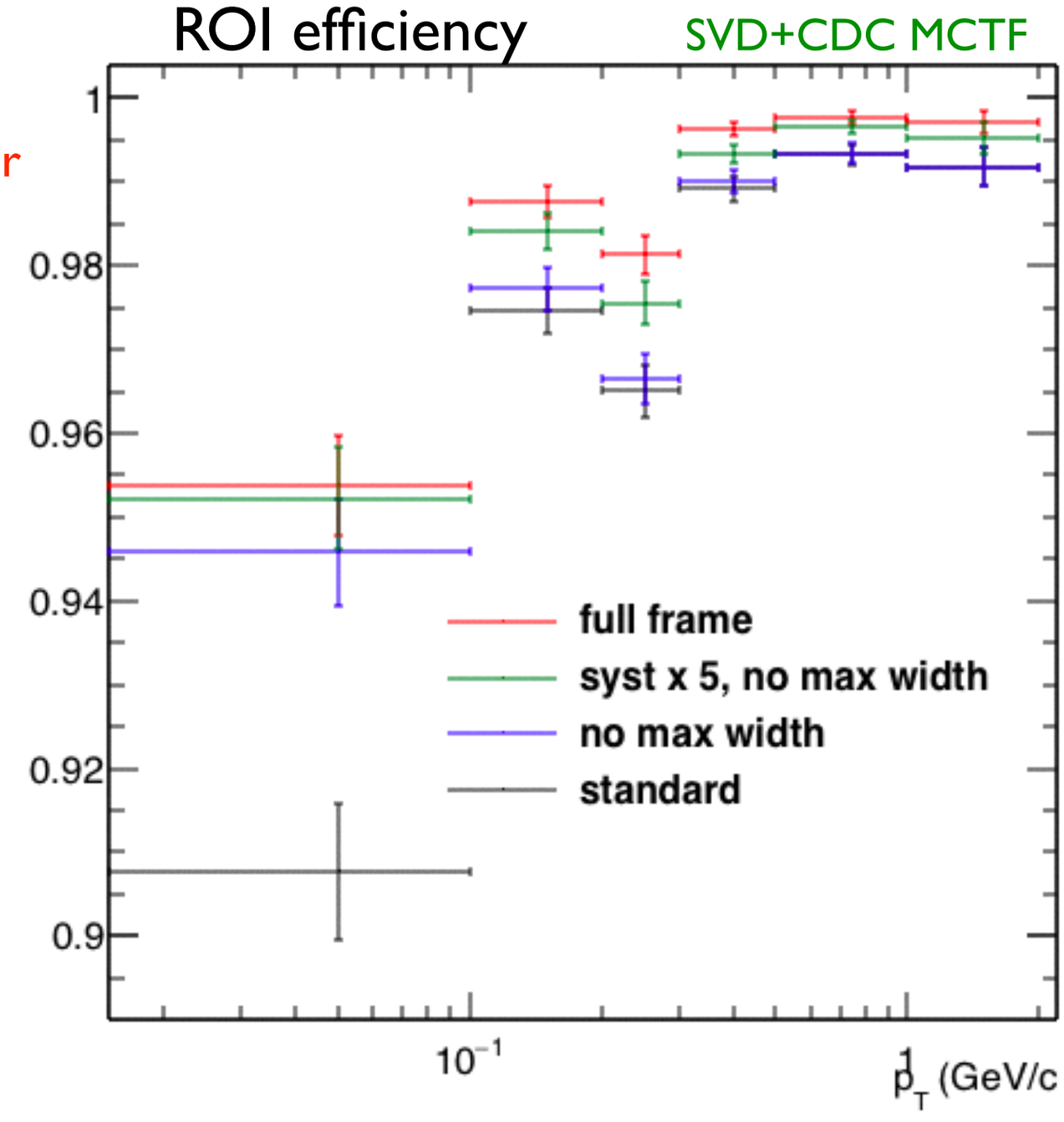
- I have played a bit with the parameters to see the effect on the efficiency



Playing with the ROI size, MCTF

- MCTF, full-frame ROIs
  - ε_{PTD} = (98.97±0.07)%
  - 0.15% have intercept on the wrong sensor
  - 0.88% miss the intercept
  - with perfect track-finding, this is the best we can achieve if we do not worry about the reduction factor…

- MCTF, removed max width
  - ε_{PTD} = (98.68±0.08)%
  - 0.28% have too small ROIs
  - 0.88% miss the intercept
  - can't afford this because of the reduction factor



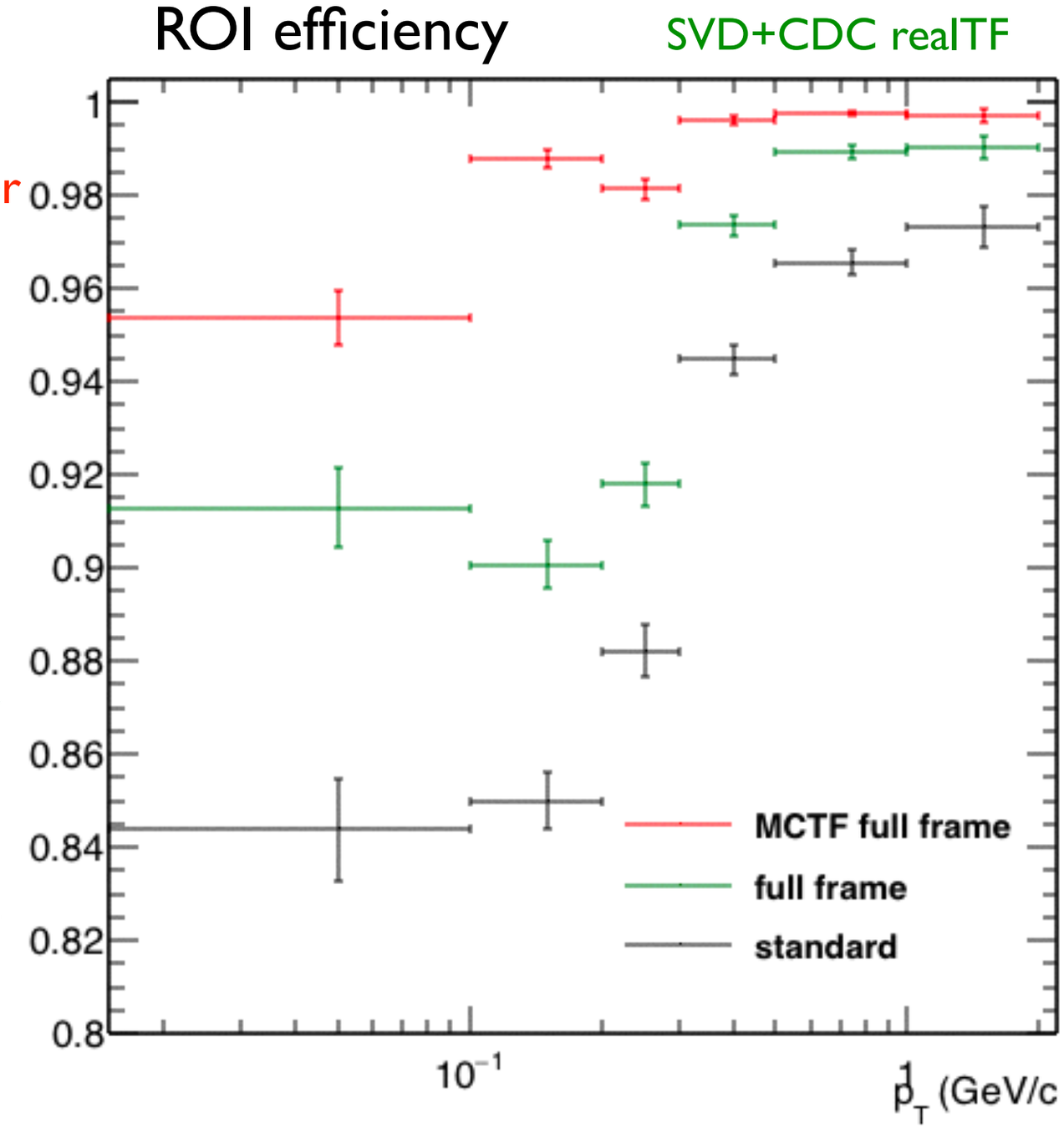
Full-FrameROI size, MCTF vs realTF

➡ **MCTF**, full-frame ROIs

- ε_{PTD} = **(98.97±0.07)%**
- 0.15% have intercept on the wrong sensor
- 0.88% miss the intercept
- with perfect track-finding, this is the best we can achieve if we do not worry about the reduction factor…

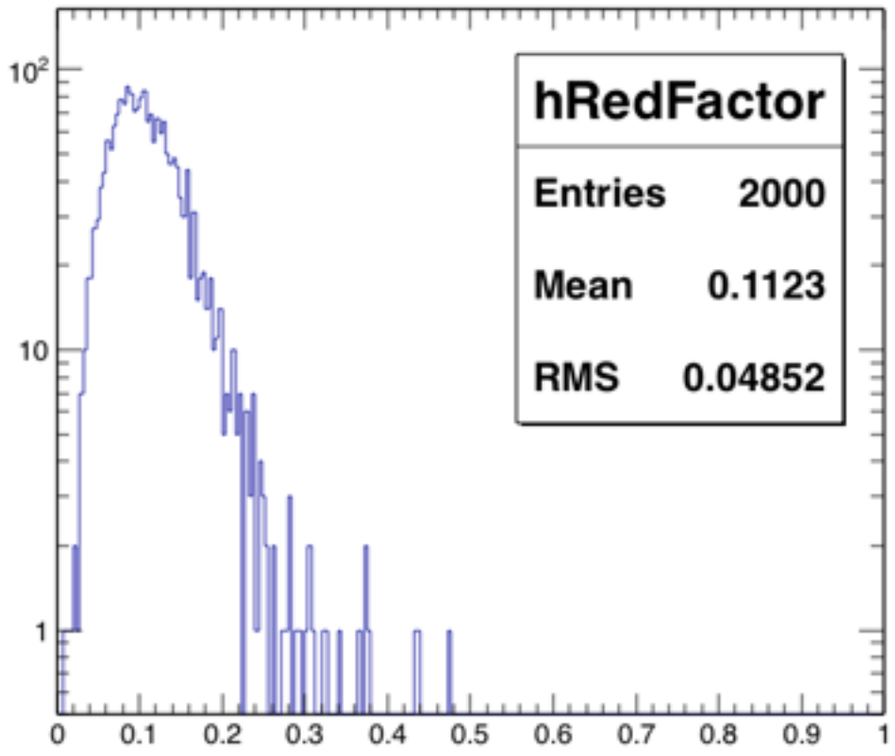
➡ **realTF**, full-frame ROIs

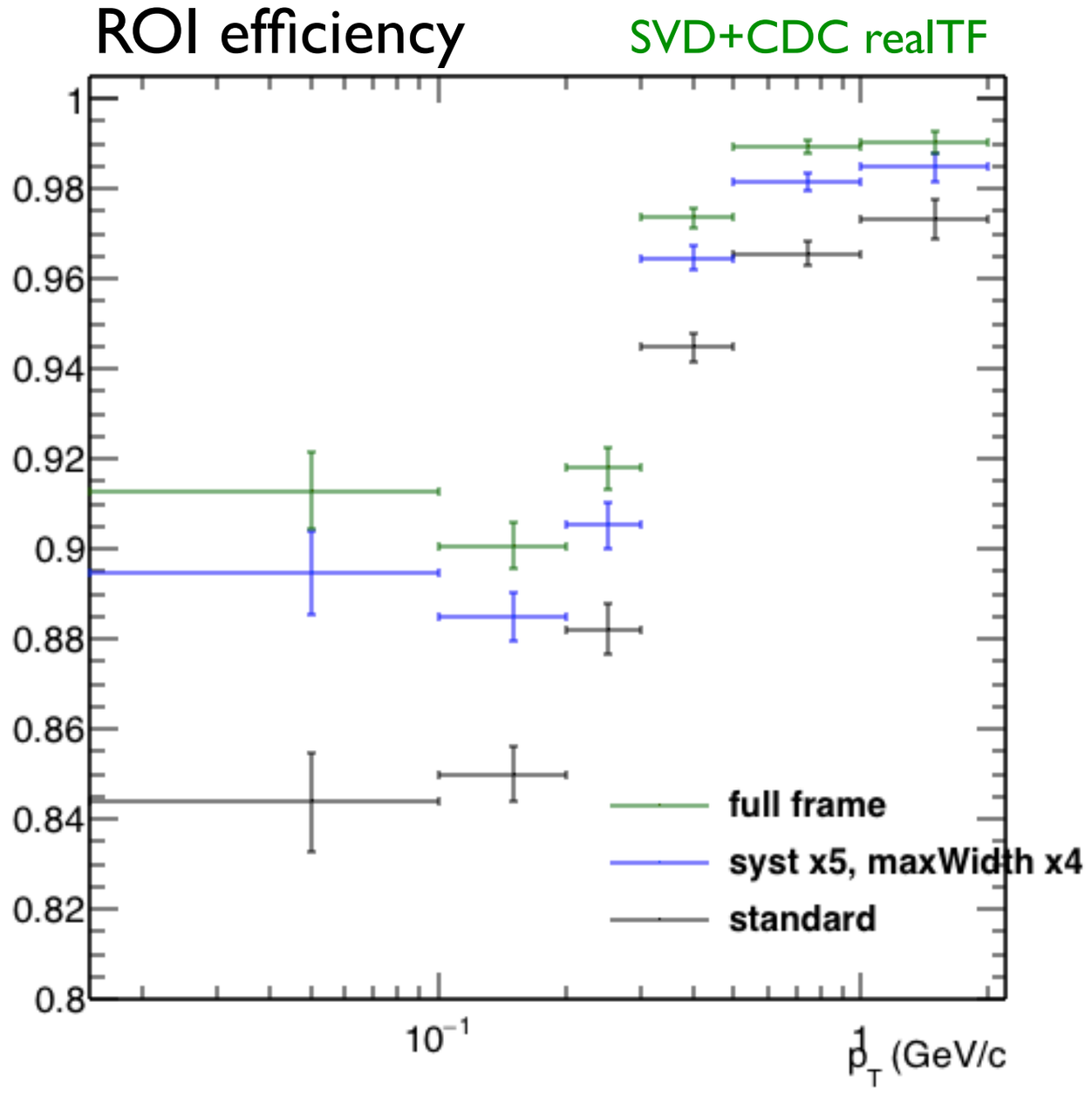
- ε_{PTD} = **(95.2±0.2)%**
- 0.2% have intercept on the wrong sensor
- 4.6% miss the intercept
- with our current tracking, this is the best we can achieve if we do not worry about the reduction factor…



A Reasonably Working ROI Setting

- ➡ real**TF**, larger ROIs but not too large
  - syst = 1mm (x5), maxWidth x4
  - ε_{PTD} = (94.1±0.1)%
  - 1% have too small ROIs
  - 4.6% miss the intercept
  - the reduction factor average is 11%, with an RMS of 5% and maximum below 50%





Conclusions

- After correcting for the wrong efficiency estimationn, the ROI finding efficiency with *real TrackFinding* and *not-optimized ROI settings* are:

ε_{PTD} = (91.8±0.2)% ε_{DGT} = (89.3±0.1)%

- The ROI efficiency with the MC TrackFinder is **ε_{PTD} = (97.7±0.1)%**
- Roughly half of the inefficiency is due to **missing intercepts** related to MCParticles
  - to be investigated
- The other half of the inefficiency is due to the **size of the ROI**
  - full-frame ROIs increase the efficiency up to ε_{PTD} = (95.2±0.2)% for the realTF and (98.97±0.07)% for the MC TF
  - a dedicated study is needed in order to optimise the efficiency keeping the reduction factor under control

PXDInterceptor (2)

current version

RecoTracks

```
void
PXDInterceptor::appendIntercepts(StoreArray<PXDIntercept>* listToBeFilled,
                                 std::list<ROIDetPlane> planeList, RecoTrack* recoTrack, int recoTrackIndex,
                                 RelationArray* recoTrackToPXDIntercepts) {
  PXDIntercept tmpPXDIntercept;
  genfit::Track& gfTrack = RecoTrackGenfitAccess::getGenfitTrack(*recoTrack);

  std::list<ROIDetPlane>::iterator itPlanes = planeList.begin();

  double lambda = 0;

  for (int propDir = -1; propDir <= 1; propDir += 2) {
    gfTrack.getCardinalRep()->setPropDir(propDir);

    while (itPlanes != planeList.end()) {
      genfit::MeasuredStateOnPlane state;
     try {
          state = gfTrack.getFittedState();
          lambda = state.extrapolateToPlane(itPlanes->getSharedPlanePtr());
      }  catch (...) {
        B2WARNING("extrapolation failed");
        itPlanes++;
        continue;
      }

      const TVectorD& predictedIntersect = state.getState();
      const TMatrixDSym& covMatrix = state.getCov();

      [ tmpPXDIntercept.set: CoorU CoorV, SigmaU, SigmaV, SigmaUprime, SigmaVprime, Lambda, VxdID ]

      listToBeFilled->appendNew(tmpPXDIntercept);

      gfTrackCandToPXDIntercepts->add(theGFTrackCandIndex, listToBeFilled->getEntries() - 1);

      itPlanes++;
    }
  }
};
```

can't change the propagation direction of a RecoTrack!

check both propagation directions

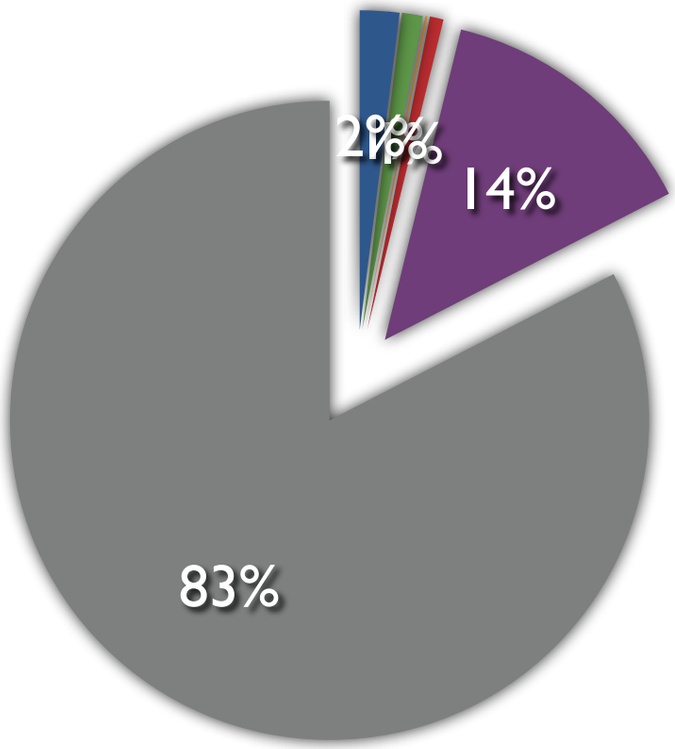
get the state of the track

extrapolate to the plane

compute the Intercept Infos

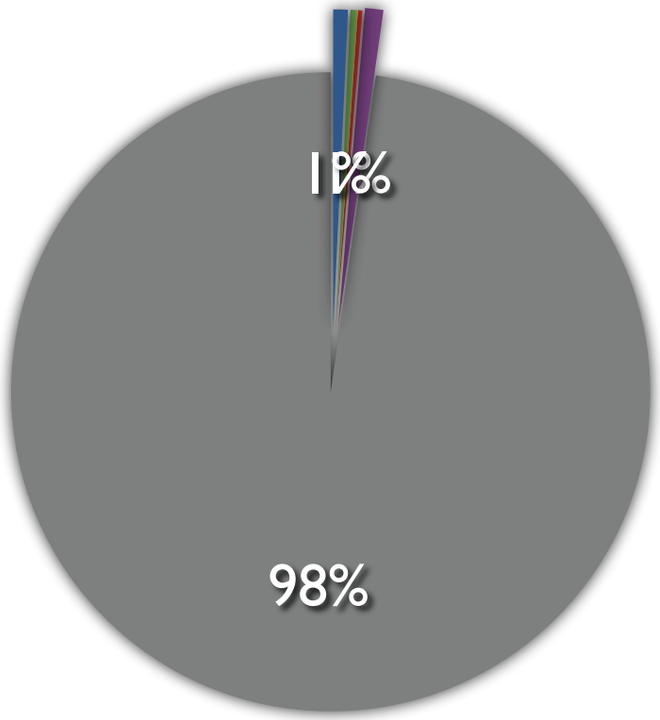
The TrackFinder (& Merger) Effect

real TrackFinder



- ROI with correct VxdID (213)
- ROI with wrong VxdID (112)
- no ROI, Intercept with correct VxdID (11)
- no ROI, Intercept with wrong VxdID (74)
- no Intercept (1454)
- ok: Digit inside ROI (8826)

MCTrackFinder



- ROI with correct VxdID (71)
- ROI with wrong VxdID (31)
- no ROI, Intercept with correct VxdID (5)
- no ROI, Intercept with wrong VxdID (22)
- no Intercept (89)
- ok: Digit inside ROI (9420)

Classification of Tracks with No Digits in ROI

- In 1k events, 10690 tracks have at least one related PXD Digit, 8826 tracks have at least one related PXD Digit contained in an ROI (**82.6%**).
- What about the other 1864 tracks (17.4%)?
- We need a track classification, but it's not straightforward!
  - *loop on each digit and for each digit, loop on all intercepts*

    = (#intercepts x #digits) cases to judge for each track, then choose a track status
- Classification of the Track is based on:
  - existence of intercept/ROI, intercept/ROI sensor (VxdID).
  - choose of the "least serious" problem among all the (#intercepts x #digits) cases

given a digit and an intercept, these are the possible cases:

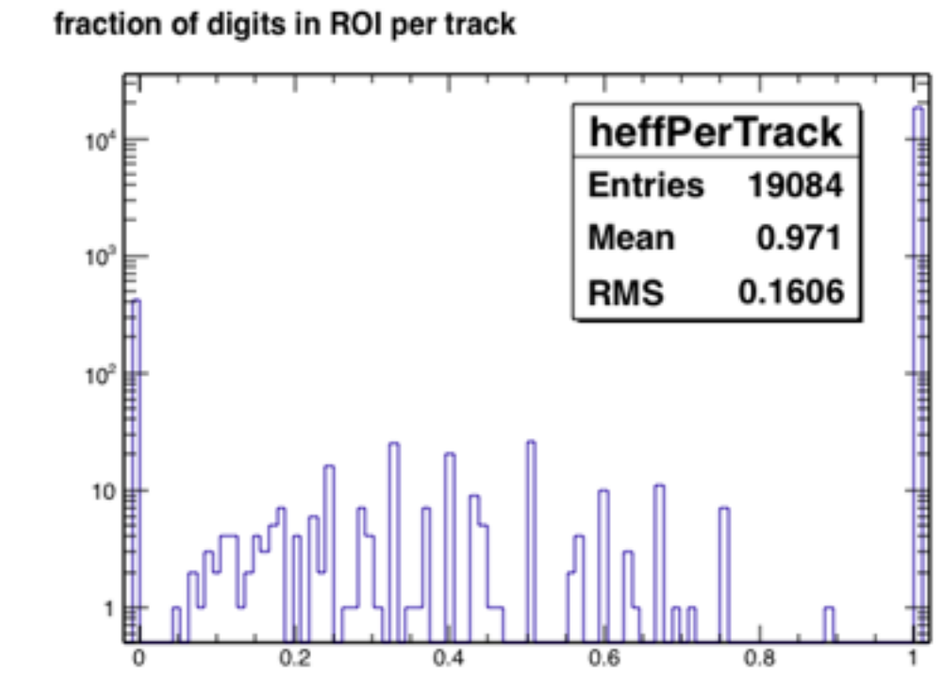
- ROI with correct VxdID
- ROI with wrong VxdID
- no ROI, Intercept with correct VxdID
- no ROI, Intercept with wrong VxdID
- no Intercept

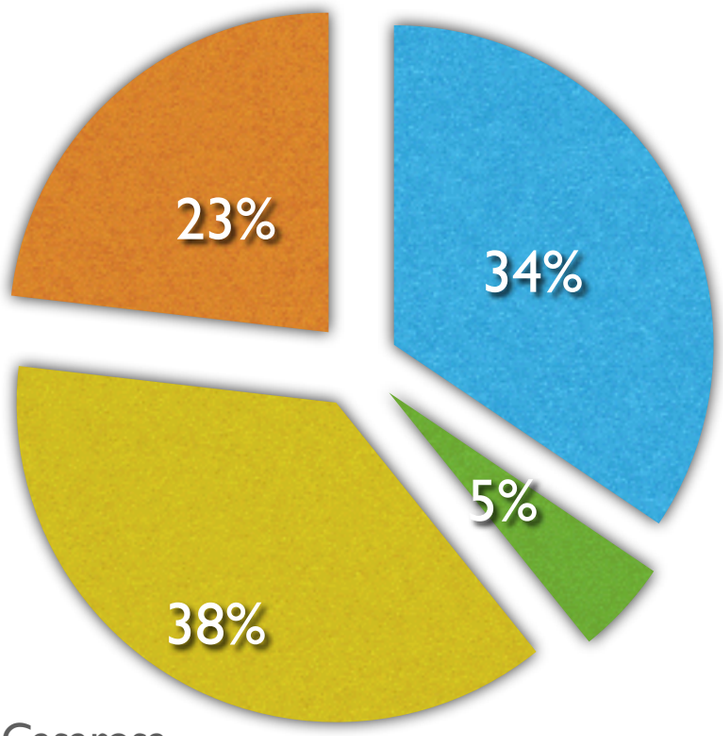
least serious

most serious

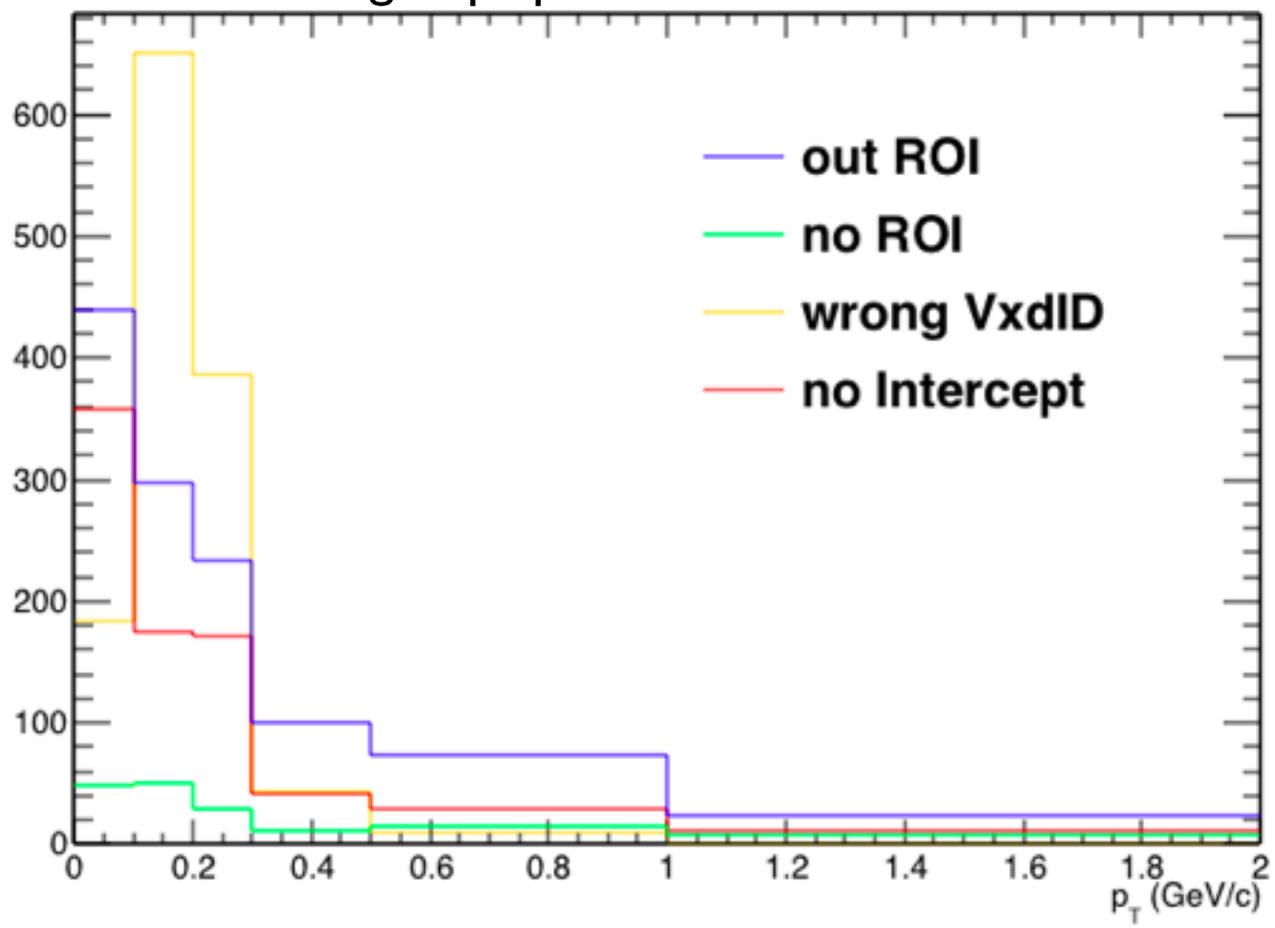
SVD+CDC MC TrackFinder: PXDDigits

- **95.2% PXDDigits related to MCParticles(*) are contained in an ROI**





PXDDigits: p_T profile of the inefficiencies



PXDDigits not in ROI:

- digit is outside the ROI defined on the digit sensor
- no ROI defined on the digit sensor
- intercepts exists but none on the digit sensor
- intercept does not exist

(*) MCParticles with at least one related RecoTrack and one related PXDDigit