ROI FINDING EFFICIENCY

Giulia Casarosa

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Alexander von Humboldt Stiftung/Foundation

Virtual Tracking Meeting ~ January 13th 2017

Outline

Plans Towards a "Maintenance Mode"

shown @ last F2F DESY meeting

➡ What is completely missing:

- *validation* plots for the simulation:
  - efficiency vs p_T and polar angle
  - reduction factor
  - mean and RMS of the distribution of the number of ROIs per module
  - statistical error of the intercept in the two directions, per layer

➡ What I would like to improve:

- efficiency (…)
- speed ✓
- carefully check the tracking script

```
add_tracking_for_PXDDataReduction_simulation(path, components=None, skipGeometryAdding=False)
```

➡ What needs to be re-thought:

- DQM plots: most of the validation plots + something else

ROI-Finding Performance Review

➡ Use the standard SVD+CDC tracking

- SVD & CDC pattern recognition algorithms
- merging of SVD and CDC tracks (→ SVD+CDC tracks, SVD tracks, CDC tracks)
- track fitting

➡ Feed the tracks into the PXDDataReduction Module

- determine the PXD planes toward which extrapolate

(*) • extrapolate the track towards these planes (`PXDInterceptor`)

- determine the position and widths of the ROI (in cm, local position)

(*) • translate the bottom-left and the top-right corners positions in pixels cell IDs of the sensor (`ROIPixelTranslator`)

- if the ROI is overlapping with the sensor, then it is appended to the final list

➡ Performance of the Module

- consider only tracks with related PXD Digits (true also in this presentation)

(*) • efficiency = fraction track-related PXDDigits contained in ROIs

(*) = to be discussed today

PXDInterceptor (I)

old version

genfit::Tracks

```
void
ROIGeometry::appendIntercepts(StoreArray<PXDIntercept>* listToBeFilled,
                              genfit::Track* theTrack, int theGFTrackCandIndex,
                              RelationArray* gfTrackCandToPXDIntercepts){
  PXDIntercept tmpPXDIntercept;

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

  double lambda = 0;
  for (int propDir = -1; propDir <= 1; propDir += 2) {
    theTrack->getCardinalRep()->setPropDir(propDir);

    while (itPlanes != m_planeList.end()) {

      genfit::MeasuredStateOnPlane state = theTrack->getFittedState();

     try {
          genfit::SharedPlanePtr plane(new ROIDetPlane(*itPlanes));
          lambda = state.extrapolateToPlane(plane);
      }  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++;
    }
  }
};
```

check both propagation directions

get the state of the track

extrapolate to the plane

compute the Intercept Infos

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

PXDInterceptor (3)

future version ?

RecoTracks

```
void
PXDInterceptor::appendIntercepts(StoreArray<PXDIntercept>* listToBeFilled,
                                 std::list<ROIDetPlane> planeList, RecoTrack* recoTrack, int recoTrackIndex,
                                 RelationArray* recoTrackToPXDIntercepts) {
  PXDIntercept tmpPXDIntercept;

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

  double lambda = 0;
```

don't change the propagation direction

```
   while (itPlanes != m_planeList.end()) {
```

get the state from the FIRST hit

```
     genfit::MeasuredStateOnPlane state = recoTrack->getMeasuredStateOnPlaneFromFirstHit();

   try {
        genfit::SharedPlanePtr plane(new ROIDetPlane(*itPlanes));
        lambda = state.extrapolateToPlane(plane);
      }  catch (...) {
        B2WARNING("extrapolation failed");
        itPlanes++;
        continue;
      }
```

extrapolate to the plane

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

compute the Intercept Infos

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

      listToBeFilled->appendNew(tmpPXDIntercept);

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

      itPlanes++;
    }
  }
};
```

Preliminary studies on a limited sample show that the impact on efficiency is negligible

Efficiency Definition

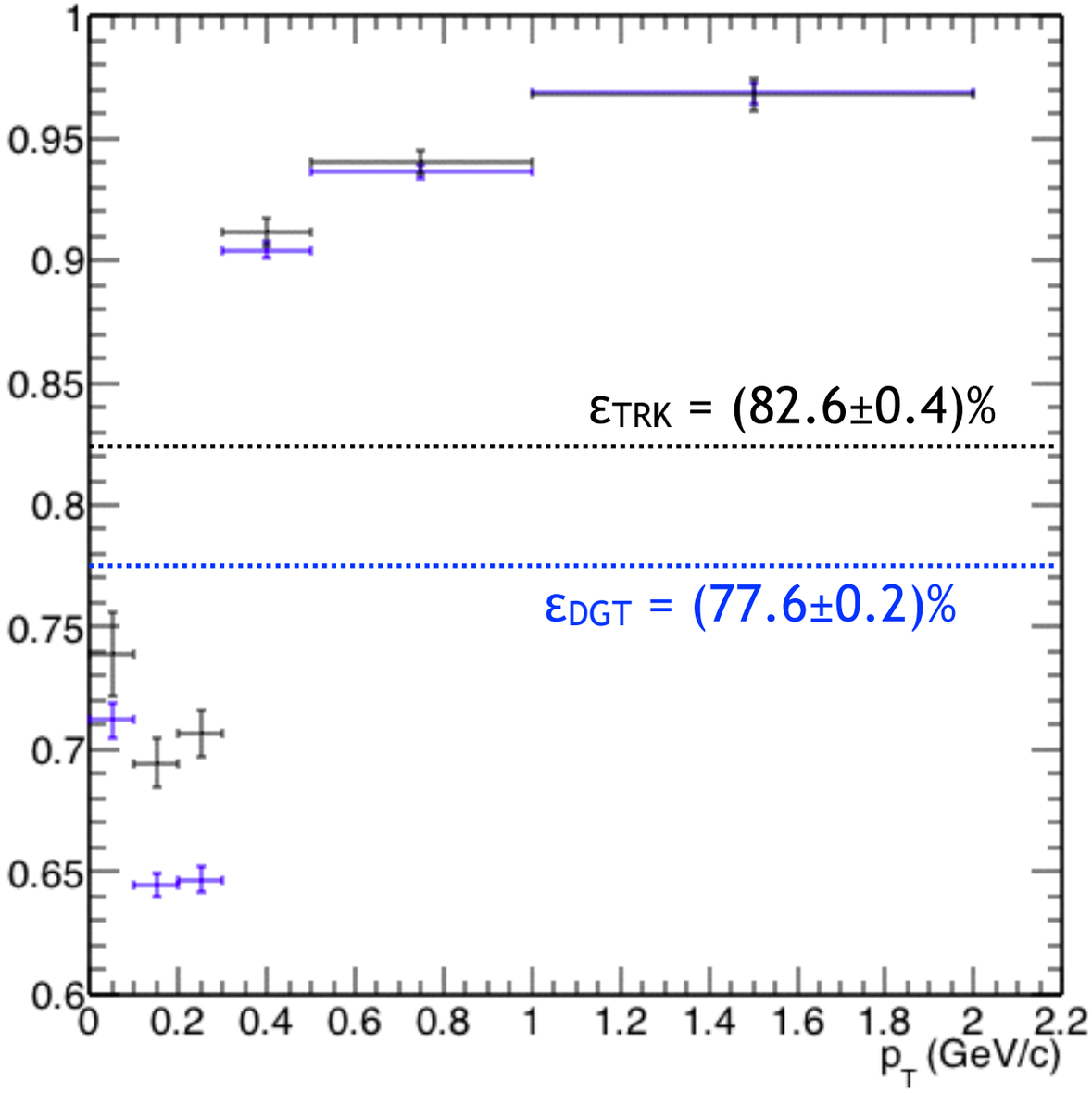
- The current definition of efficiency does not focus on what is actually important for the physics performances

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

- The real interesting thing is how many tracks have at least one related pixel inside an ROI
- Define a new efficiency:

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

- Where is the inefficiency in ε_{TRK} coming from?
  - hp: the track is not "good" enough to be used to estimate the position of the intercepts with the PXD planes



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

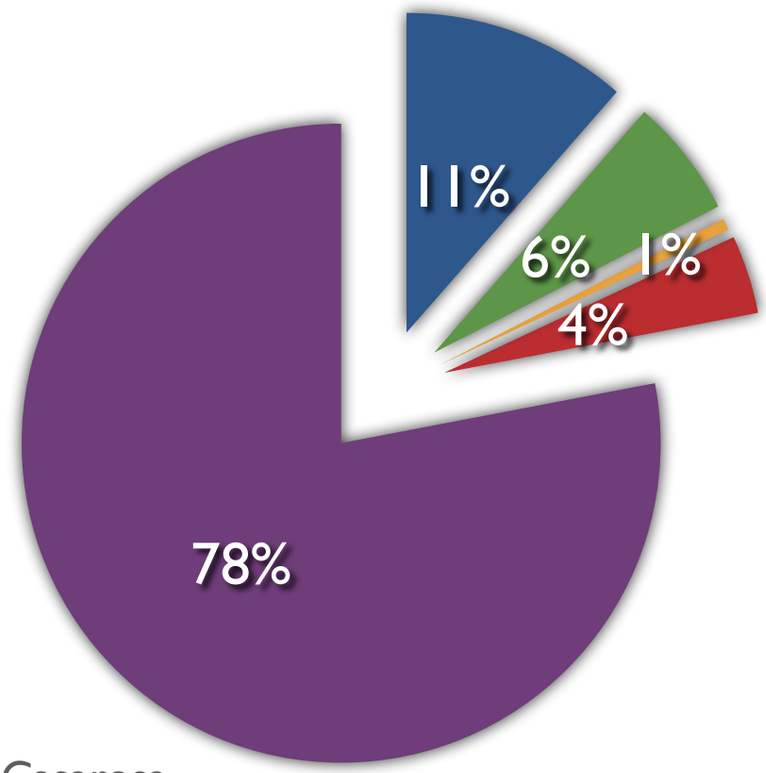
least serious

most serious

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 .
  - choose of the "least serious" problem among all the (#intercepts x #digits) cases



Tracks with No Digits in ROI:

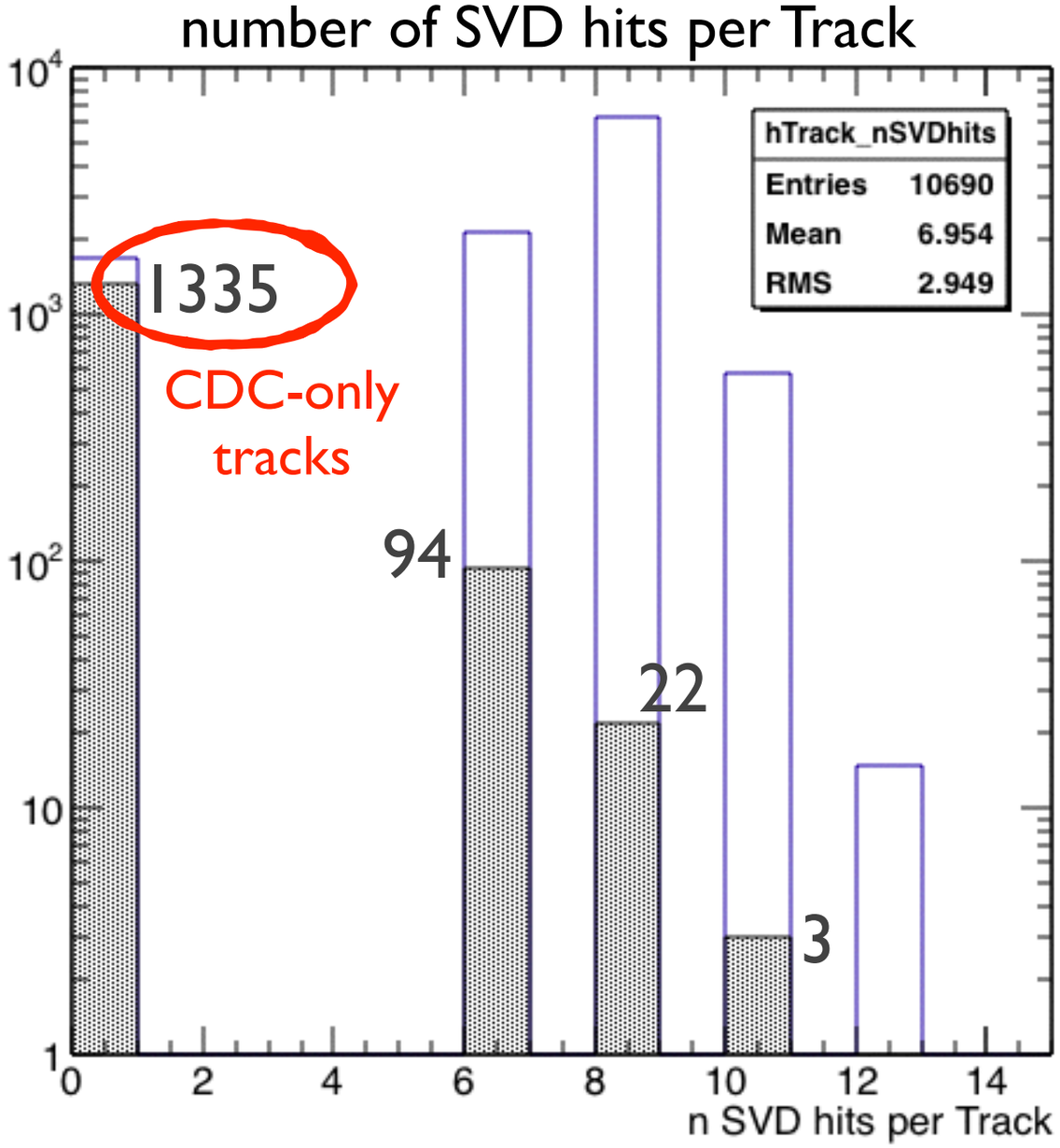
- ROI with correct VxdID (213)
- ROI with wrong VxdID (112)
- no ROI, Intercept with correct VxdID (11)
- no ROI, Intercept with wrong VxdID (74)
- no Intercept (1454)

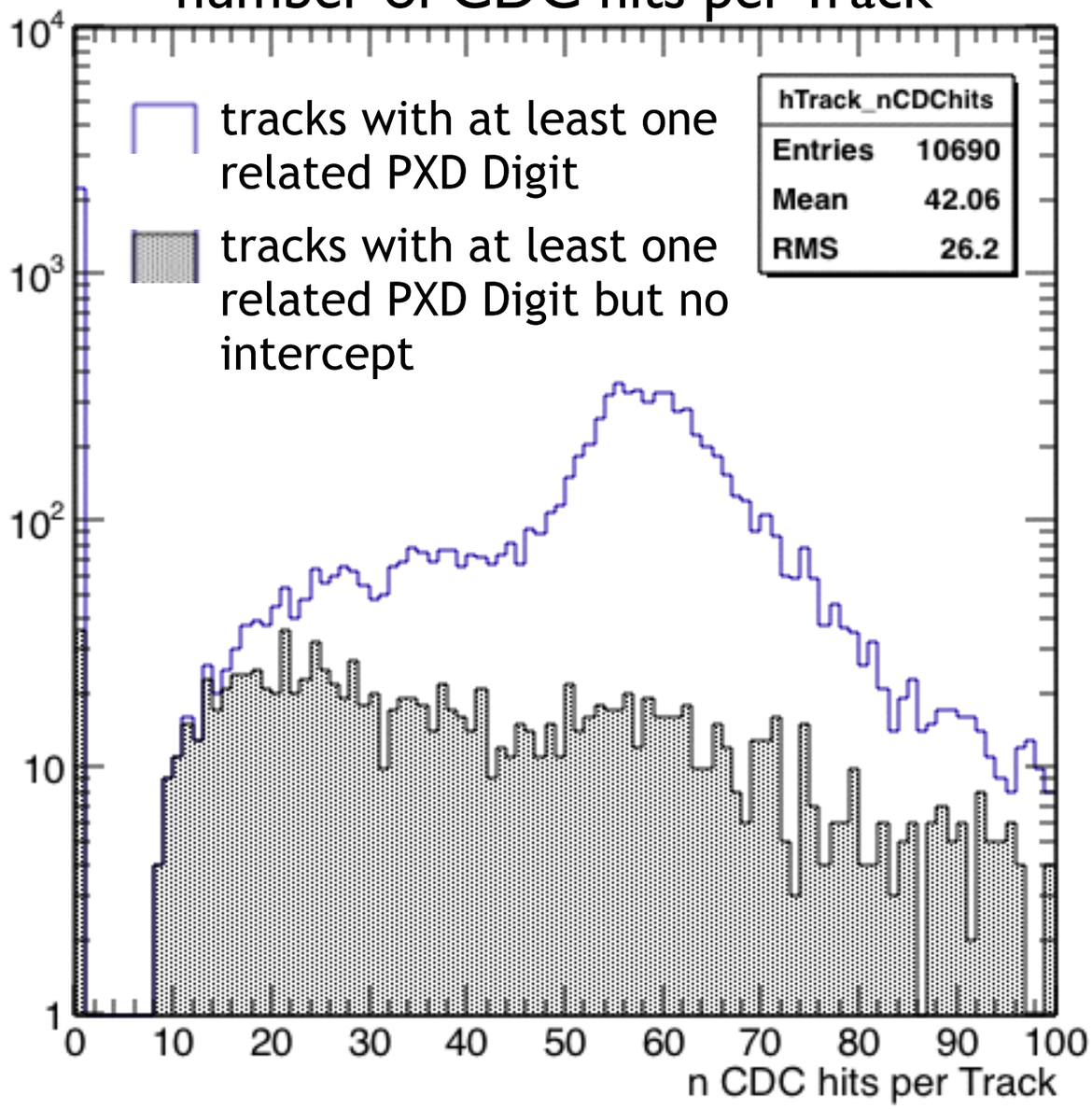
least serious

most serious

Tracks with No Intercepts

- CDC-only tracks represent the 92% of the tracks with no associated intercepts with the PXD planes

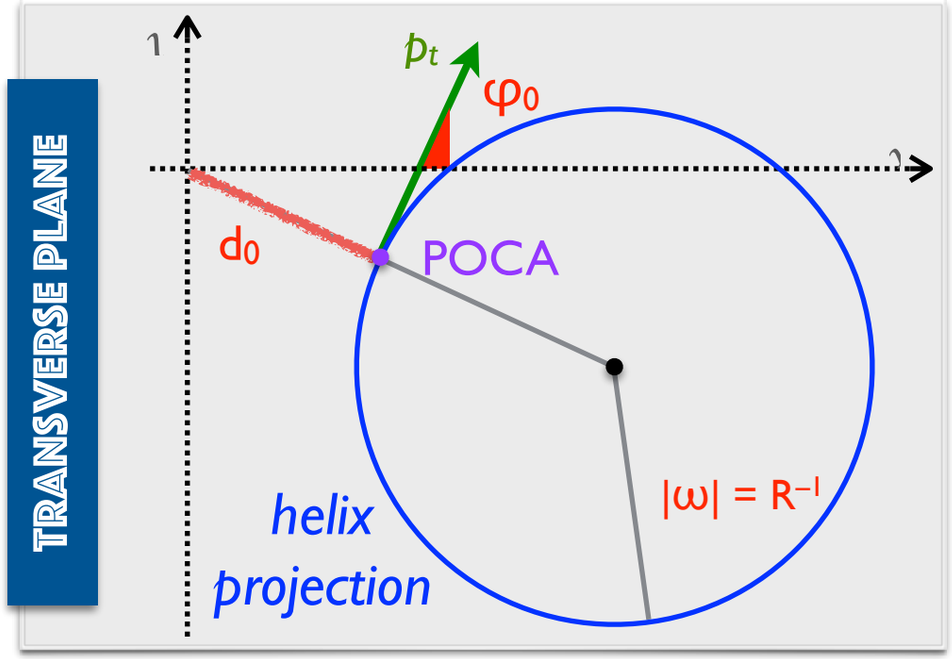




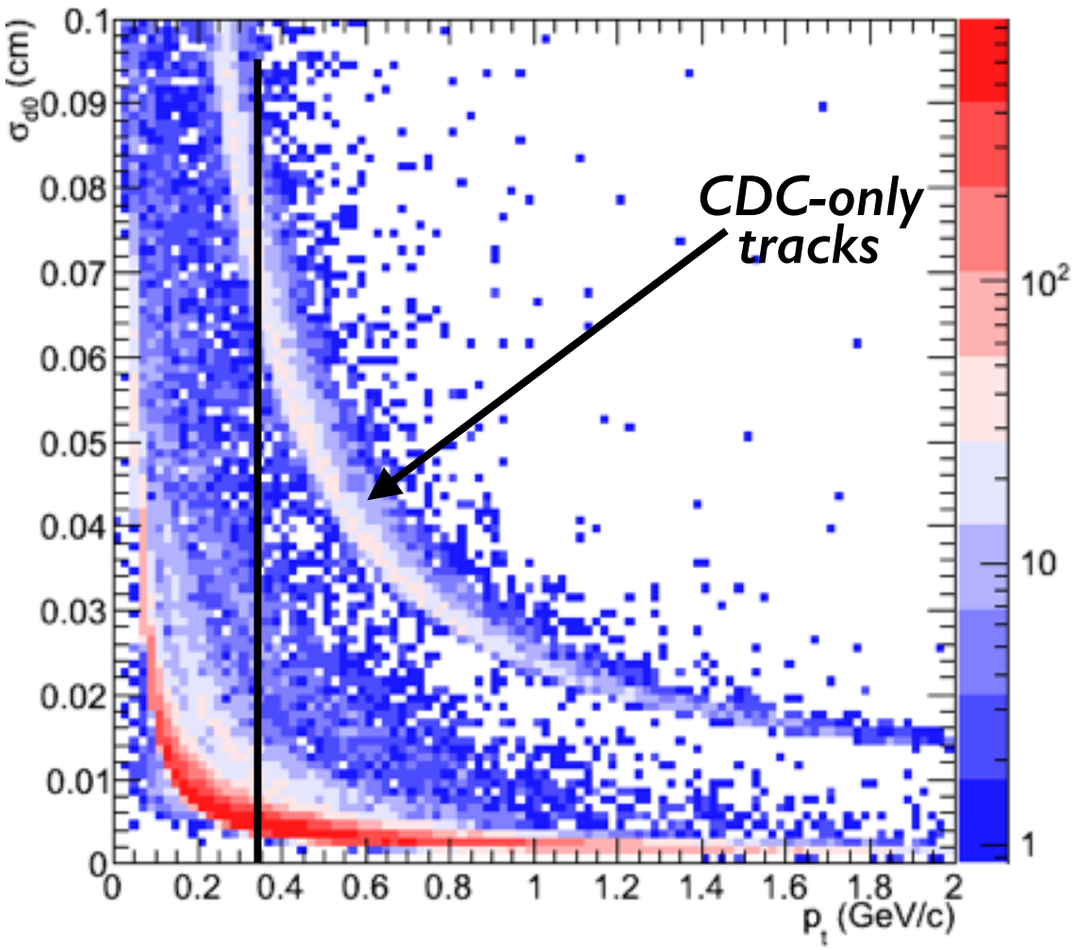
Tracks with No Intercepts

➡ CDC-only Tracks

- The most important tracking parameters for the definition of the ROI are the *impact parameters*, the CDC resolution on the impact parameters is not enough at low p_T



σ_{d0} resolution vs p_T



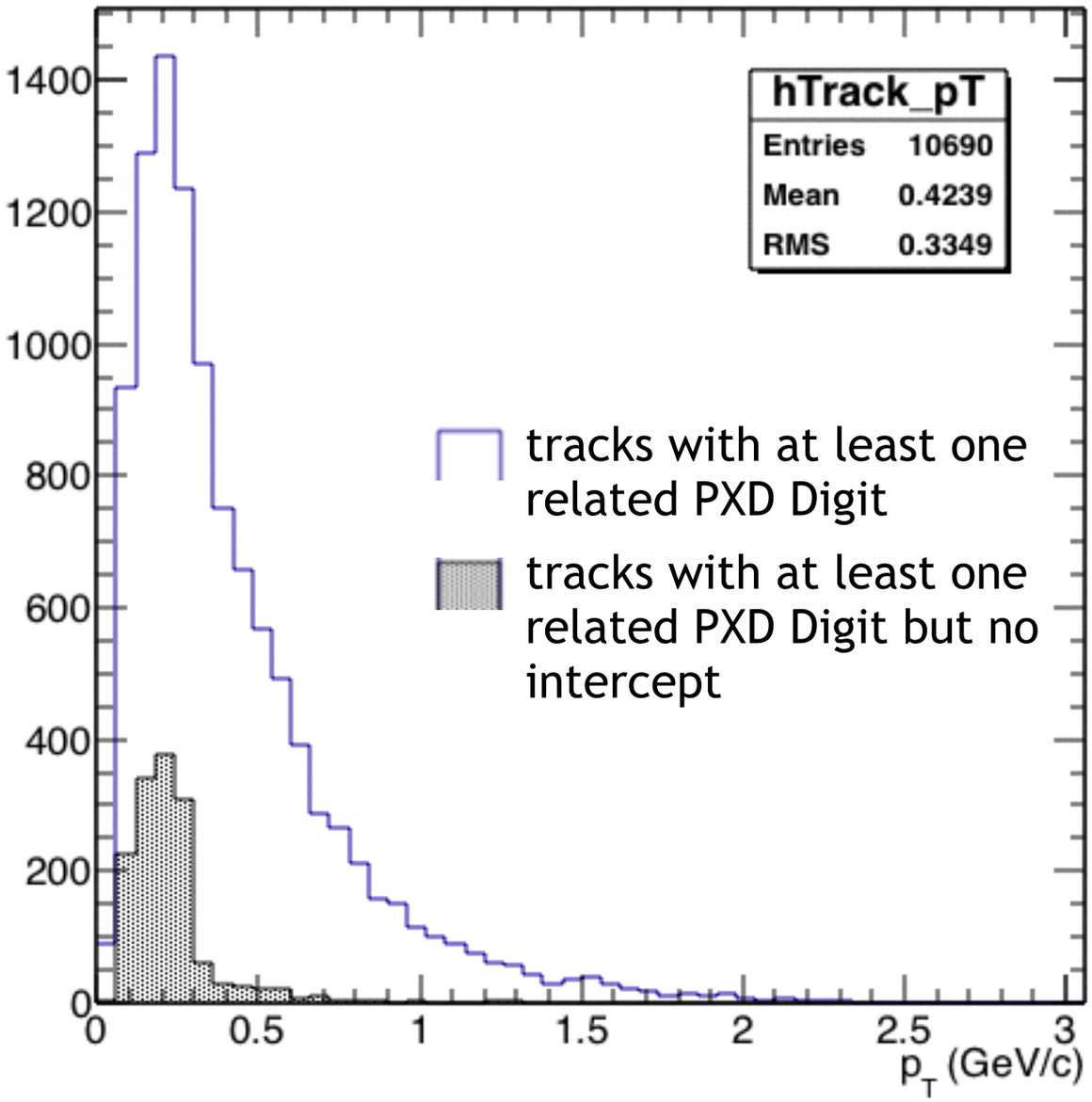
reminder: all considered tracks have at least one associated PXD Digit

➡ SVD track-finding does not find these patterns, WHY ?

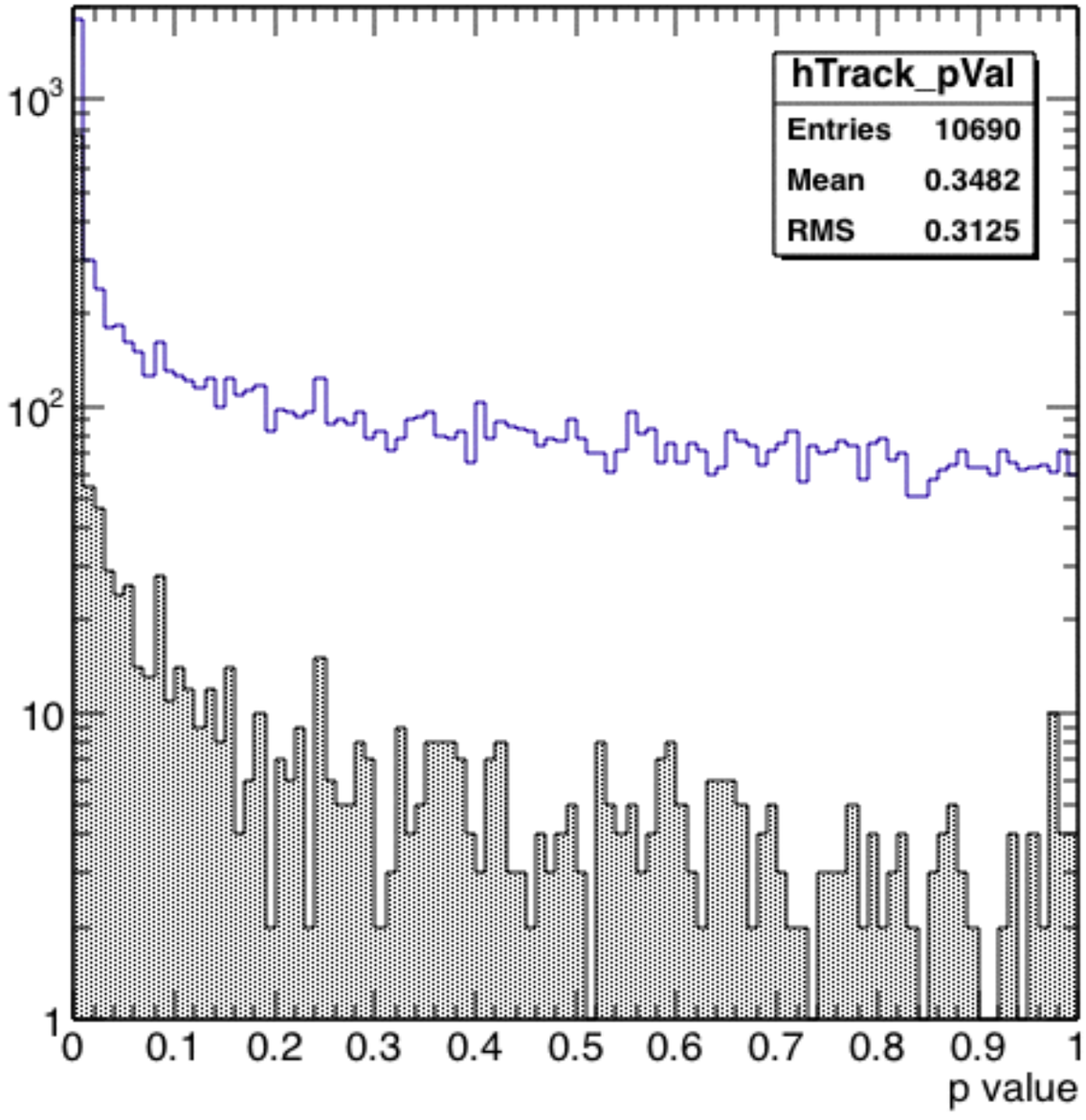
- these tracks have more than 5 hits in the SVD

p_T and p-value of the No-Intercept Tracks

transverse momentum distribution



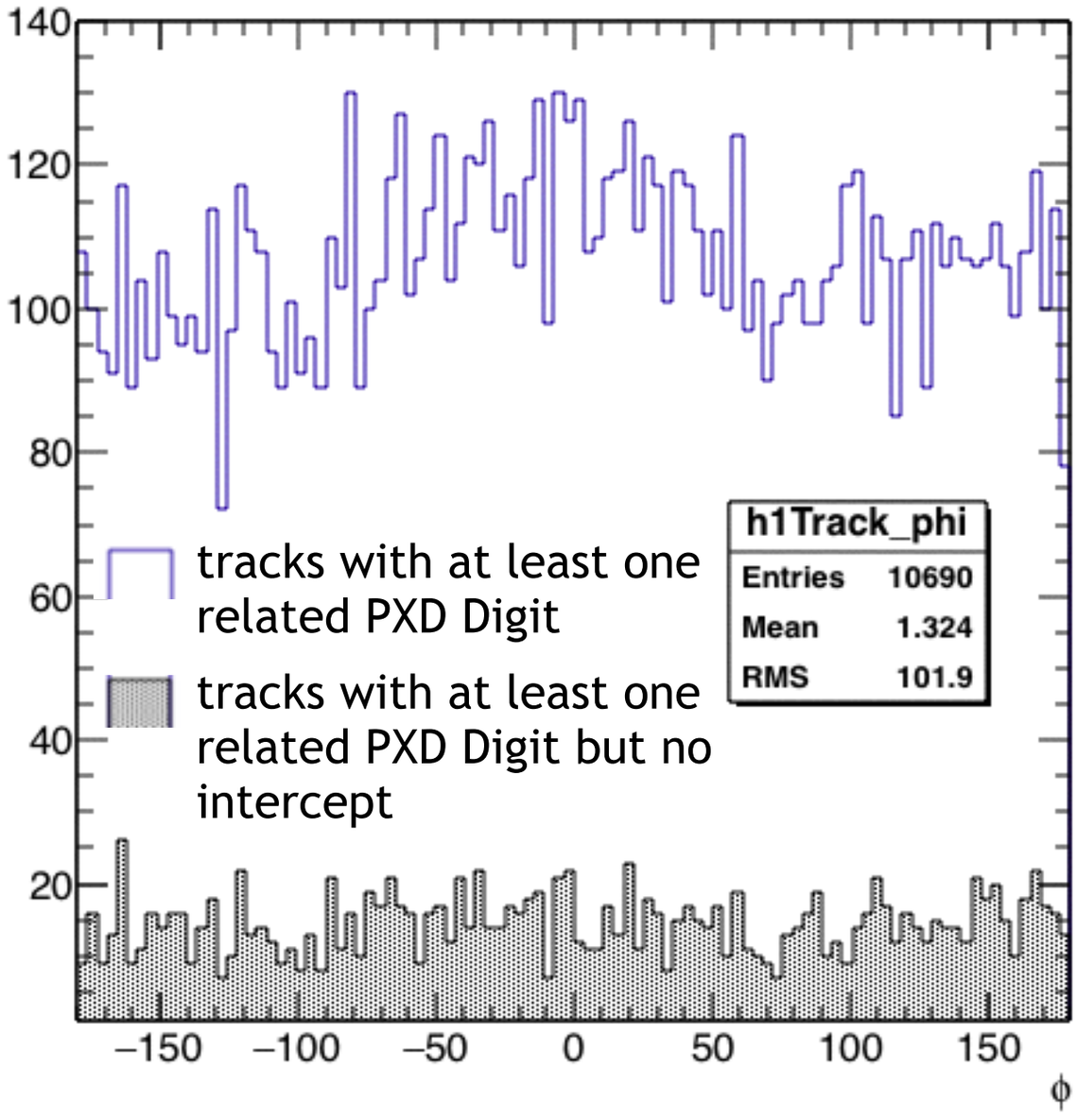
p-value distribution



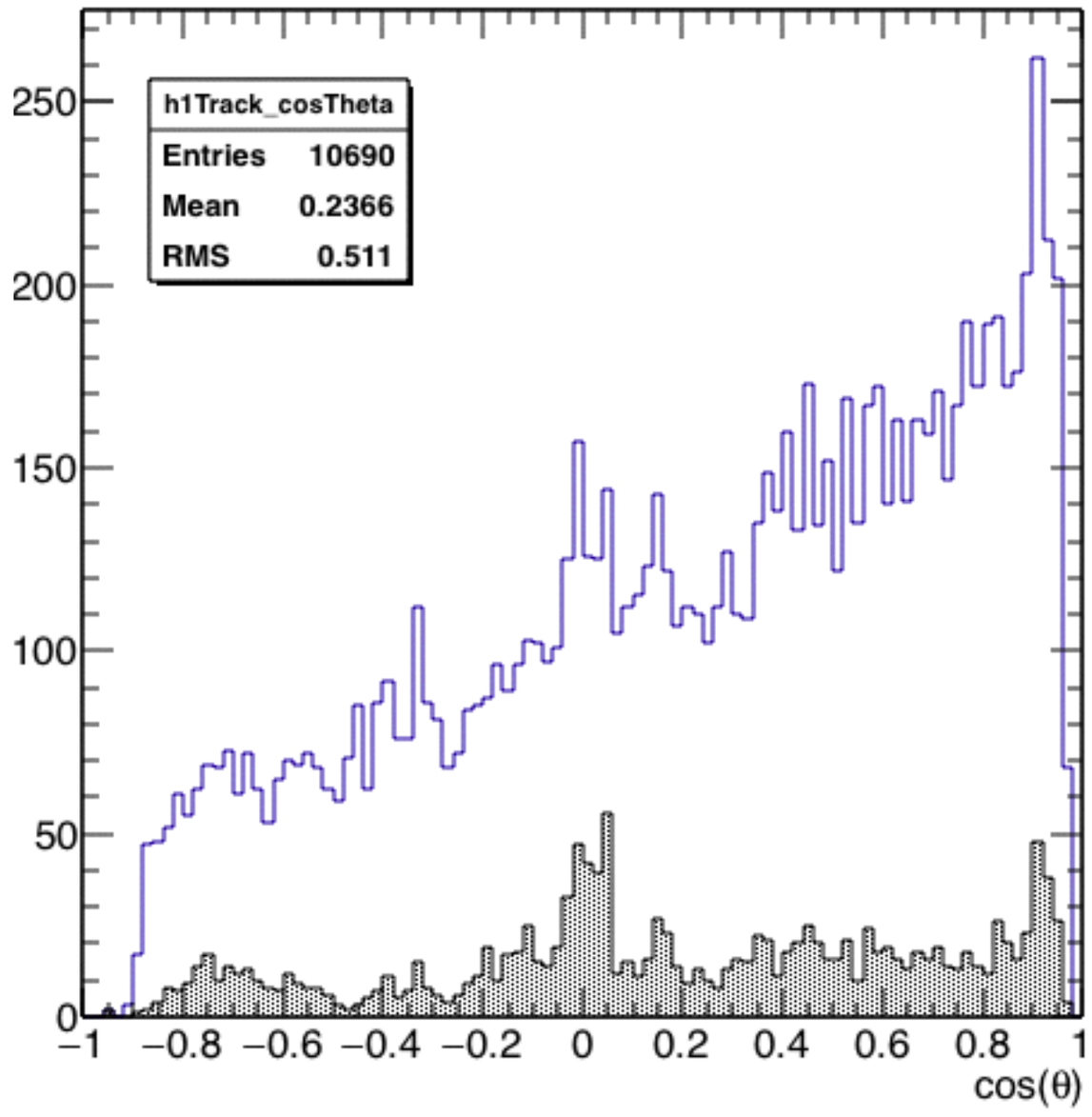
- ➡ No-Intercept Tracks have a maximum transverse momentum of 300 MeV/c
- ➡ Most of them (~800 over 1454) have a poor p-value

Direction of the No-Intercept Tracks

azimuthal angle distribution



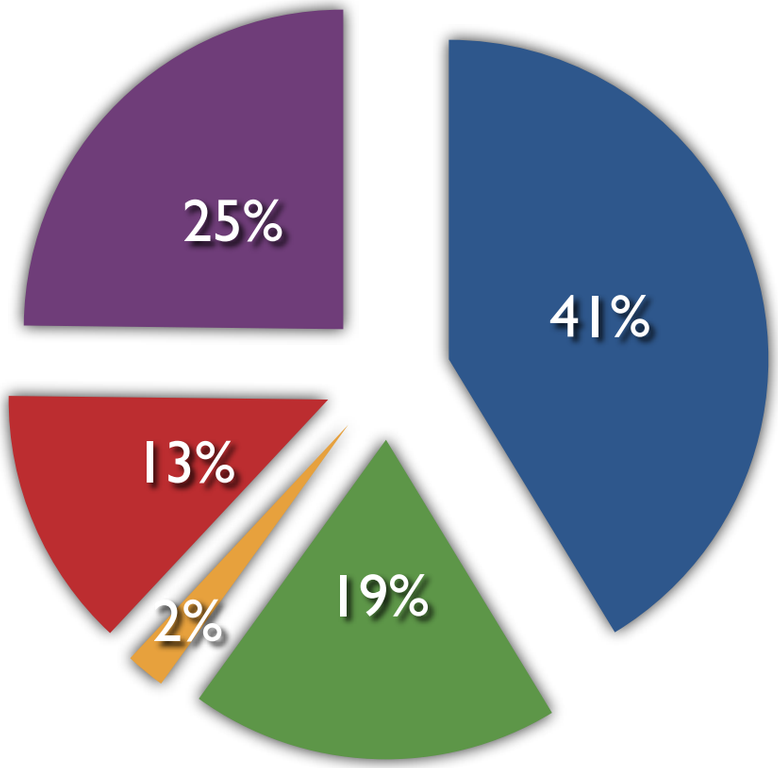
polar angle distribution



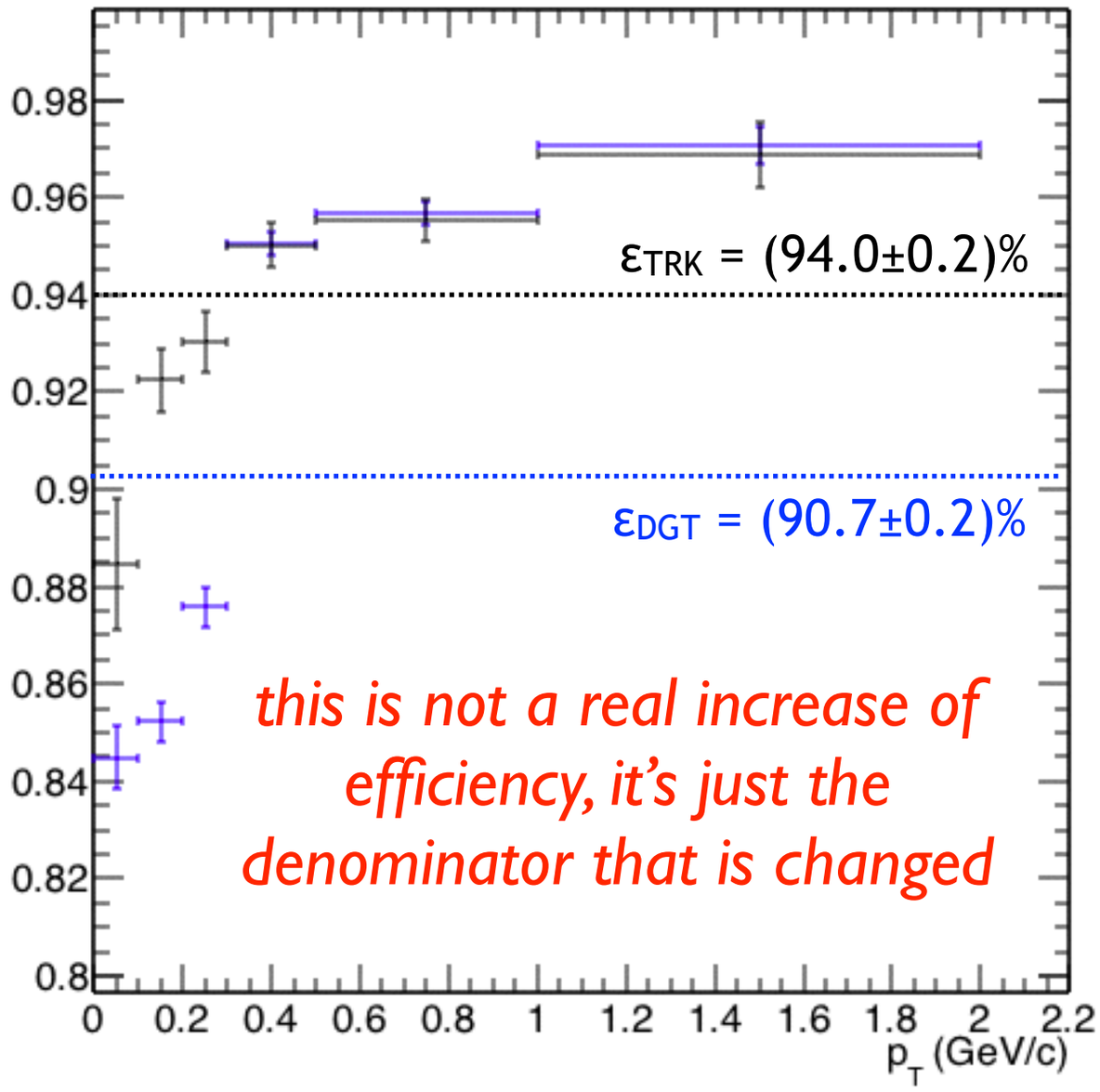
- ➡ No dependence on the azimuthal angle is present
- ➡ In the polar direction the very forward direction and the 90 degree one seem more populated

SVD-Tracking

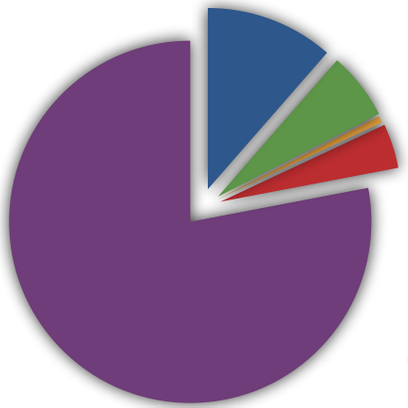
- In 1k events, 9262 tracks have at least one related PXD Digit, 8710 tracks have at least one PXD Digit contained in an ROI.
- What about the other 552 tracks?



- ROI with correct VxdID (228)
- ROI with wrong VxdID (103)
- no ROI, Intercept with correct VxdID (11)
- no ROI, Intercept with wrong VxdID (73)
- no Intercept (137)

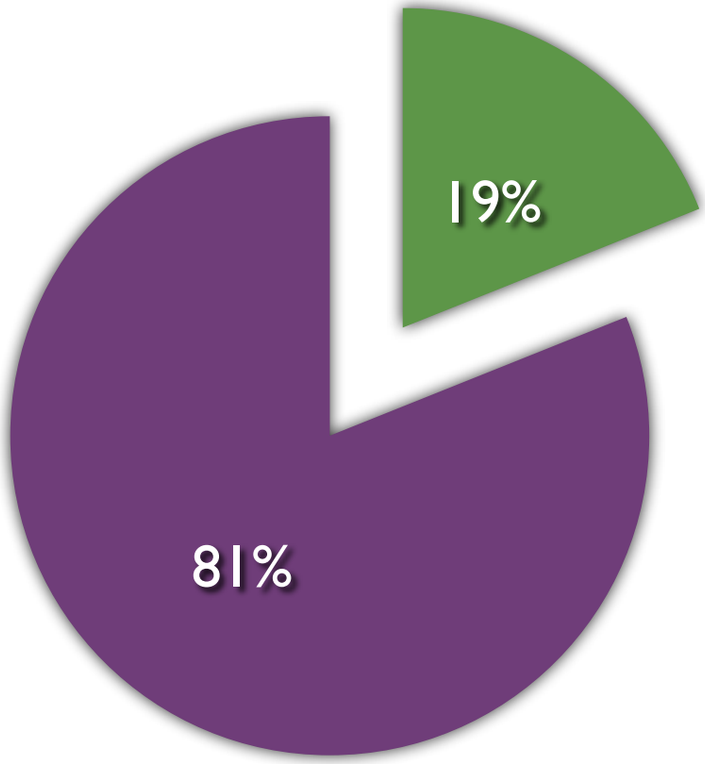


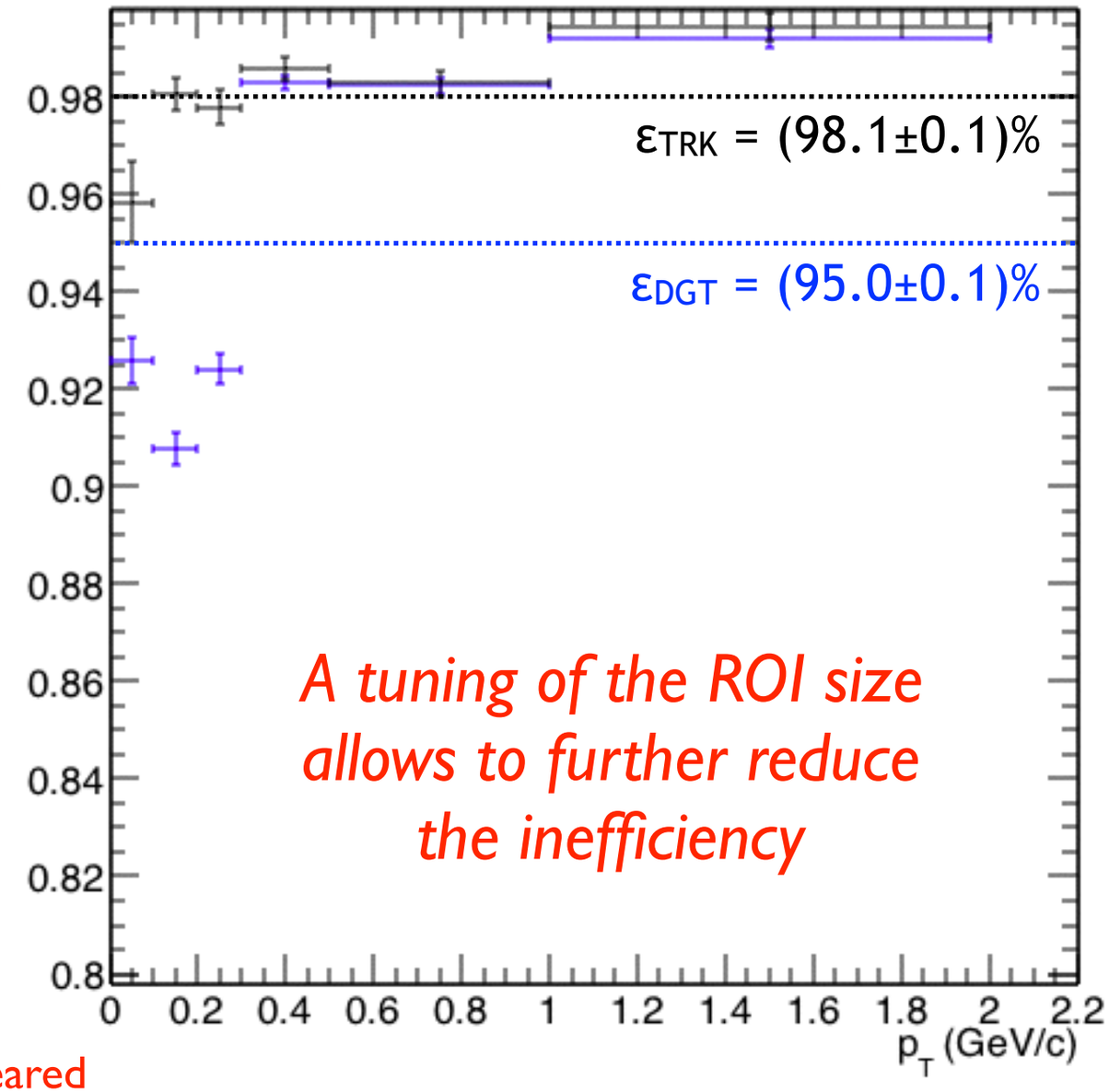
SVD+CDC scenario



SVD-Tracking & Full Frame ROIs

- Among 1k events, 9262 tracks have at least one related PXD Digit, 9093 tracks have at least one PXD Digit contained in an ROI.
- What about the other 169 tracks?





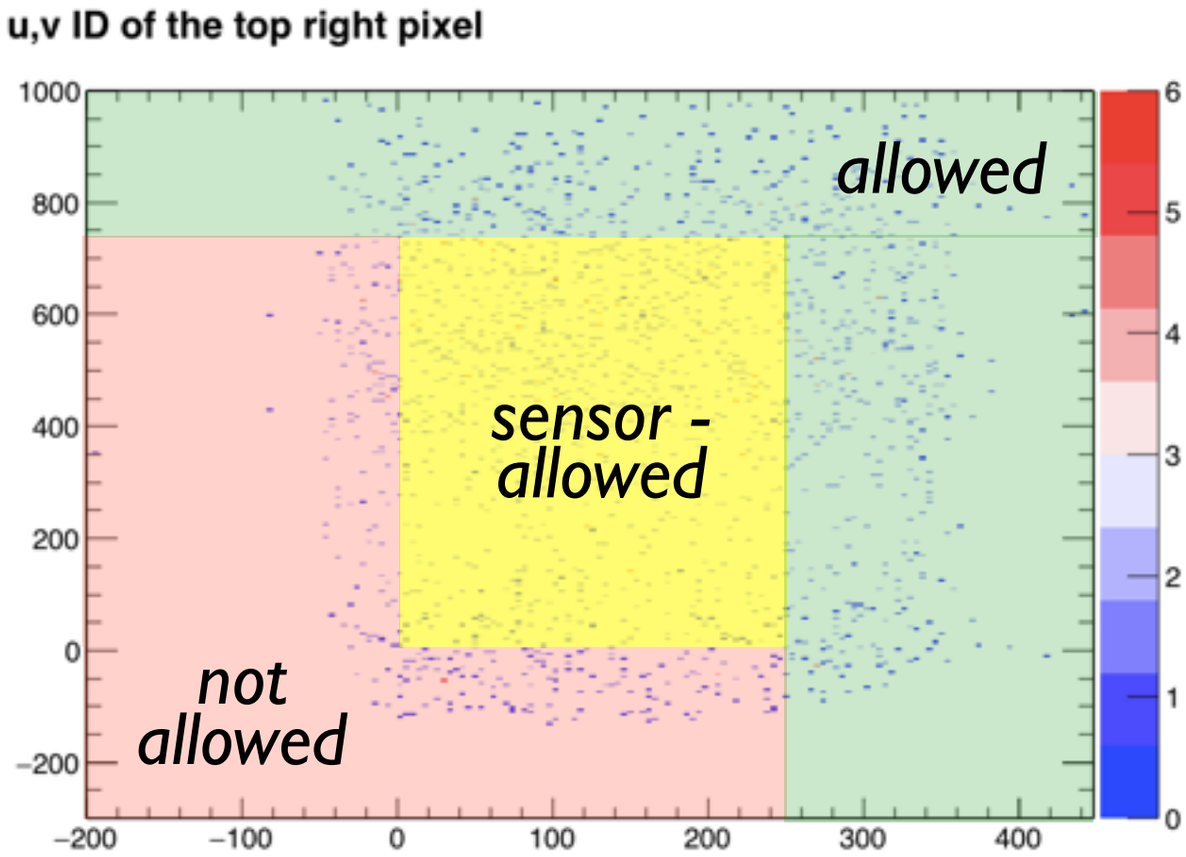
A tuning of the ROI size allows to further reduce the inefficiency

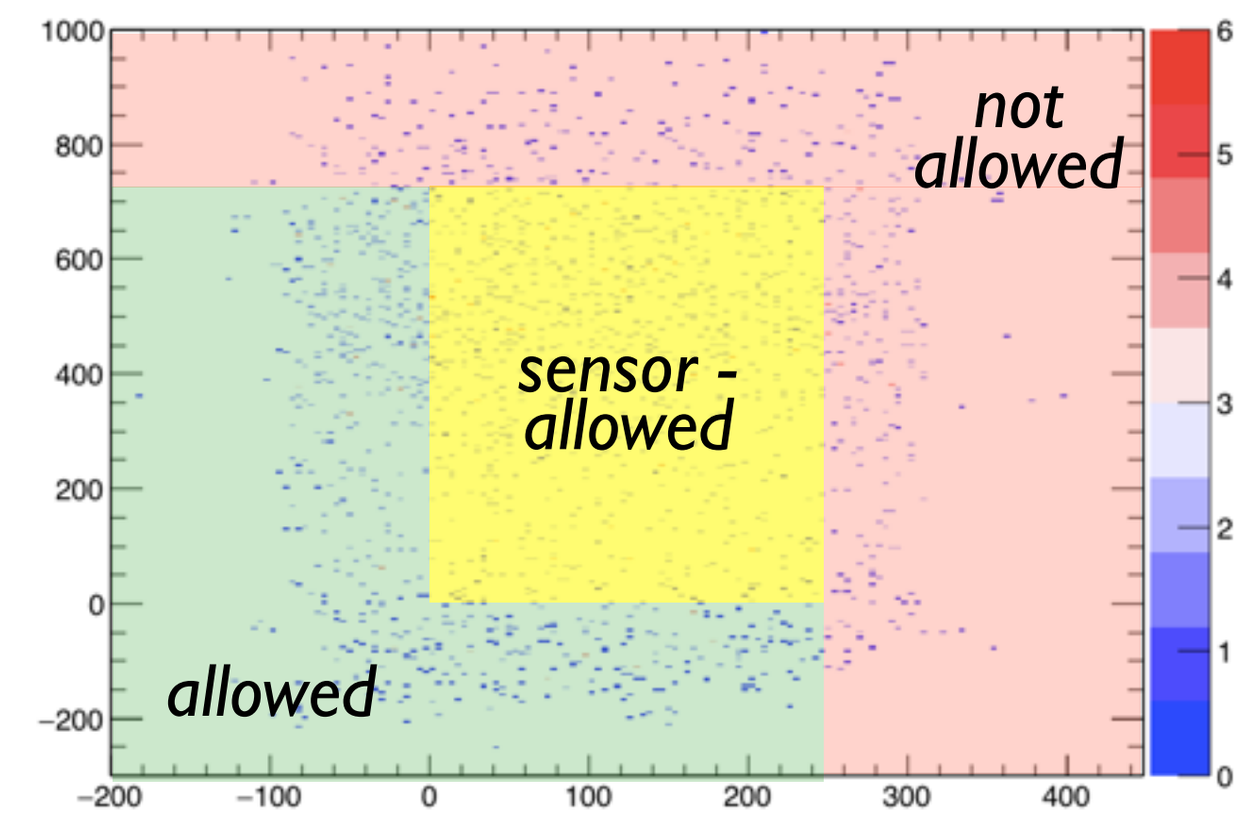
- ROI with correct VxdID (0) ← disappeared
- ROI with wrong VxdID (32) ← reduced to 1/3
- no ROI, Intercept with correct VxdID (0) ← disappeared
- no ROI, Intercept with wrong VxdID (0) ← disappeared
- no Intercept (137) ← same

do we understand the migration?

ROI Corners Definition (`ROIPixelTranslator`)

- intercepts can be outside the sensor area, as well as the ROI corners
- translate the bottom-left and the top-right corners positions in pixels cell IDs of the sensor (`ROIPixelTranslator`)
- only if the ROI *overlaps* with the sensor, then it is appended to the final list



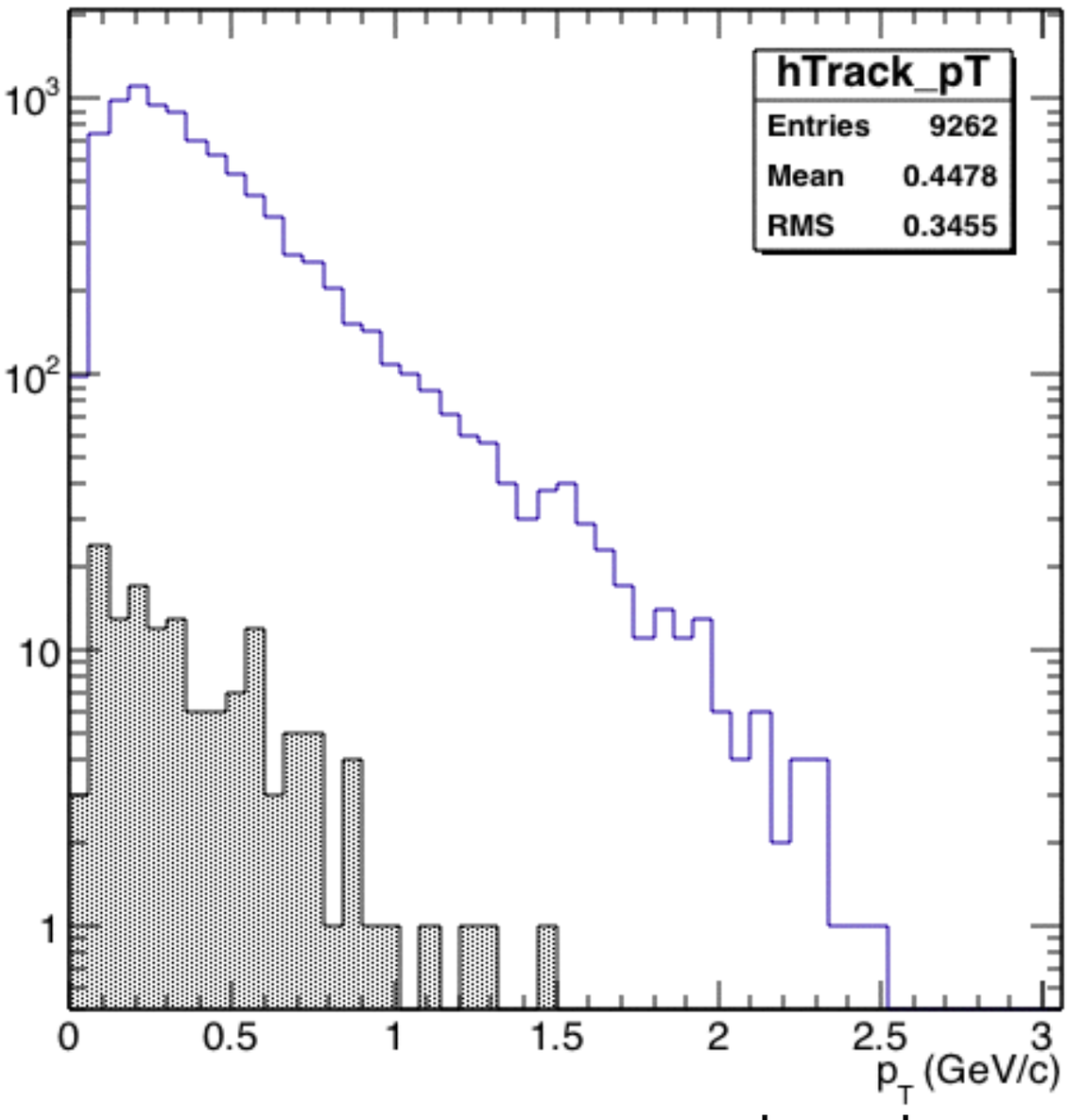


✓ Increasing the widths of the ROIs:

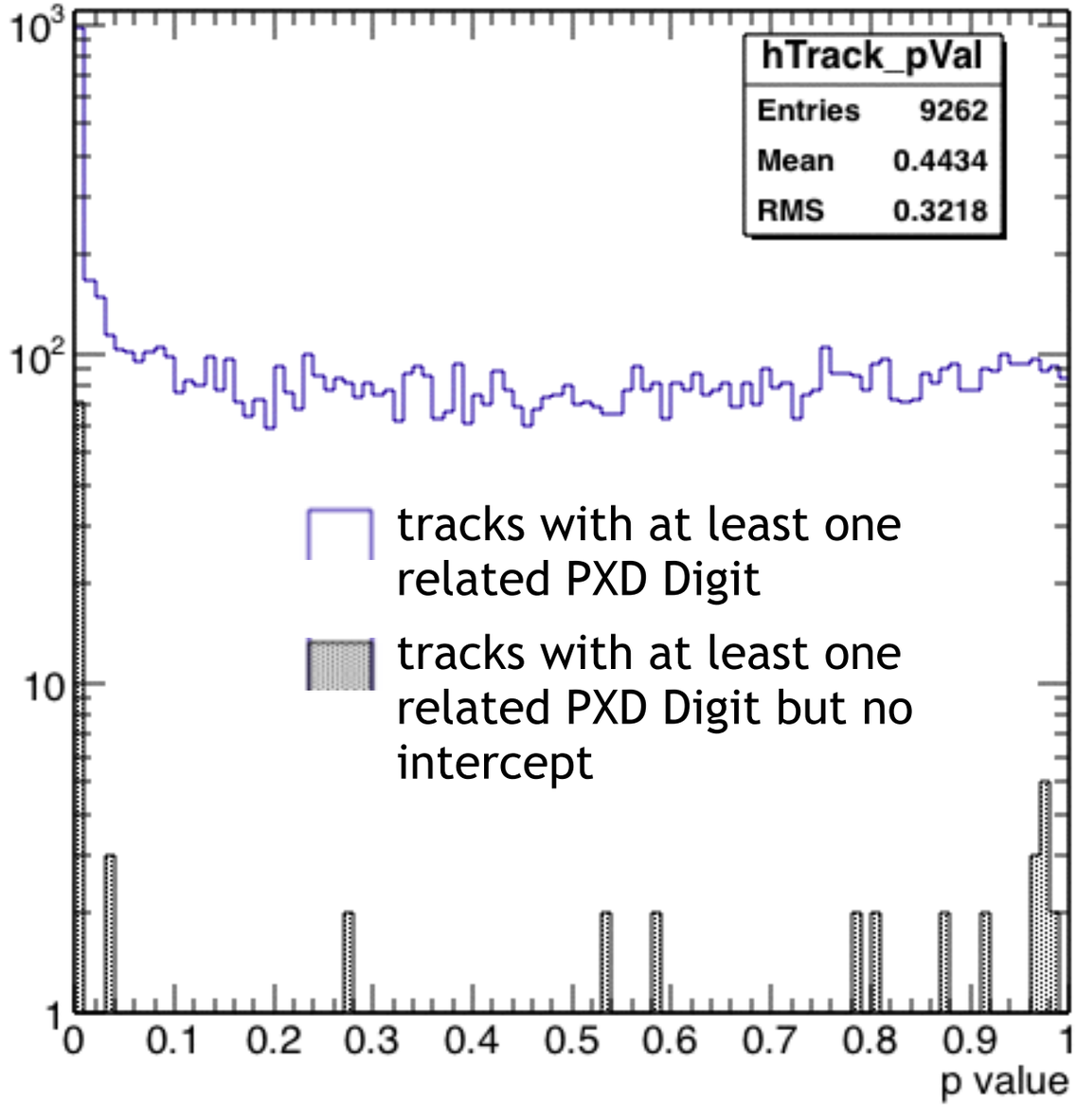
- ROI with correct VxdID: now contain the PXD Digit
- ROI with wrong VxdID: intercepts with correct VxdID existed but ROIs did not overlap with the sensor
- no ROI with correct VxdID: intercepts with correct VxdID existed but ROIs did not overlap with the sensor
- no ROI with wrong VxdID: same as above

Full Frame ROIs, SVD-only tracks (1)

transverse momentum distribution



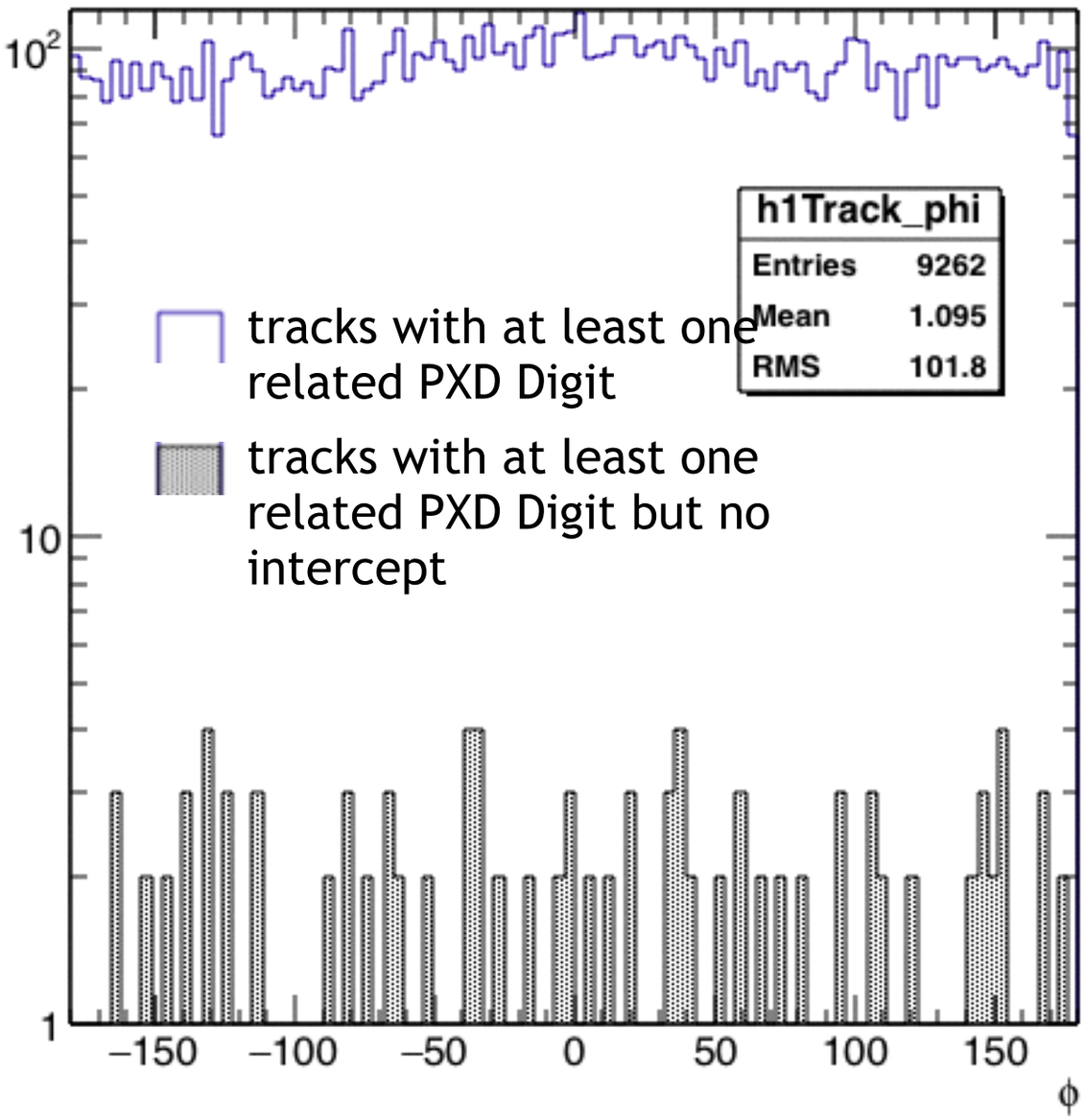
p-value distribution



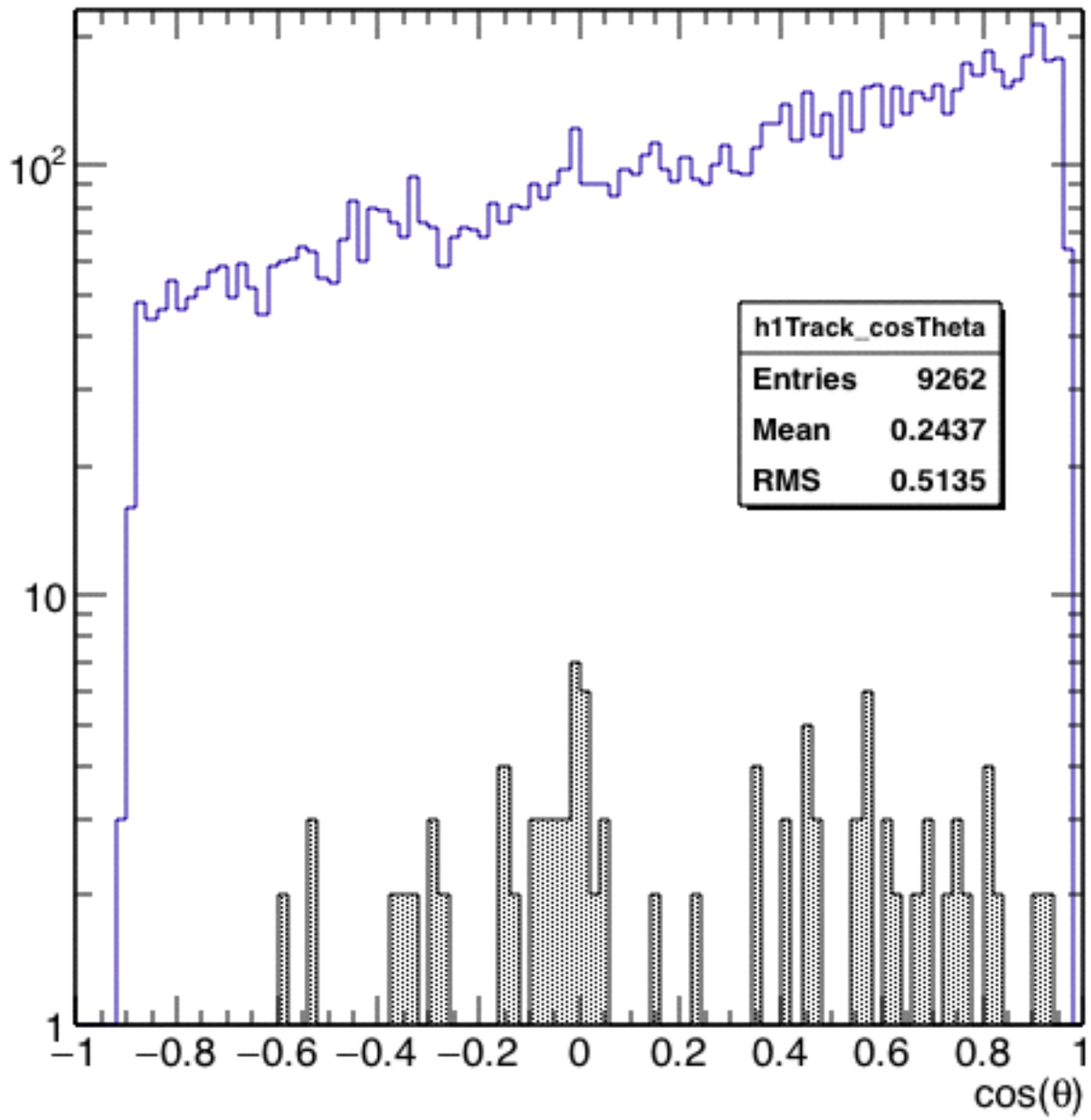
- ➡ no tranverse momentum dependence
- ➡ most of the No-Intercept tracks (~70 over 169) have a poor p-value

Full Frame ROIs, SVD-only tracks (2)

azimuthal angle distribution



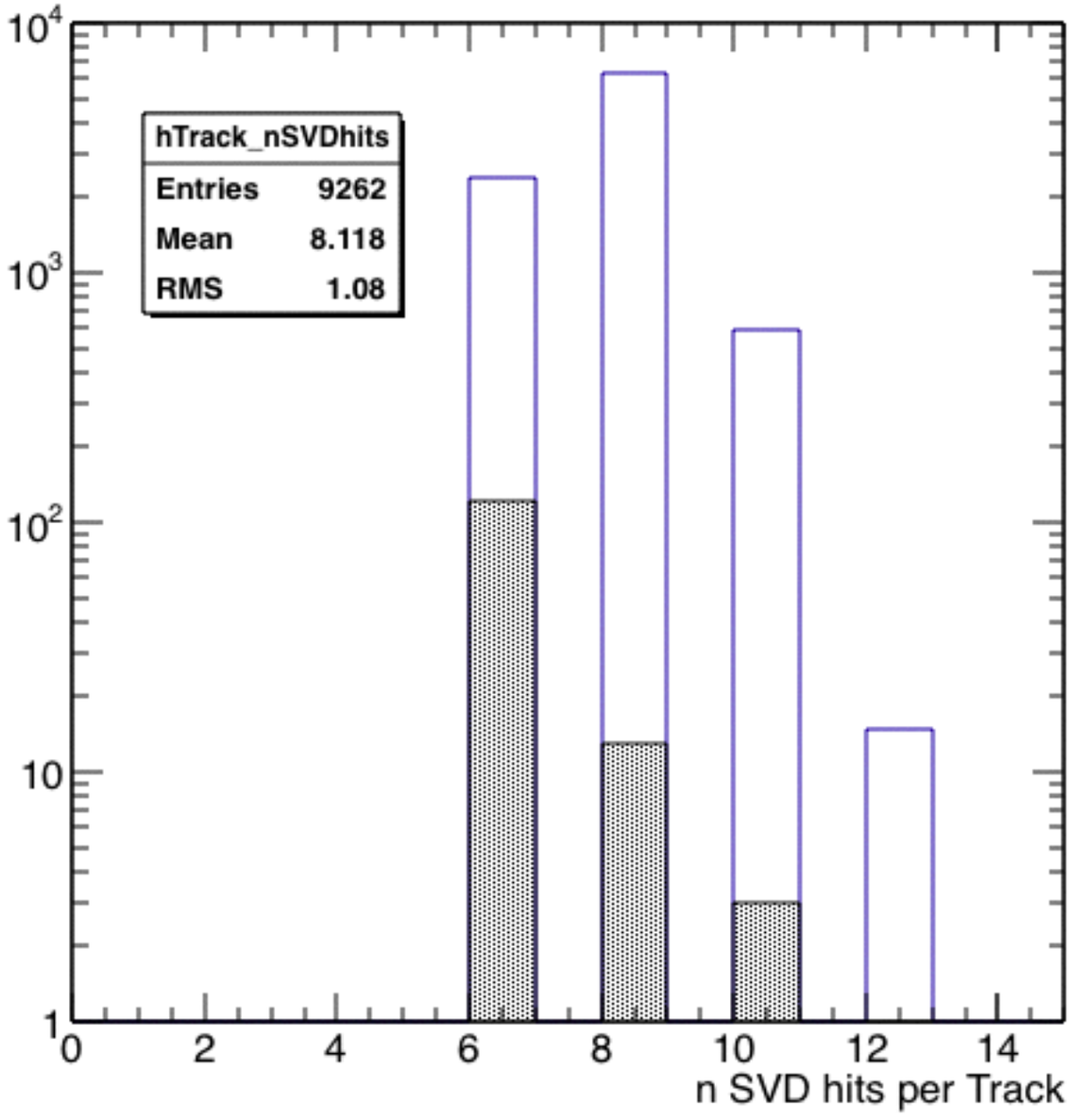
polar angle distribution



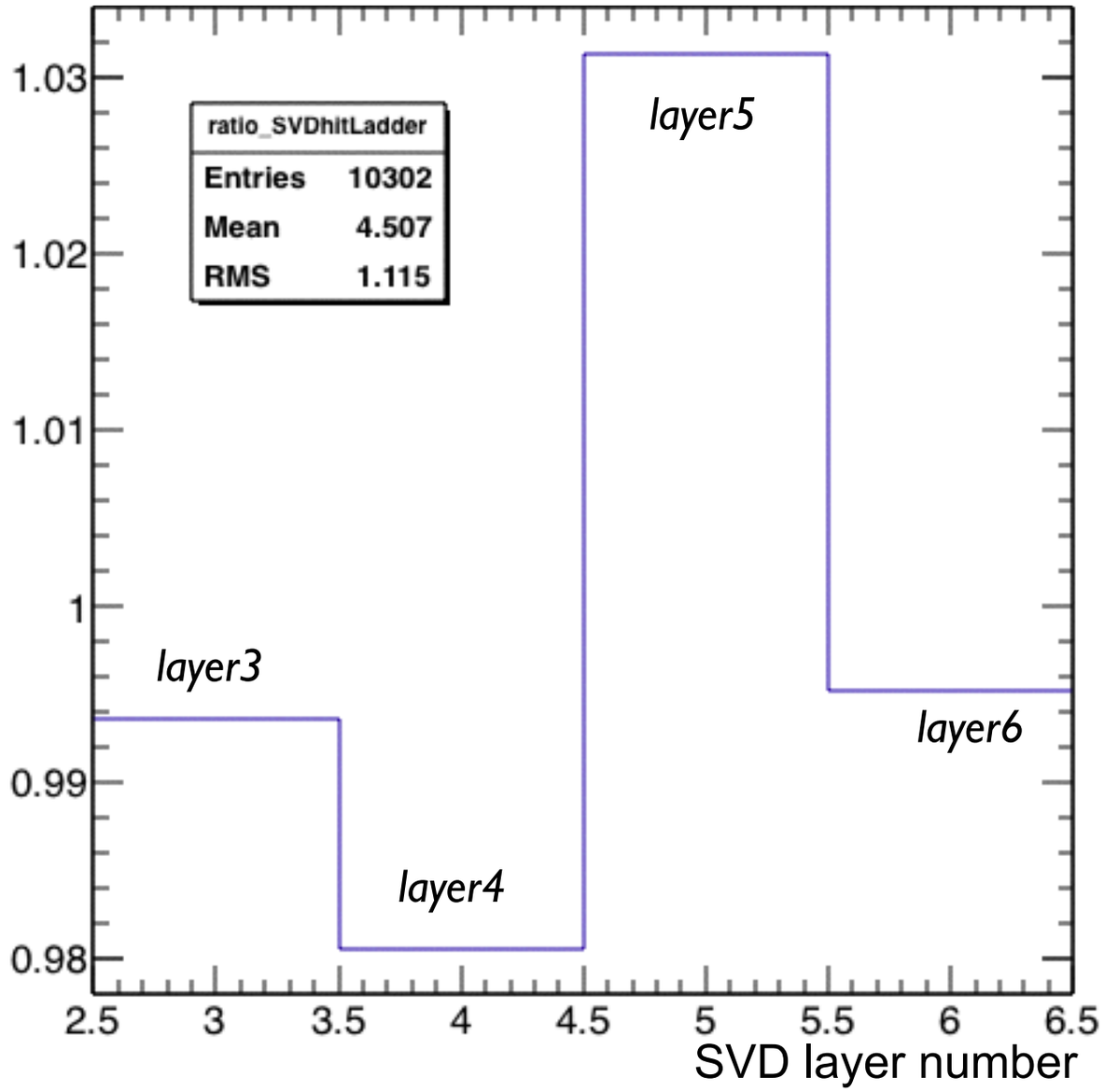
- ➡ No dependence on the azimuthal angle is present
- ➡ A hint of clusterization at 90 degree in the polar angle can be seen

Full Frame ROIs, SVD-only tracks (3)

number of SVD hits per Track



$$\frac{\text{number of hits of Track}}{\text{number of hits of a No-Intercept Track}}$$



- ➡ No-Intercept SVD-only tracks have more than 6 hits in the SVD
- ➡ no hints that the track-finding problem is related to the hit SVD layer, only a few % differences

Conclusions

- Much better understanding of the ROI-finding performances
  - 80% of the inefficiency is due to missing intercepts
  - improved SVD track-finding will directly reflect in an improved ROI-finding efficiency
  - the size of the ROIs must be better tuned

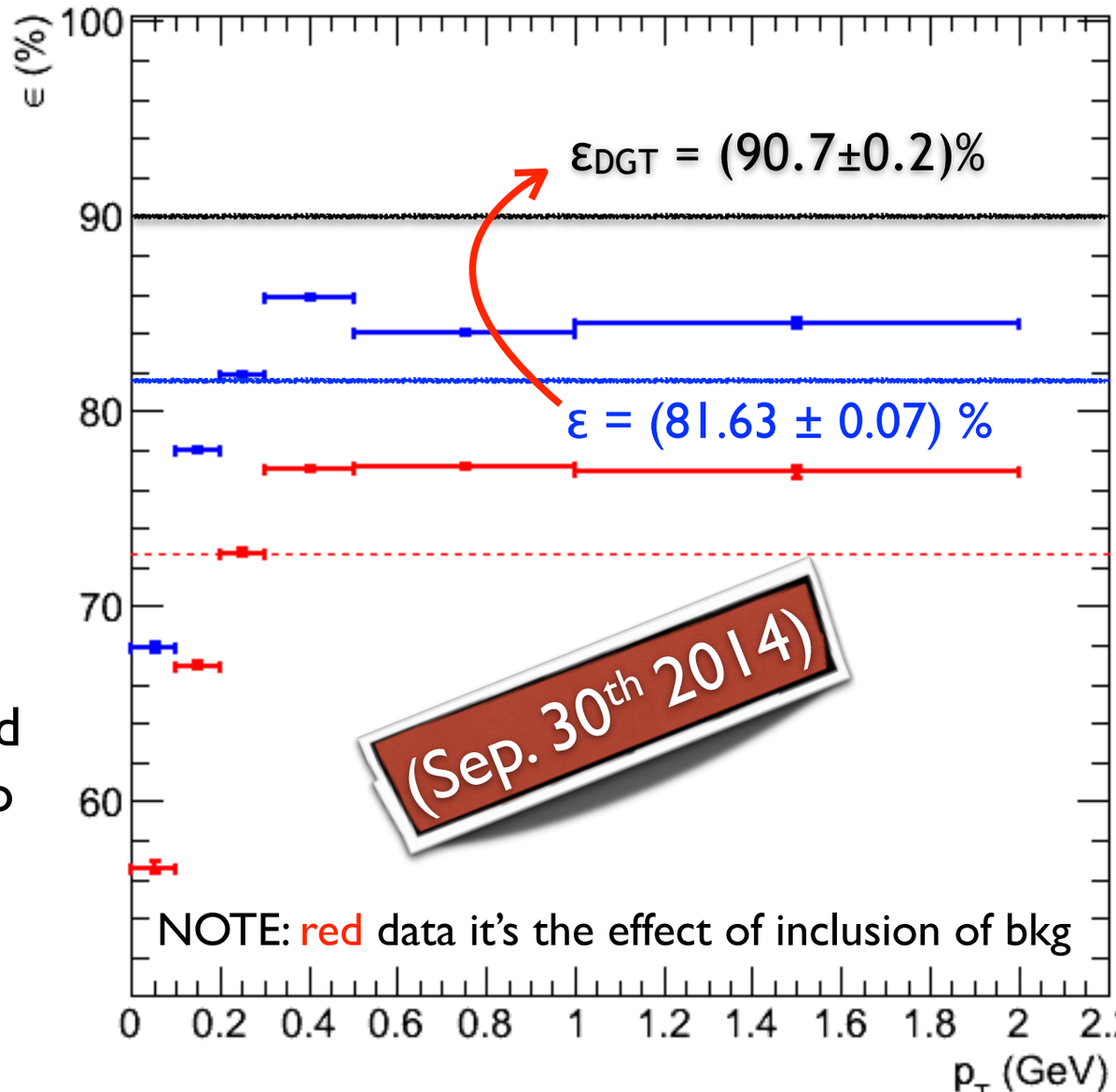
- A better analysis-suited definition of efficiency

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

- plans for the future: go back to slide 2

- With SVD-only track finding we have an increased (old-definition) efficiency of ~10% with respect to … some time ago!

comparing apples and apples:



(Sep. 30th 2014)

ROI Finding Efficiency

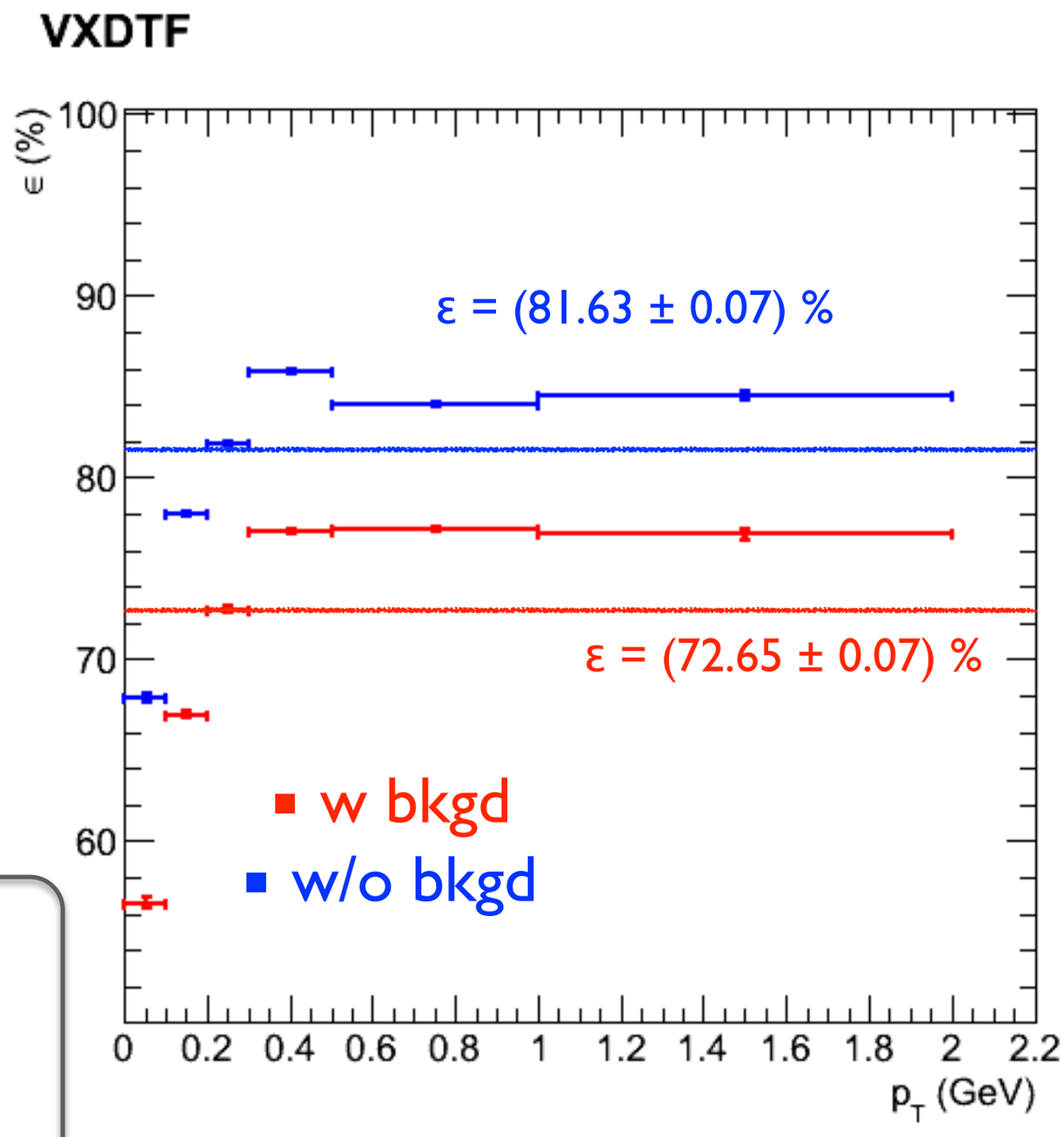
ROI finding efficiency evaluated on 10k generic B decays:

- ➡ Efficiency w/o bkg = (81.63±0.07)%
  - increased by ~6.5% w.r.t. previous studies (9th june 2104)
- ➡ Efficiency w bkg = (72.65±0.07)%
- ➡ In both cases inefficiency mostly due to failures in fitting the track and finding an intercept with the sensor planes (as always)

$$\varepsilon = \frac{\#\ \text{PXDDigits inside a ROI}}{\text{total \# PXDDigits of TrackCand}}$$

inefficiencies of the pattern recognition are factorized,

but the TrackCand quality is not!



ROIPixelTranslator::fillRoiIDList( )

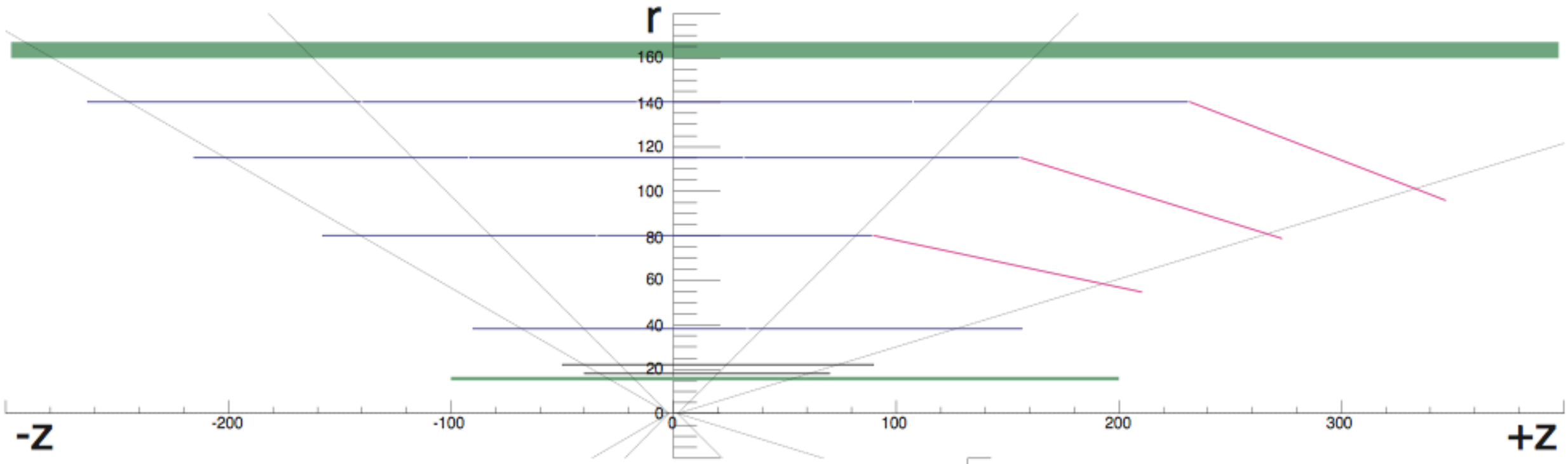
```
for (int i = 0; i < listOfIntercepts->getEntries(); i++) {

   const VXD::SensorInfoBase& aSensorInfo = aGeometry.getSensorInfo((*listOfIntercepts)[i]->getSensorID());

   double widthTotU = std::min(m_maxWidthU , sqrt((*listOfIntercepts)[i]->getSigmaU() *
                      (*listOfIntercepts)[i]->getSigmaU() + m_sigmaSystU * m_sigmaSystU) * m_numSigmaTotU);
   double minU = (*listOfIntercepts)[i]->getCoorU() - widthTotU / 2 ;
   double maxU = (*listOfIntercepts)[i]->getCoorU() + widthTotU / 2 ;
   const int nPixelsU = aSensorInfo.getUCells() - 1;

  [same for v]

   const int firstPixelID = 0;

   double bottomLeft_uID = aSensorInfo.getUCellID(minU, minV, false);
   double bottomLeft_vID = aSensorInfo.getVCellID(minV, false);
   double topRight_uID = aSensorInfo.getUCellID(maxU, maxV, false);
   double topRight_vID = aSensorInfo.getVCellID(maxV, false);

   bool inside = true;
     if (bottomLeft_uID > nPixelsU || topRight_uID < firstPixelID || bottomLeft_vID > nPixelsV || topRight_vID <
firstPixelID) {
     {  B2DEBUG(1, "  OOOPS: this pixel does NOT belong to the sensor");
        inside = false; }

   ROIid tmpROIid;

   if (inside) {
     …
     ROIidList->appendNew(tmpROIid);
     ROIid* transientROIid = (*ROIidList)[ROIidList->getEntries() - 1];

     (*listOfIntercepts)[i]->addRelationTo(transientROIid);
   }
}
```

PXDDataRedAnalysis::event()

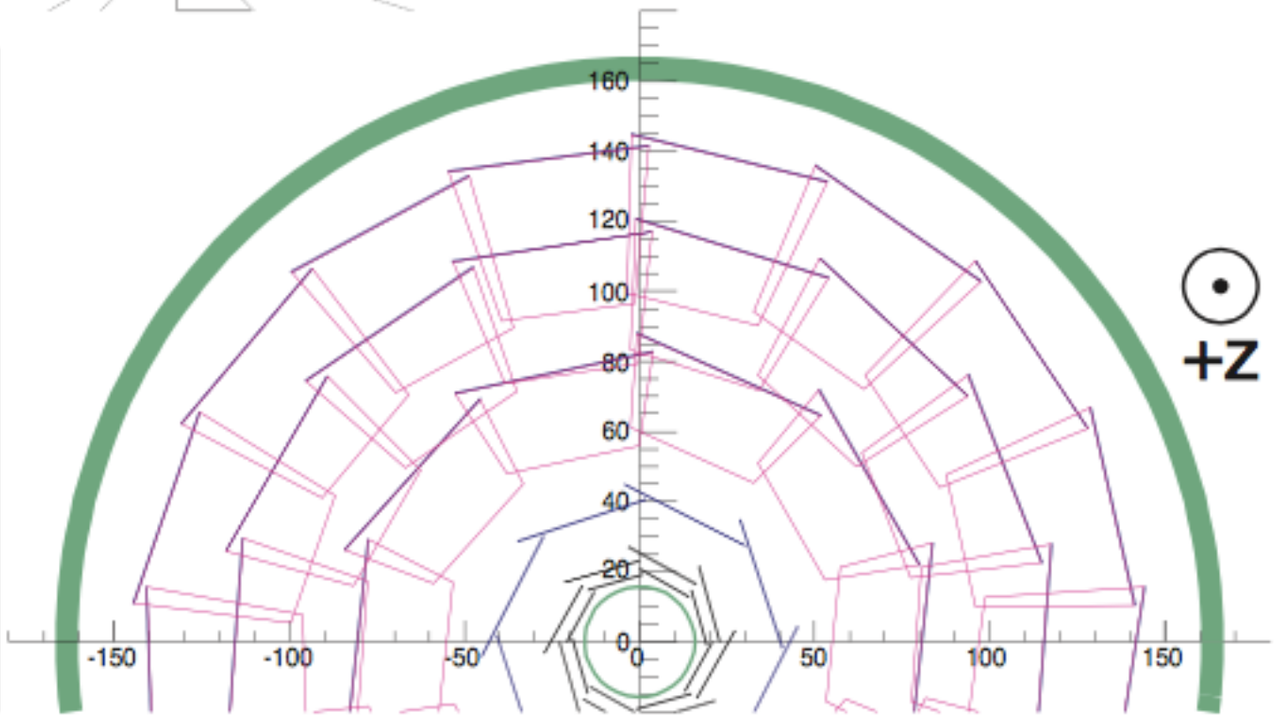
```
for (int i = 0; i < (int)trackList.getEntries(); i++) {

    RelationVector<MCParticle> MCParticles_fromTrack = DataStore::getRelationsFromObj<MCParticle>(trackList[i]);

    if (MCParticles_fromTrack.size() == 0)
      continue;

    for (int j = 0; j < (int)MCParticles_fromTrack.size(); j++) {

      MCParticle* aMcParticle = MCParticles_fromTrack[j];

      RelationVector<PXDDigit> pxdRelations = aMcParticle->getRelationsFrom<PXDDigit>();
      RelationVector<SVDCluster> svdRelations = aMcParticle->getRelationsFrom<SVDCluster>();
```

```
     PXDInterceptsFromRecoTracks  PXDIntercepts = recoTrackToPXDIntercept.getElementsFrom(trackList[i]);

     if (pxdRelations.size() == 0)
       continue;

 for (unsigned int iPXDDigit = 0; iPXDDigit < pxdRelations.size(); iPXDDigit++) {

    for (; thePXDInterceptIterator != thePXDInterceptIteratorEnd; thePXDInterceptIterator++) {

        const PXDIntercept* theIntercept = thePXDInterceptIterator->to;

        const ROIid* theROIid = theIntercept->getRelatedTo<ROIid>(m_ROIListName);
```

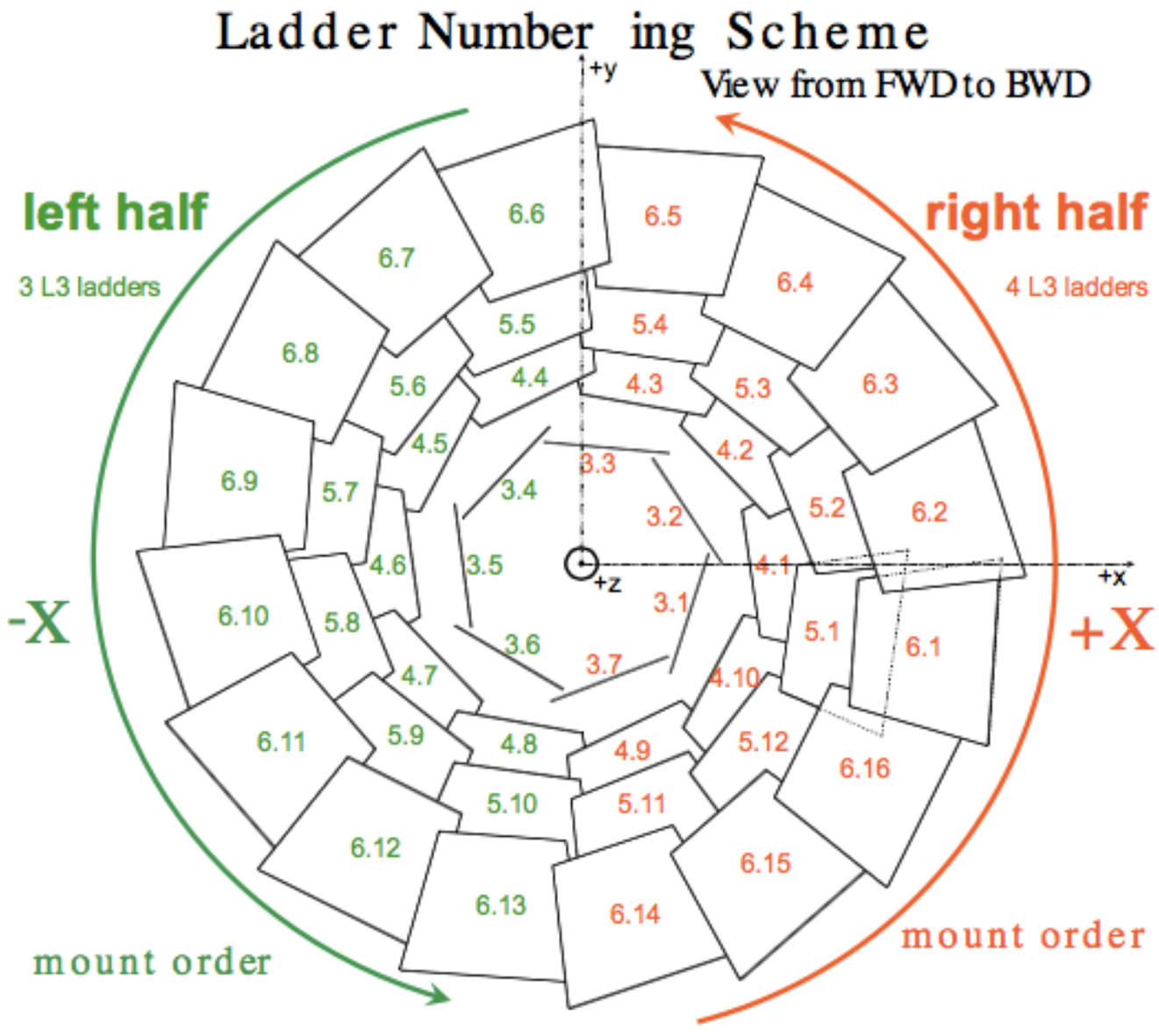
SVD Layout




```
conf100210.dat
W -300 400 -20 180 1
B       0.0 0.0 15.0 16.5 -100.0 200.0  0 0 0
O 0.0 0.0 160.0 167.0 -400 800    0 0 0
C        17   150 0.0   2.0      200.0
C        45   135 0.0   2.0      200.0
X 0.3  110.0   16.0       18.0   -40.0    3   1   0.35   -2.0    8
X 0.3  140.0   22.0       22.0   -50.0    3   1   0.35   -5.0    8
R 0.3  122.8   38.4       38.0   -90.0    1   2   0.4    -5.0    8
R 0.3  122.8   57.6       80.0   -157.6   1   2   0.83   -8.0    10
R 0.3  122.8   57.6      115.0   -215.4   1   3   0.83   -8.0    14
R 0.3  122.8   57.6      140.0   -263.2   1   4   0.6    -8.0    17
S 0.3  122.8   57.6  38.4  140.0   232.0  -21.1     0.6   -8.0   17
S 0.3  122.8   57.6  38.4  115.0   156.0  -17.2     0.83  -8.0   1
S 0.3  122.8   57.6  38.4   80.0    90.0  -11.9     0.83  -8.0   10
*** Total Sensor area=1.2366 / m2 ***
```



SVD Layout


Ladder Numbering Scheme
View from FWD to BWD
+y
left half
3 L3 ladders
right half
4 L3 ladders
6.6
6.5
6.7
6.4
5.5
5.4
6.8
4.4
4.3
5.3
6.3
5.6
4.5
4.2
6.9
5.7
3.3
3.4
3.2
5.2
6.2
4.6
3.5
4.1
+z
+x
3.1
-X
+X
6.10
5.8
5.1
6.1
3.6
3.7
4.7
4.10
6.11
5.9
4.8
4.9
5.12
6.16
5.10
5.11
6.12
6.15
6.13
6.14
mount order
mount order