PXD Data Reduction on the HLT

Giulia Casarosa & Eugenio Paoloni

INFN - Sezione di Pisa

F2F Tracking Meeting

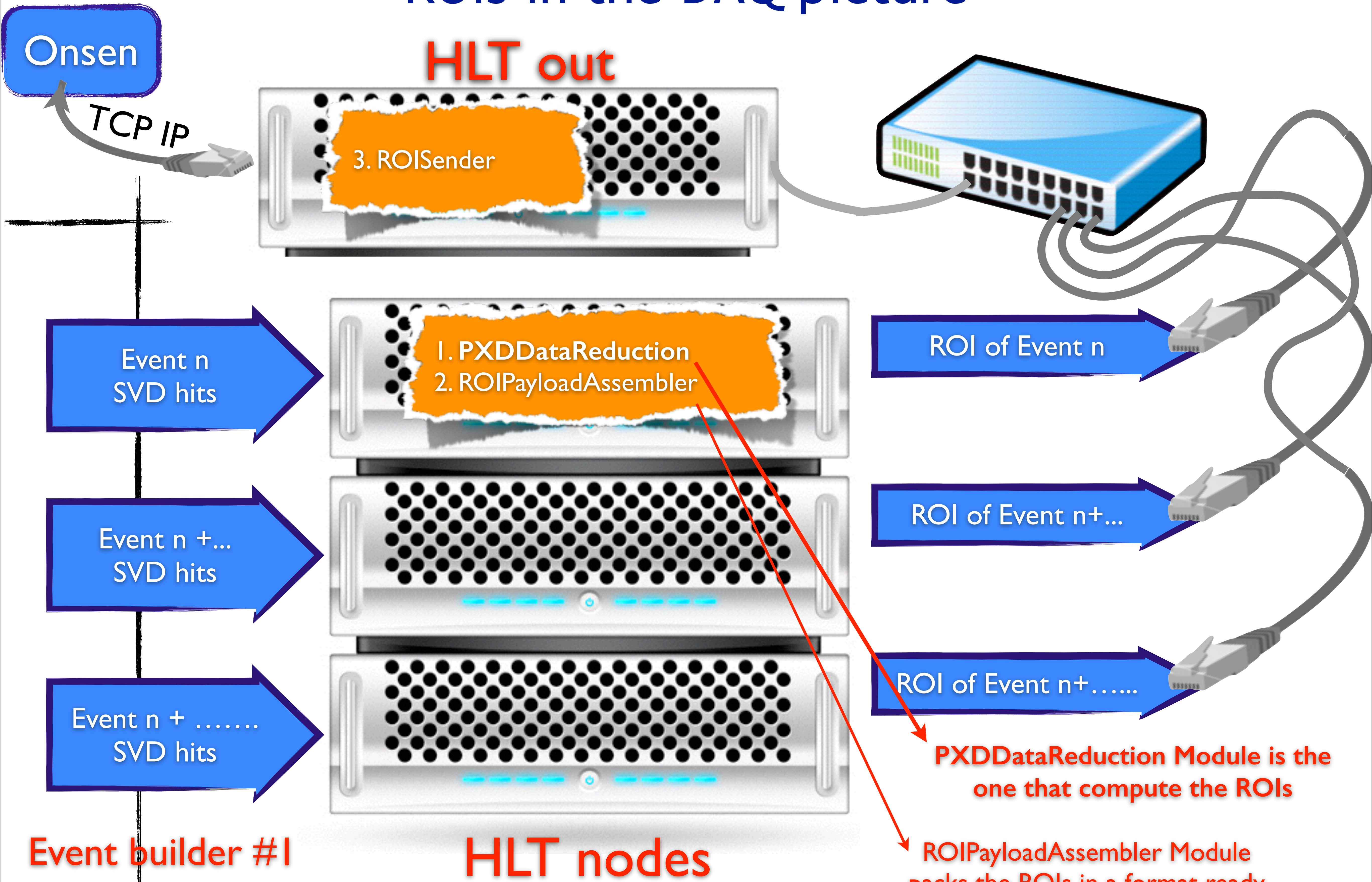
Prague - 12th December 2013

Outline

- PXD Data Reduction basf2-modules description
- What’s happening after the changes to the software (all at once):
  - genfit2
  - new externals v00-05-00 (python 2.7)
  - [TB geometry (not freezed yet)]
- The first (very) preliminary results after the migration to genfit2
- short- & long-term list of to-dos
- Conclusions

basf2 Modules

ROIs in the DAQ picture



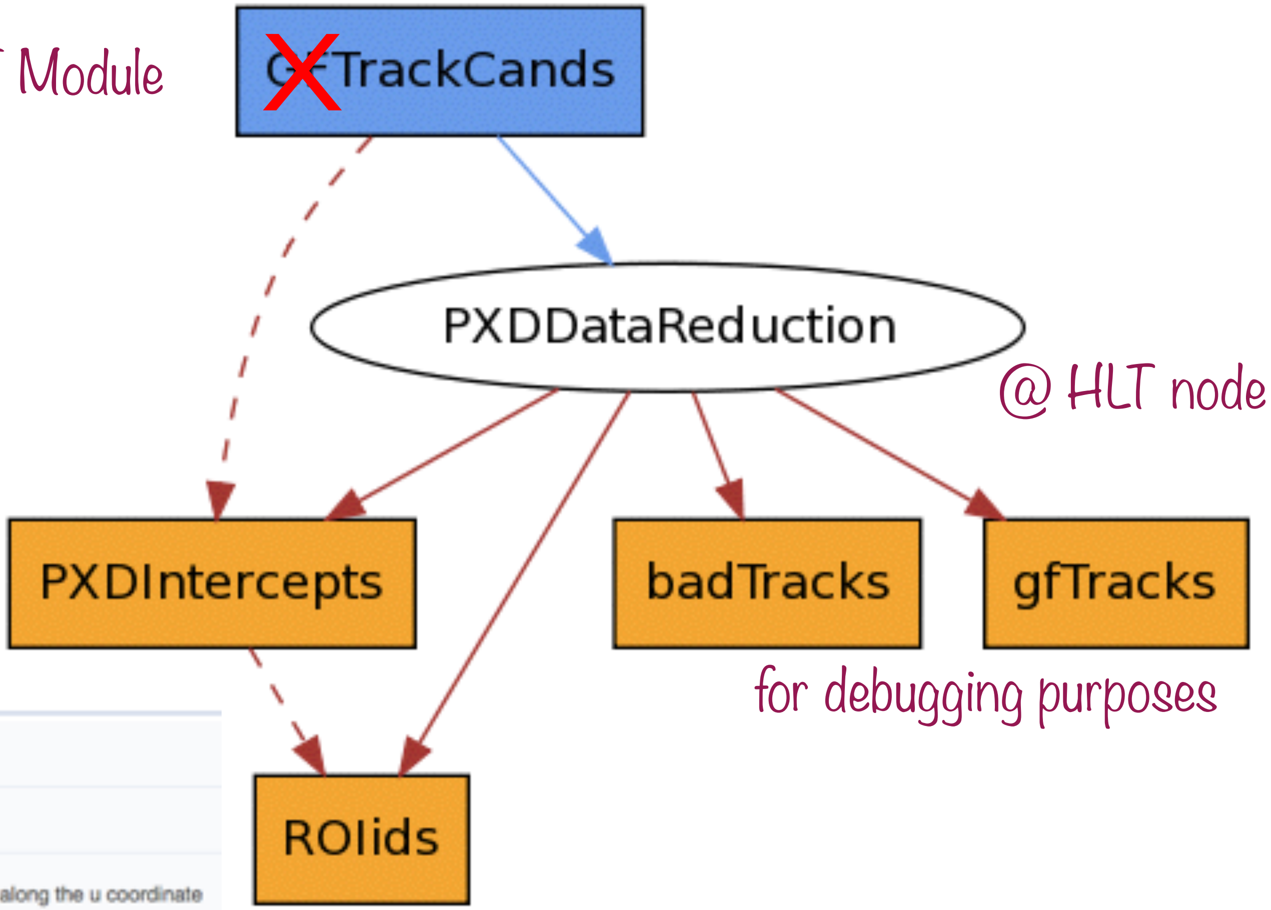
PXDDataReduction Module is the one that compute the ROIs

ROIPayloadAssembler Module packs the ROIs in a format ready for the ONSEN

HLT nodes
running basf2 executables

Main Modules Descriptions

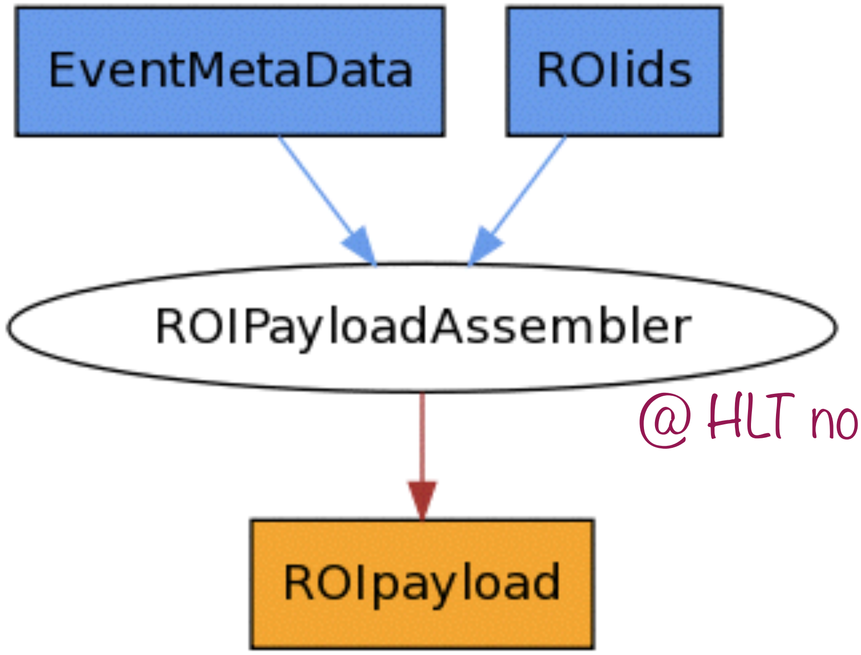
from VXDTF Module



@ HLT node

for debugging purposes

evt number, run number,
exp number



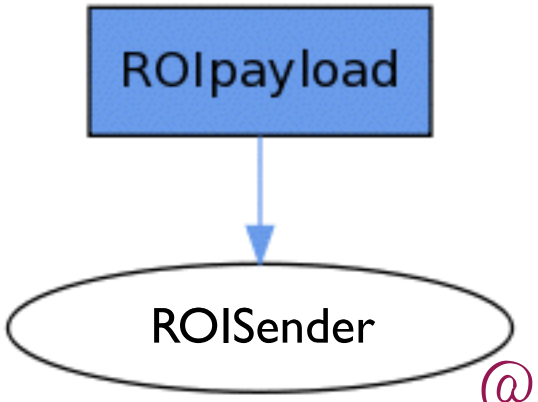
@ HLT node

Private Attributes

double	m_coorU	u coordinate of the intercept
double	m_coorV	v coordinate of the intercept
double	m_sigmaU	statistical error of the extrapolation along the u coordinate
double	m_sigmaV	statistical error of the extrapolation along the v coordinate
double	m_sigmaUprime	statistical error of the extrapolation of U prime
double	m_sigmaVprime	statistical error of the extrapolation of V prime
double	m_lambda	length of the track
VxdID::baseType	m_sensorID	sensor ID

Private Attributes

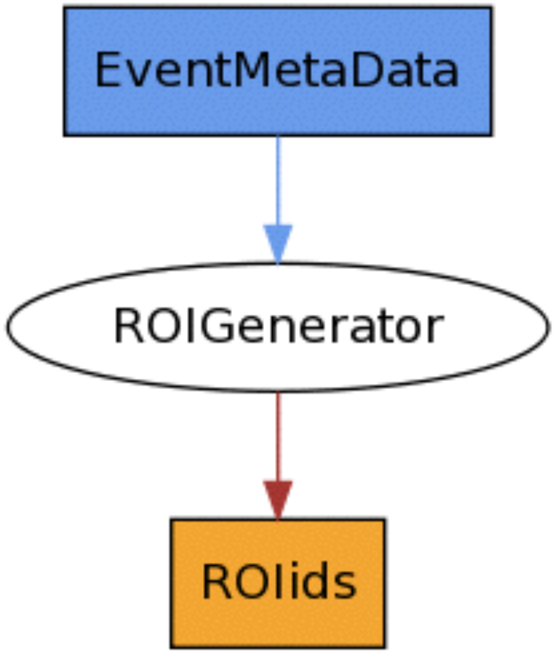
int	m_minUid	u ID of the bottom left pixel of the ROI
int	m_maxUid	u ID of the top right pixel of the ROI
int	m_minVid	v ID of the bottom left pixel of the ROI
int	m_maxVid	v ID of the top right pixel of the ROI
VxdID::baseType	m_sensorID	sensor ID



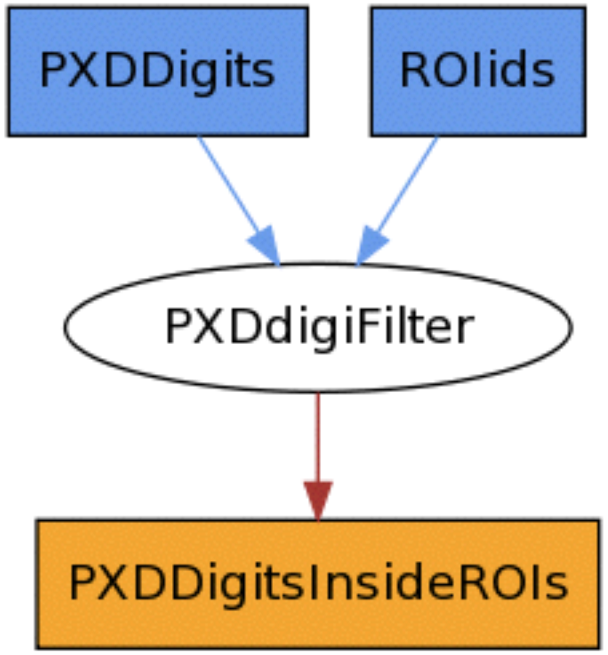
@ HLT out

sends the payload
via TCP-IP to the onsen

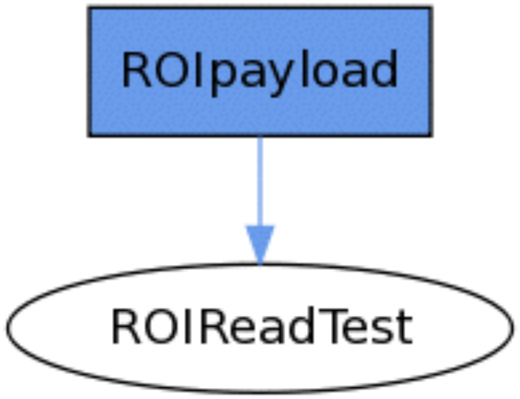
Ancillary Modules



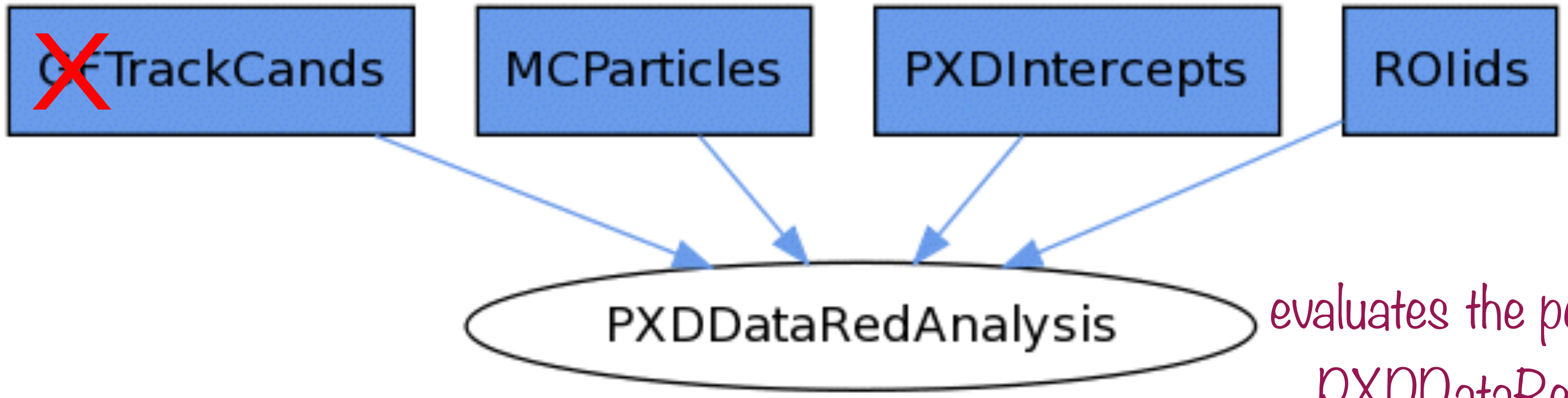
generates ROIs



removes the PXDDigit outside the ROIs
(not really an ancillary module,
we need it for simulation)



to test the payload format



evaluates the performance of the
PXDDataReduction Module
(efficiency...)

Additional Informations

- ➡ At present, all modules and classes are in the *tracking* package
- ➡ The PXDdigiFilter will be moved to the *pxd* package
- ➡ The ROIid dataobject will be renamed to ROI and will be moved to the *pxd* package
- ➡ The PXDDataRedAnalysis will be re-designed in order to have an easier and more efficient tool of analysis (after the test beam)
- ➡ Apply rules on naming scheme / coding convention discussed yesterday

What Happened after the Last Changes the Software

r7793 and externals v00-05-00 (1)

- first solved compilations problems due to `trg/SConscript` and `trg/cdc/include/TRGCDC.h`
- migration to genfit2 done by Johannes:

```
[KEK@~/releases/r7793]$ svn log tracking/modules/pxdDataReduction/
------------------------------------------------------------------------
r7708 | Jojo1987 | 2013-12-05 03:00:04 +0900 (Thu, 05 Dec 2013) | 1 line

genfit2 merge
------------------------------------------------------------------------
```

- first attempt to run the module:

```
[KEK@~/releases/r7793]$ basf2 tracking/examples/pxdDataReduction.py
```

[…]

```
Terminate current event processing.
[DEBUG] %%%%%%%% EVENT # of tracks =  12  { module: PXDDataReduction @tracking/modules/pxdDataReduction/src/PXDDataReductionModule.cc:128 }
[DEBUG]  %%%%%  Fit track candidate Nr. : 1  { module: PXDDataReduction @tracking/pxdDataReductionClasses/src/PXDInterceptor.cc:49 }
[DEBUG] appendIntercepts, checking 40 planes  { module: PXDDataReduction @tracking/pxdDataReductionClasses/src/ROIGeometry.cc:49 }
[DEBUG] before predictedIntersect  { module: PXDDataReduction @tracking/pxdDataReductionClasses/src/ROIGeometry.cc:69 }
Error in <TVectorT<double>::operator=(const TVectorT<Element> &)>: vectors not compatible
[DEBUG] after predictedIntersect  { module: PXDDataReduction @tracking/pxdDataReductionClasses/src/ROIGeometry.cc:71 }
[DEBUG] after covMatrix  { module: PXDDataReduction @tracking/pxdDataReductionClasses/src/ROIGeometry.cc:74 }
Error in <TVectorT<double>::operator()>: Request index(3) outside vector range of 0 - 0
```

- problem solved:

```
//  TVectorD predictedIntersect;
//  TMatrixDSym covMatrix(theTrack->getCardinalRep()->getDim());

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

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

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

r7793 and externals v00-05-00 (2)

➡ *unresolved problem*: Segmentation Violation occurring at different times (# events and/or # track candidate) depending on where we run the code (KEKCC or Pisa):

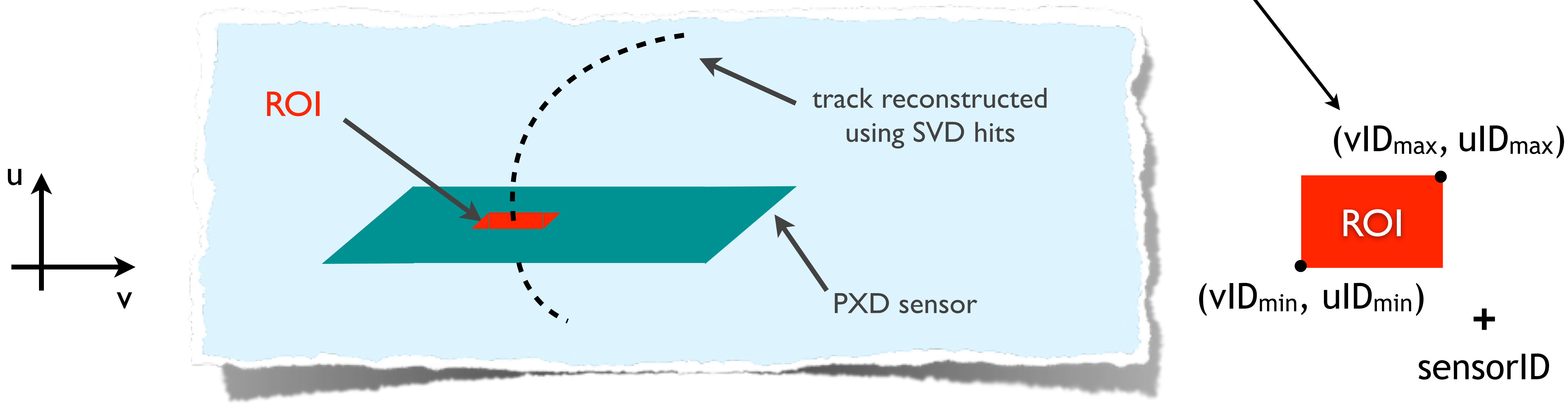
```
[DEBUG]  %%%%%  Fit track candidate Nr. : 9  { module: PXDDataReduction @tracking/pxdDataReductionClasses/src/PXDInterceptor.cc:49 }

Program received signal SIGSEGV, Segmentation fault.
0x00002aaaab111879 in TObject::TestBit (this=0x10, f=16384) at /sw/belle2/externals/v00-05-00/root/Linux_x86_64/debug/include/TObject.h:171
171    Bool_t   TestBit(UInt_t f) const { return (Bool_t) ((fBits & f) != 0); }
(gdb) where
#0  0x00002aaaab111879 in TObject::TestBit (this=0x10, f=16384) at /sw/belle2/externals/v00-05-00/root/Linux_x86_64/debug/include/TObject.h:
171
#1  0x00002aaaae3c0b45 in TVectorT<double>::IsValid (this=0x10) at include/TVectorT.h:89
#2  0x00002aaaae3fec69 in TVectorT<double>::operator() (this=0x10, ind=0) at include/TVectorT.h:211
#3  0x00002aaab9353eab in genfit::RKTrackRep::getCharge (this=0xb23edb0, state=...) at genfit2/code/trackReps/src/RKTrackRep.cc:643
#4  0x00002aaab92f9649 in genfit::KalmanFitter::processTrackWithRep (this=0xac4d248, tr=0x7fffffff9580, rep=0xb23edb0, resortHits=true) at
genfit2/code/fitters/src/KalmanFitter.cc:240
#5  0x00002aaab92b95c6 in genfit::AbsFitter::processTrack (this=0xac4d248, tr=0x7fffffff9580, resortHits=true) at genfit2/code/core/src/
AbsFitter.cc:28
#6  0x00002aaacb79308f in Belle2::PXDInterceptor::fillInterceptList (this=0xac4d240, listToBeFilled=0x7fffffff9990, trackCandList=...,
gfTrackCandToPXDIntercepts=0x7fffffff9a80) at tracking/pxdDataReductionClasses/src/PXDInterceptor.cc:62
#7  0x00002aaacc8deb6b in Belle2::PXDDataReductionModule::event (this=0x10642a0) at tracking/modules/pxdDataReduction/src/
PXDDataReductionModule.cc:146
#8  0x00002aaaab0b16b2 in Belle2::EventProcessor::processCore (this=0x7fffffffa660, startPath=..., modulePathList=Traceback (most recent
call last):
  File "/sw/belle2/tools/gcc/lib64/../share/gcc-4.7.3/python/libstdcxx/v6/printers.py", line 103, in children
    nodetype = find_type(self.val.type, '_Node')
  File "/sw/belle2/tools/gcc/lib64/../share/gcc-4.7.3/python/libstdcxx/v6/printers.py", line 43, in find_type
    field = typ.fields()[0]
```

PXDDataReduction Module Description and Performance

Software-Based Data Reduction Algorithm

[implemented in PXDDataReduction]

1. pattern recognition performed with SVD hits only:
   - ★ TrackCand list produced by *VXDTF* (or *MCTrackFinder* for testing purposes only)
2. fit the TrackCand using the standard kalman filter and produce a Track
   - ★ the fit is done in both directions: first inward, then outward
3. the Track is extrapolated on each of the 40 planes containing a PXD sensor
   - ★ obtain an extrapolation point on the plane and the associated statistical errors σ_{stat}
4. a rectangular region is defined given $\sigma_{stat,}$ a systematic error σ_{syst} and a total number of $\sigma = \text{sqrt}(\sigma^2_{stat} + \sigma^2_{syst})$ in each direction u,v
5. the region is intersected with the sensor and then translated in pixels ID



Definition of the Figures of Merit

➡ Definition of efficiency for PXD Data Reduction:

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

inefficiencies of the pattern recognition are factorized!!

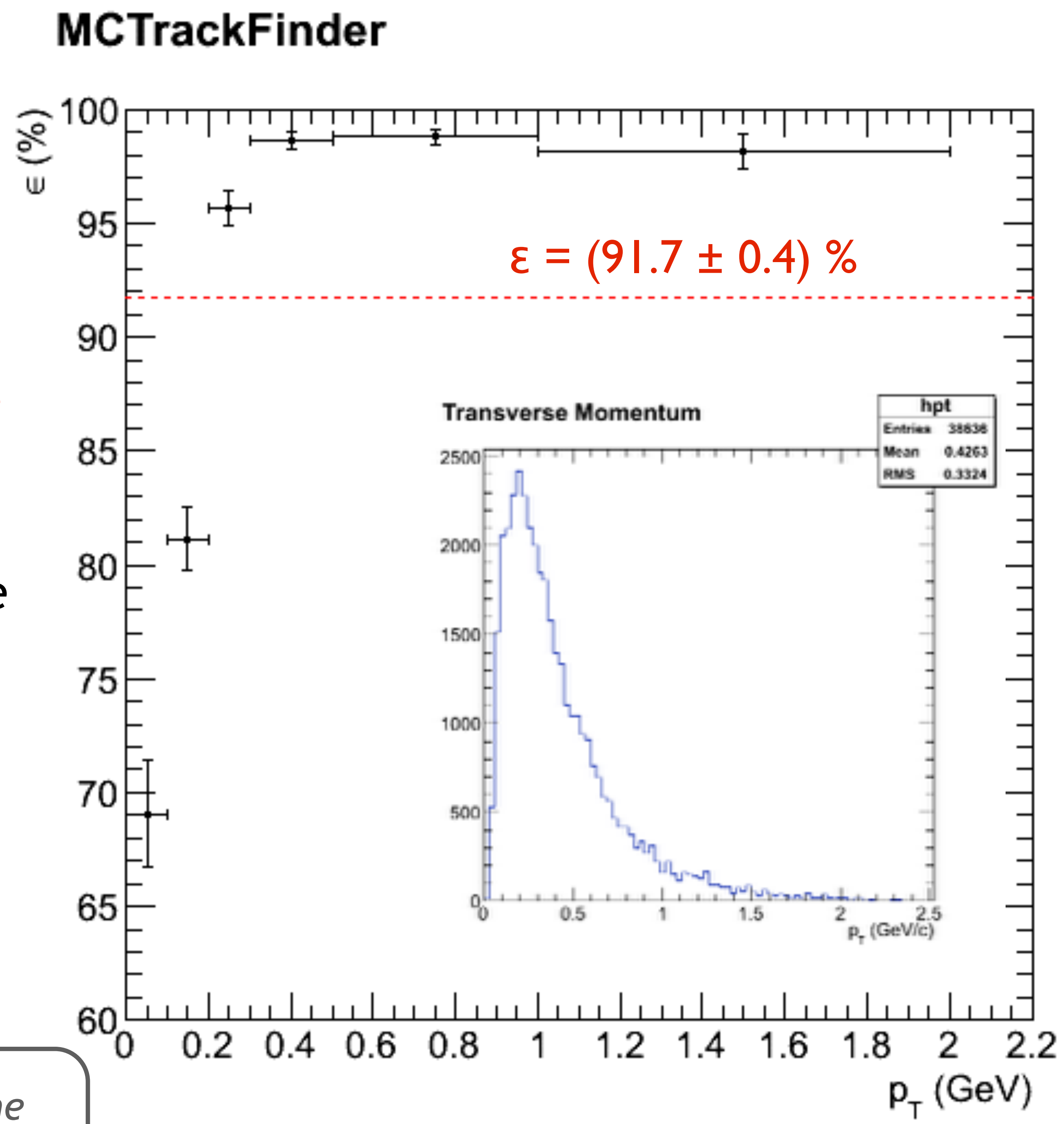
➡ Definition of data reduction factor:

$$r = \frac{\text{< \# pixels in ROI/event >}}{\text{250*768 pixels/module * 40 modules}}$$

➡ execution time: we run on the HLT, we need to be fast: benchmark = 1ms/track

ROI Efficiency with MCTrackFinder

- We simulate *100 events* using EvtGen and use the *MCTF* as *pattern recognition*:
  - ~9.5 tracks/event
  - ~4.2 PXDDigits/track
- *Efficiency* = (91.7±0.4)% = 3645/3974 PXDDigits
- *Inefficiency* mostly due to failures in fitting the track and finding an intercept with the sensor planes
  - 321 PXDDigits (97.5% of ineff.) are lost since no intercept is found
    - increasing the size of ROI will not have a significant impact on the efficiency
  - 8 PXDDigits (2.5% of ineff.) are lost since a the ROI is defined on the wrong sensor.

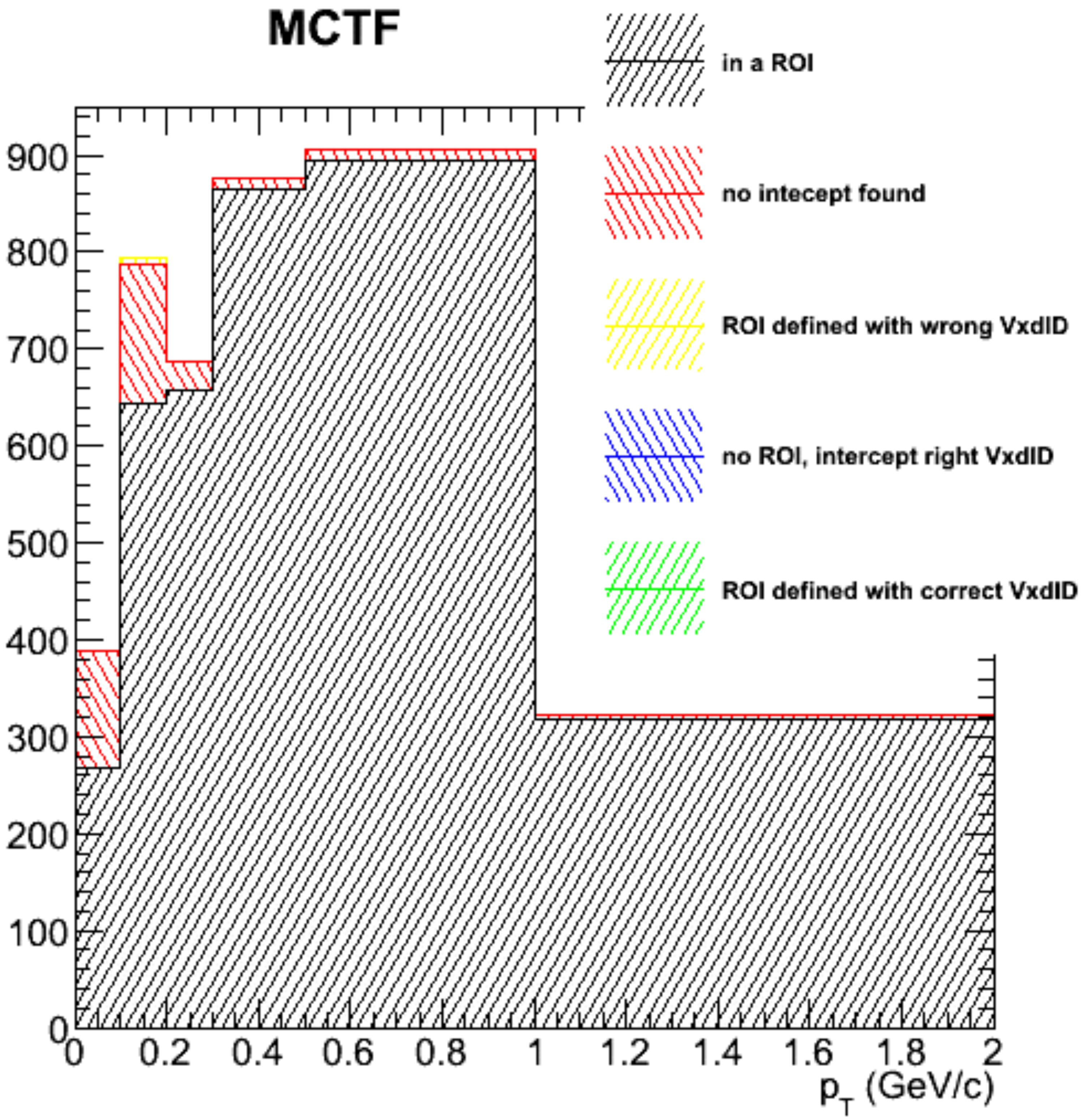


the estimated efficiency is compatible with the one observed with the old genfit

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

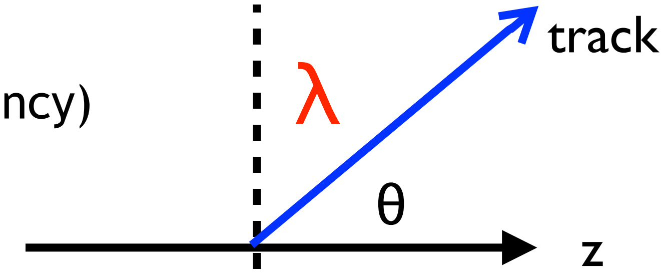
inefficiencies of the pattern recognition are factorized!!

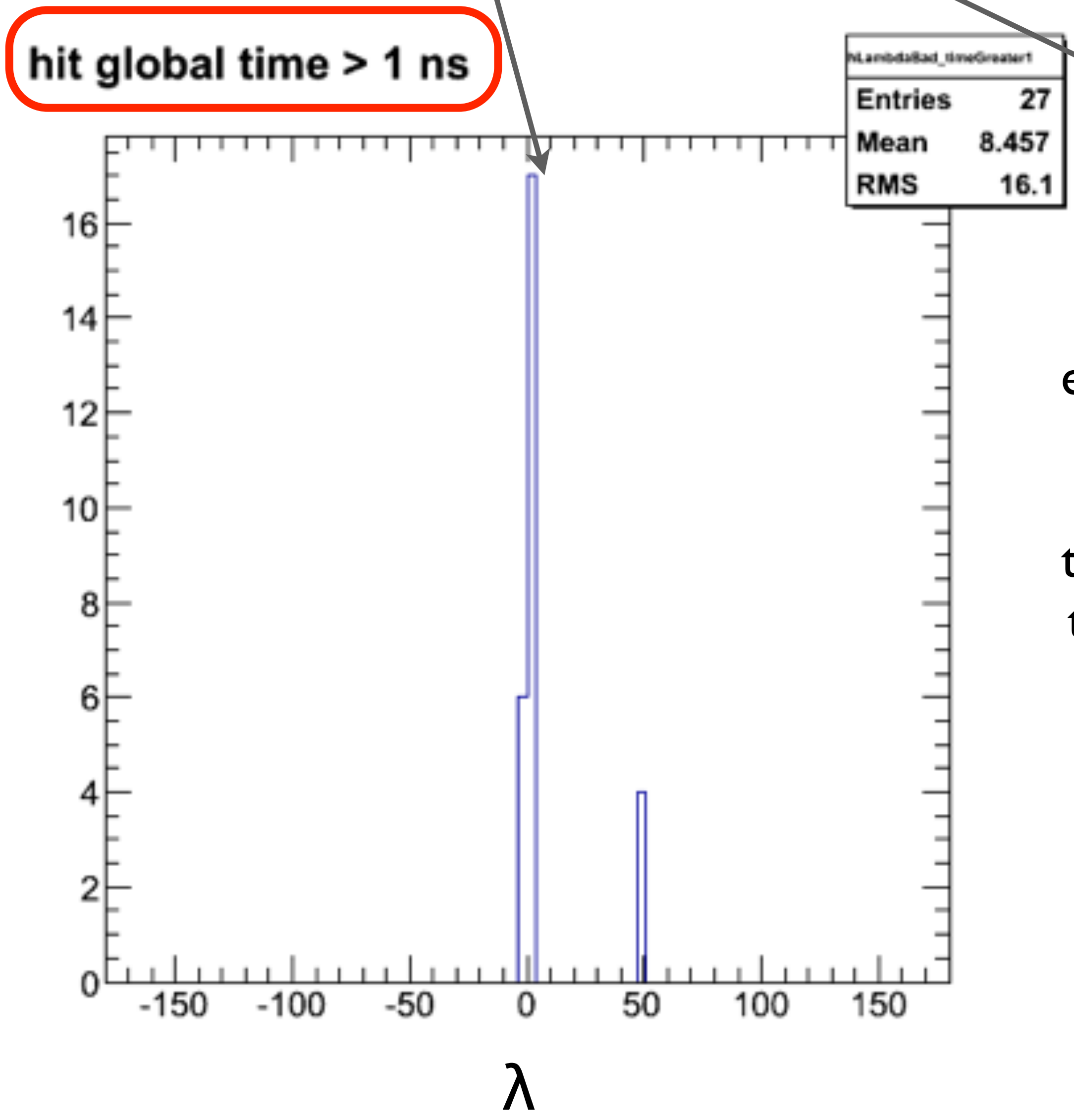
PXDDigits classification (MCTF)



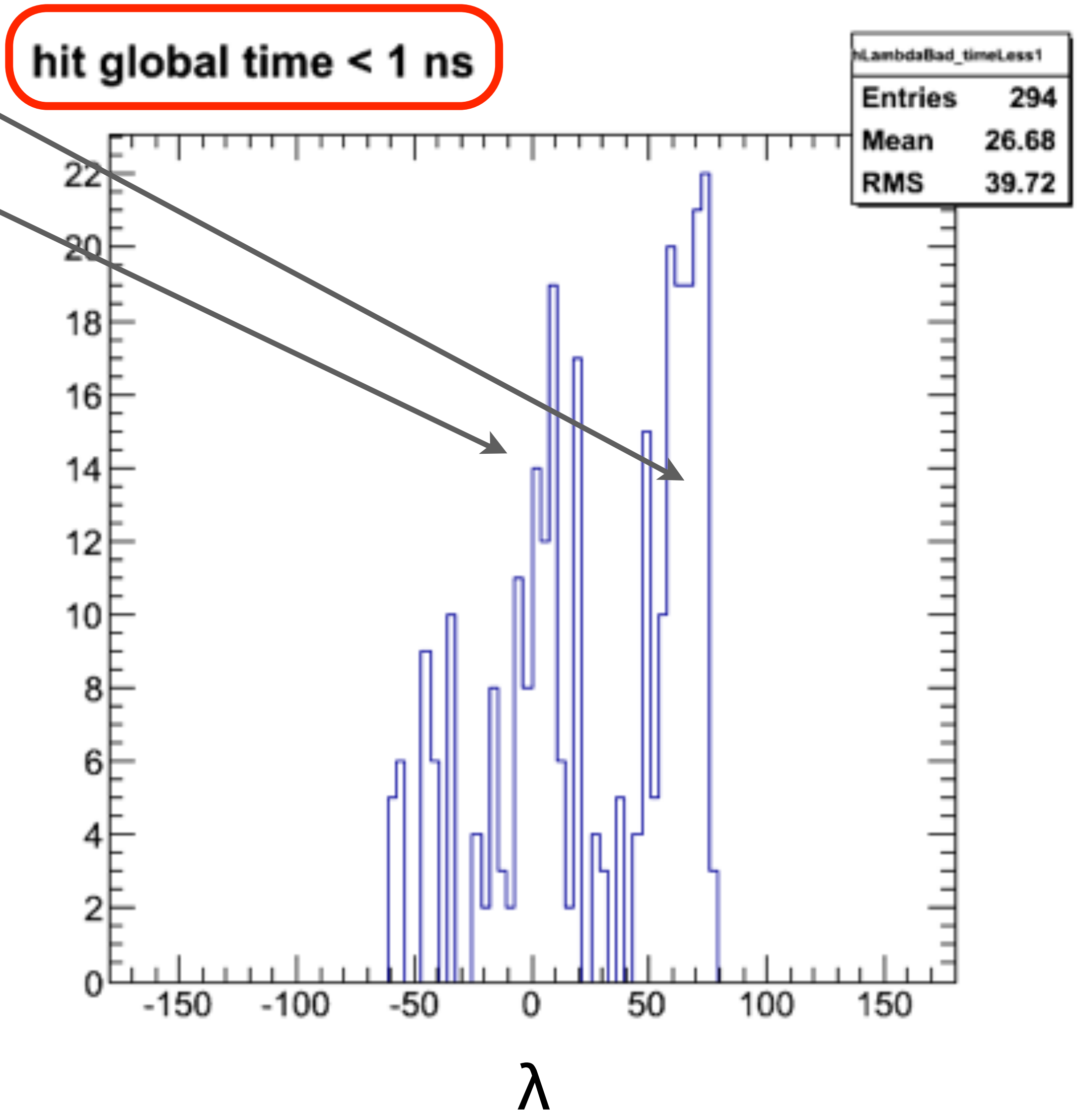
What Hits are we Missing?

- ➡ mainly hits of low transverse-momentum tracks
- ➡ later hits of tracks looping on the plane $z \approx 0$ (~10% of the inefficiency)
- ➡ first hits of tracks at $\lambda \approx 0^\circ$ and $\lambda \approx 65^\circ$ (~ 90% of the inefficiency)





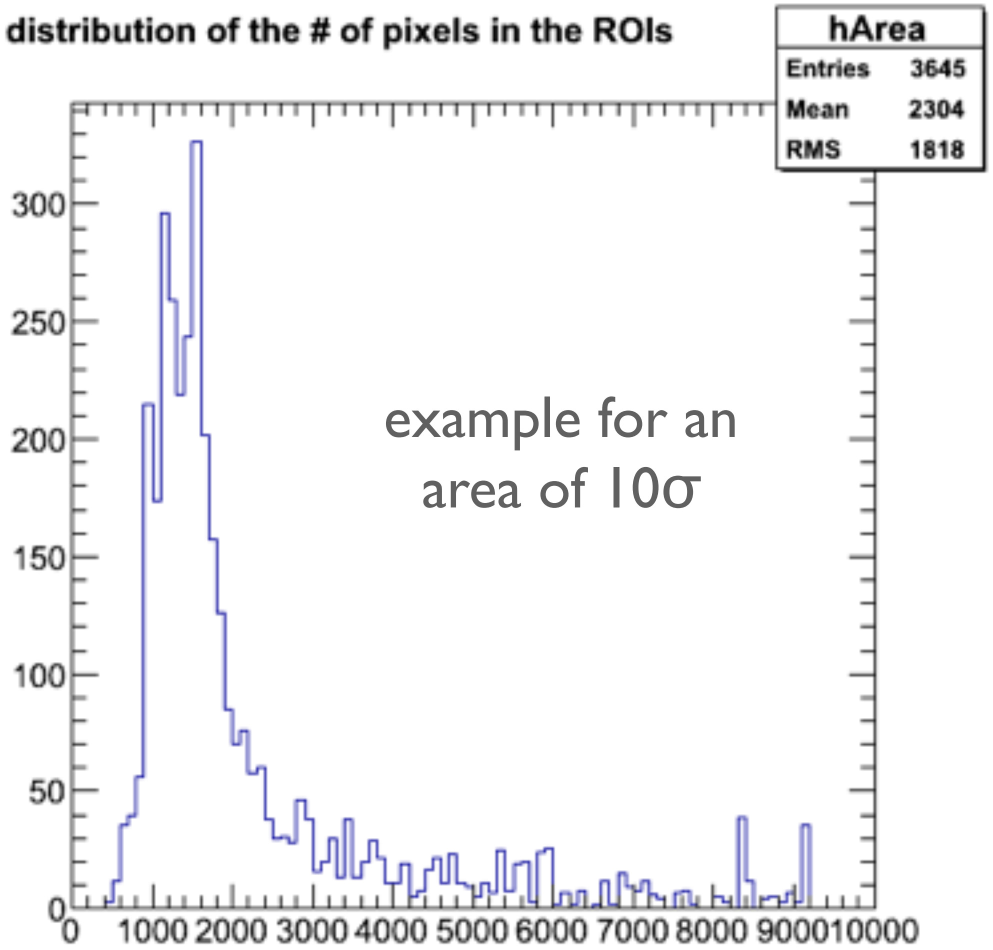
entries = tracks for which the fit has a “bad track status” or the intercept is not found



Data Reduction Factor & Execution Time

➡ Data Reduction Factor = $\frac{\text{< \# pixels in ROI/event >}}{\text{250*768 pixels/module * 40 modules}}$

- 1% taking 10σ in each direction
- to be tested with background...



example for an area of 10σ

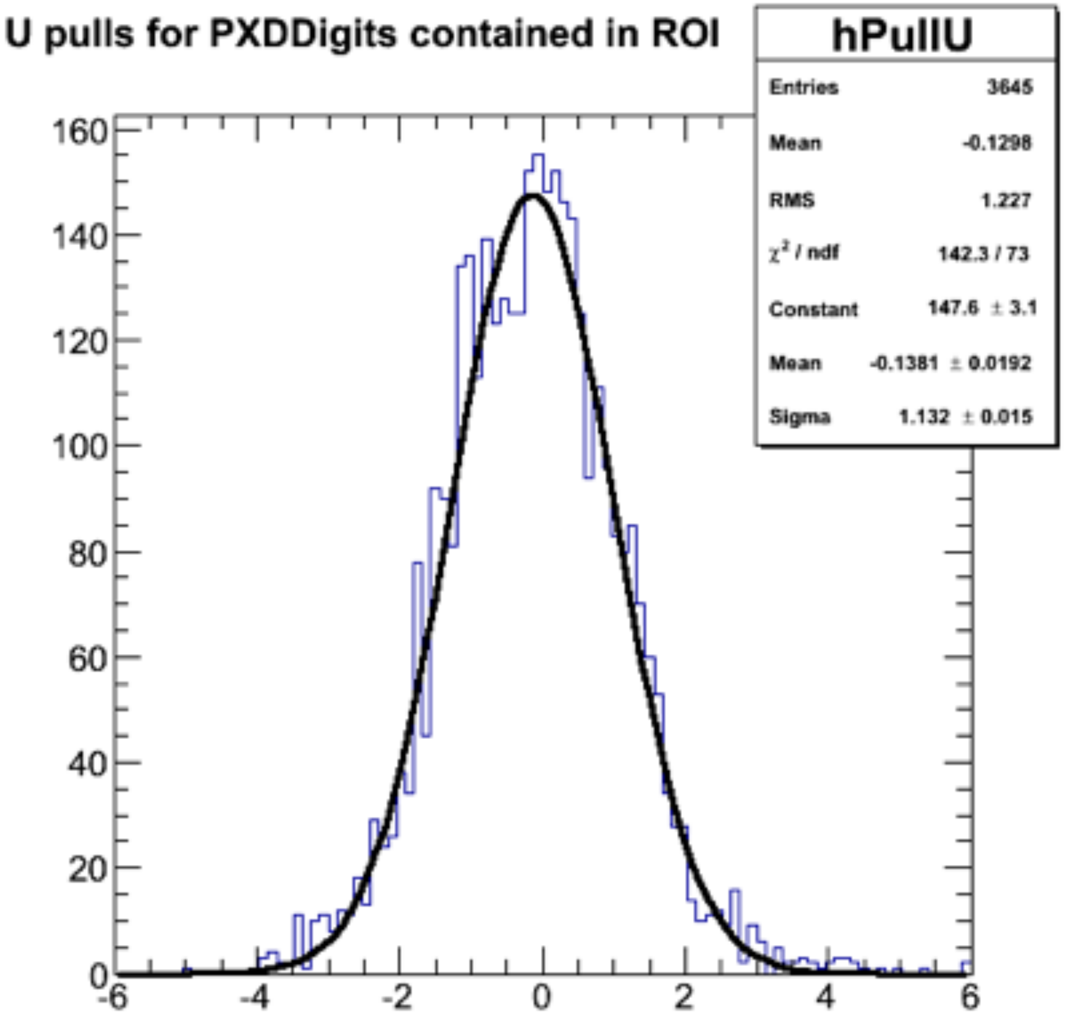
➡ Execution time (code compiled in *debug* mode):

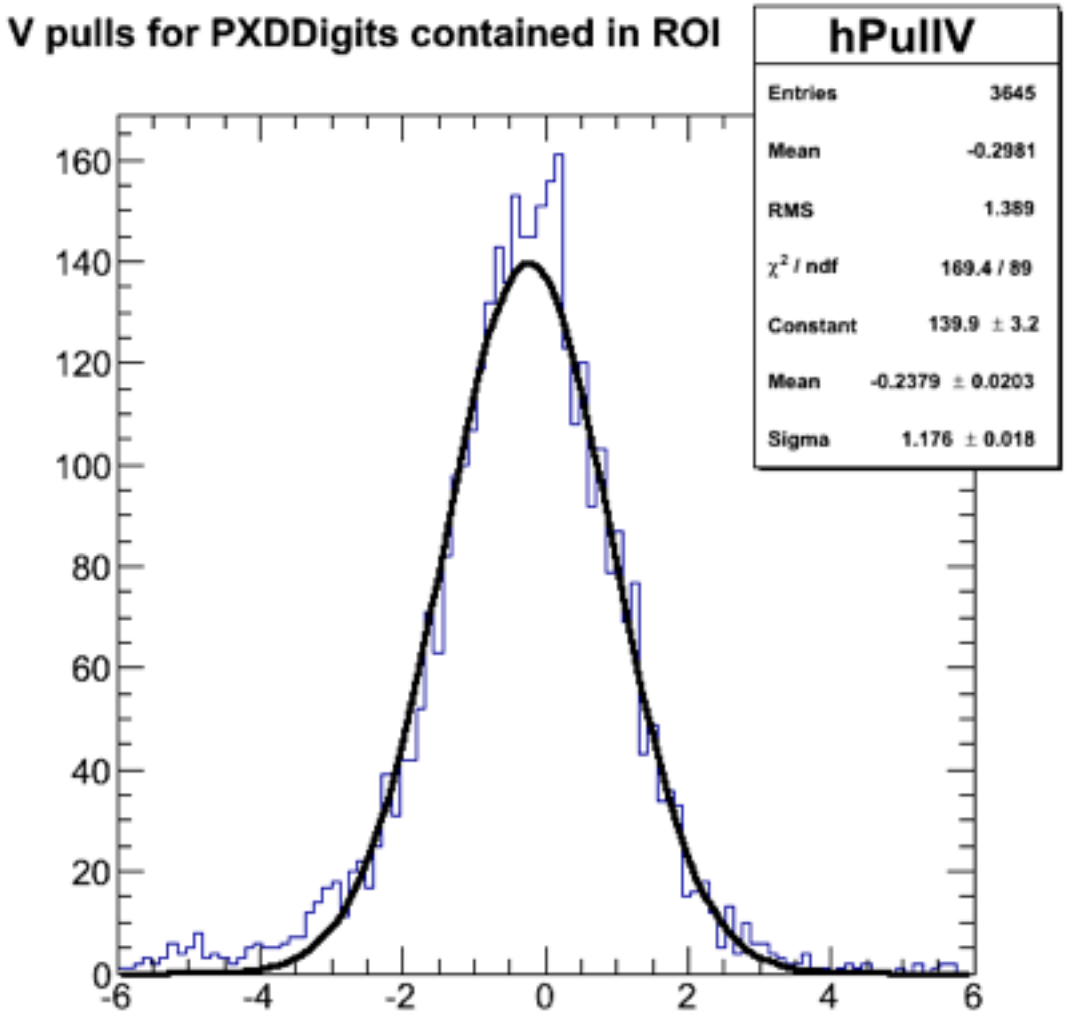
★ 600-700 ms/track (with 5 iterations of the kalman filter)

- at least 10 times slower than before (genfit1, externals v00-04-02)!
- the benchmark is 1ms/track...

the deterioration of the execution time is not understood and need to be investigated

ROI Definition, the U and V pulls





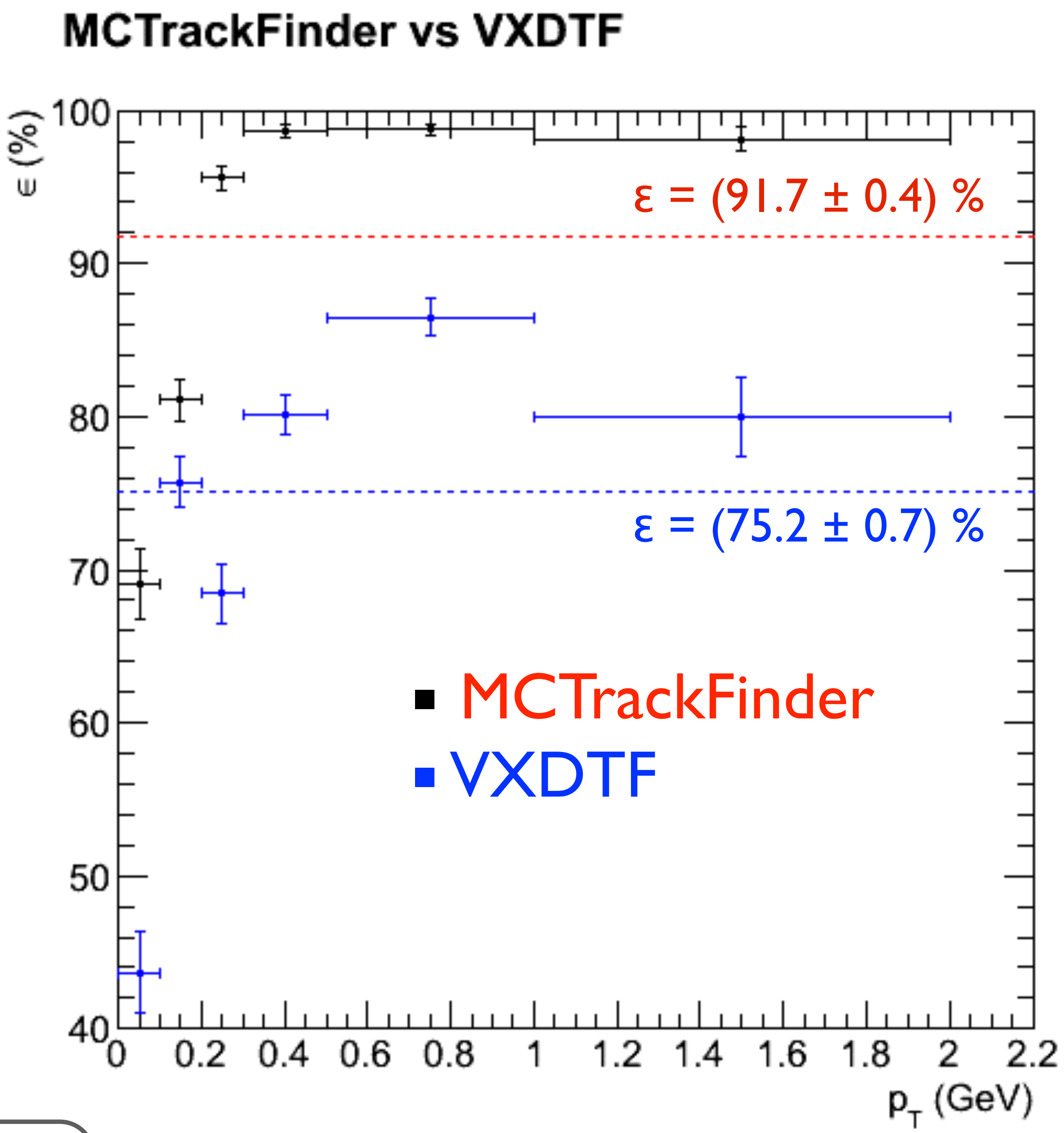
- ➡ pull = (intercept – center of the fired pixel)/(stat error on intercept)
- ➡ U (V) pull are slightly biased to negative values by 14% (24%) of the stat error
- ➡ U, V stat errors are underestimated by ~15%

MCTrackFinder vs VXDTF

- We simulate 100 events using EvtGen and use the *VXDTF* as *pattern recognition*:
  - ~8.6 tracks/event
  - ~4.1 PXDDigits/track
- Efficiency = (75.2±0.7)% = 2670/3552 PXDDigits
  - something strange in the dependence on transverse momentum!!
- Inefficiency mostly due to failures in fitting the track and finding an intercept with the sensor planes
  - 796 PXDDigits (90.2% of ineff.) are lost since no intercept is found
  - 77 PXDDigits (8.7% of ineff.) are lost since a the ROI is defined on the wrong sensor

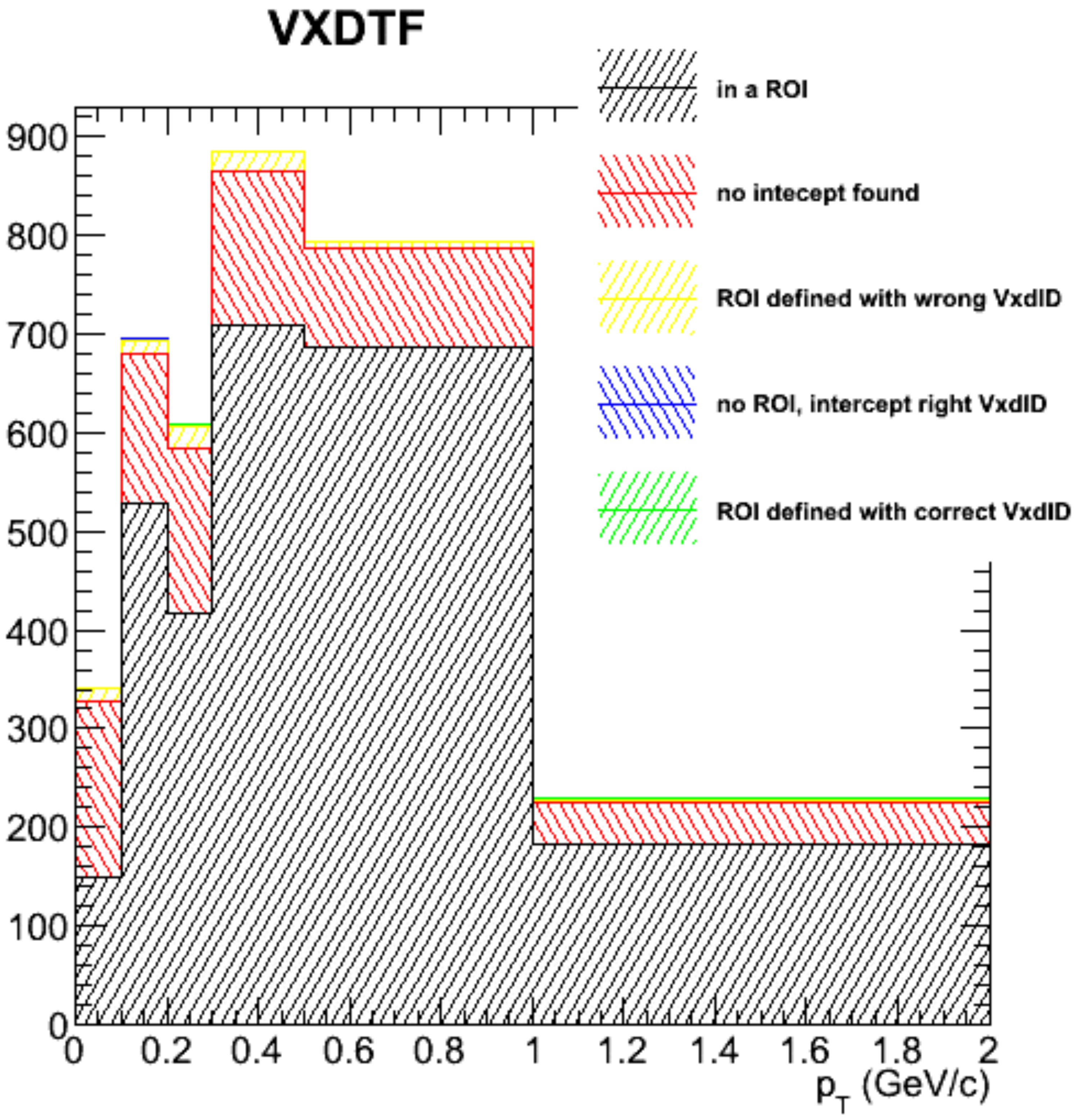
$$\varepsilon = \frac{\text{\# PXDDigits inside a ROI}}{\text{total \# PXDDigits of GFTrackCand}}$$

inefficiencies of the pattern recognition are factorized!!

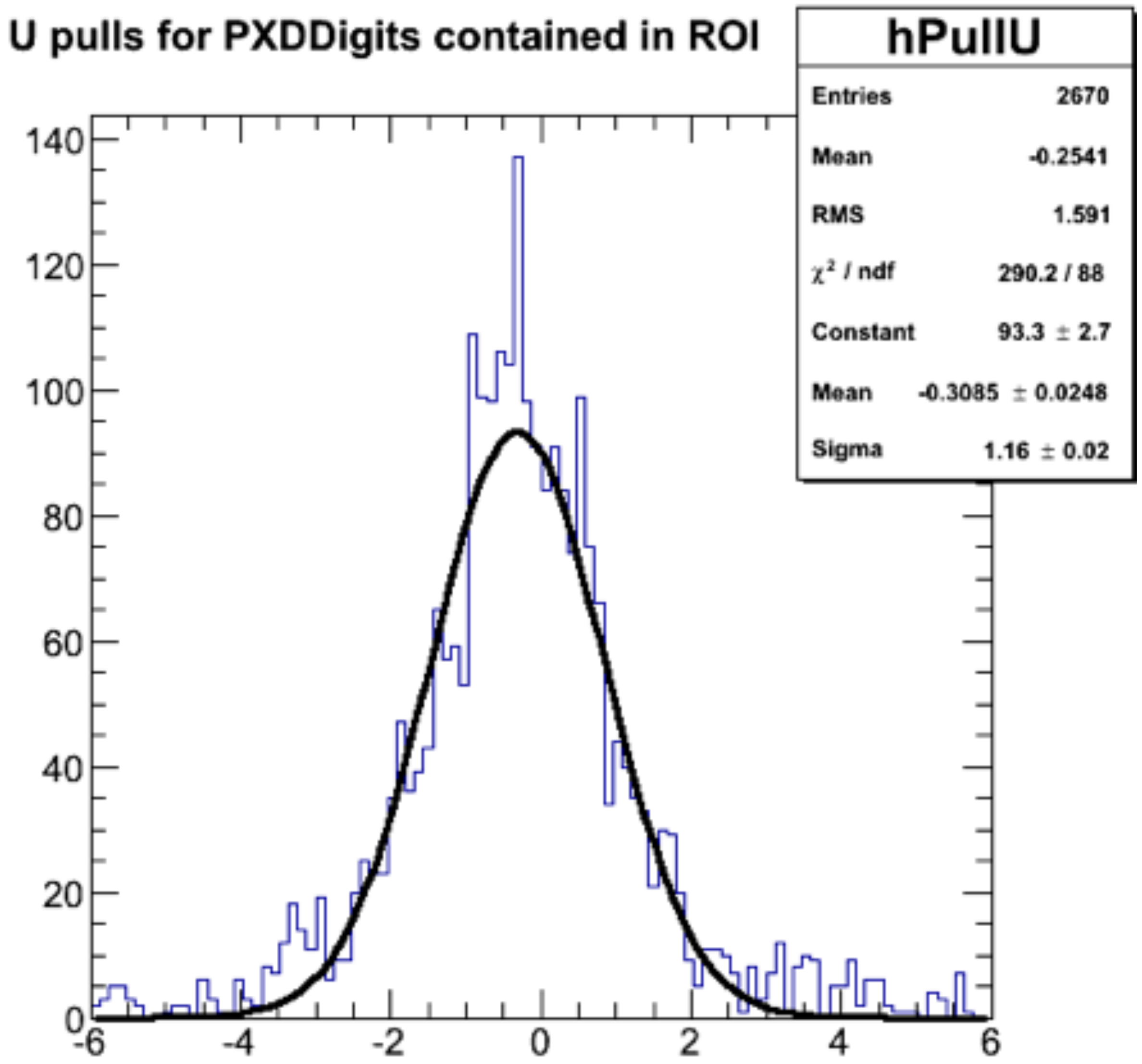


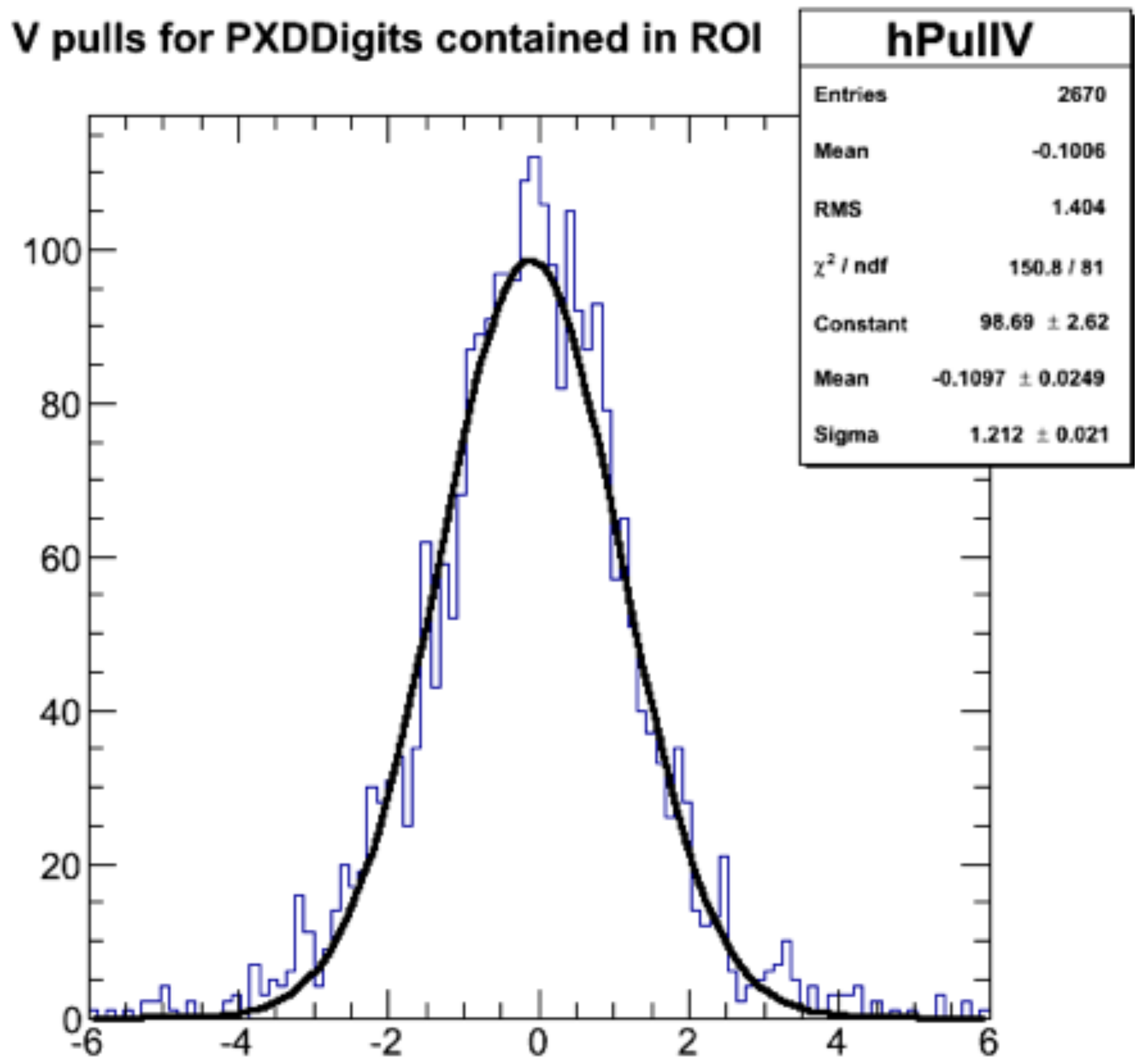
the estimated efficiency is 5% worse than the one observed with the old genfit

PXDDigits classification (VXDTF)



ROIs with the VXDTF





- ➡ U (V) Pulls are negatively biased by 30% (11%) of the statistical error
- ➡ the statistical errors are underestimated by ~15-20%
- ➡ Data Reduction Factor = 0.9% (average area compatible with the one using MCTF)
- ➡ Execution time = 740 ms/track (5 iterations of the kalman filter)

Performance with Beam Test Geometry

- We simulate 100 events using particleGun (2GeV e^-, no beam divergence) and use the *VXDTF* as *pattern recognition*:
  - ~0.45 tracks/event (0.32 with MCTrackFinder)
  - ~1.9 PXDDigits/track (3.1 with MCTrackFinder)
- Efficiency = (97.5 ± 0.7)% ((99.2±0.4)% with MCTF, (99.0±0.2)% with genfit1)
- Data Reduction Factor = 0.01% (same with MCTrackFinder)
- Execution time = 5 ms/track (same with MCTrackFinder) - both in debug mode!

Efficiency ~compatible with the one obtained with the old genfit,
execution time now a factor 2 slower

What’s next

…on the Short Term (test-beam)

- ➡ solve the Segmentation Violation ~ ASAP
- ➡ write the DQM Module ~ by Christmas

…on the Long Term

- ➡ Improve the efficiency with VXDTF:
  - ★ apply a beam-spot constraint to the track fit
  - ★ specify the seed of the momentum for the fit
  - ★ “manually” fit the hits to a helix
- ➡ Improve the speed of the module (benchmark is 1ms/track)
  - ★ understand the huge increase in execution time
  - ★ can avoid evaluation of the covariance matrix to save some time
  - ★ “manual” fit to a helix can be faster

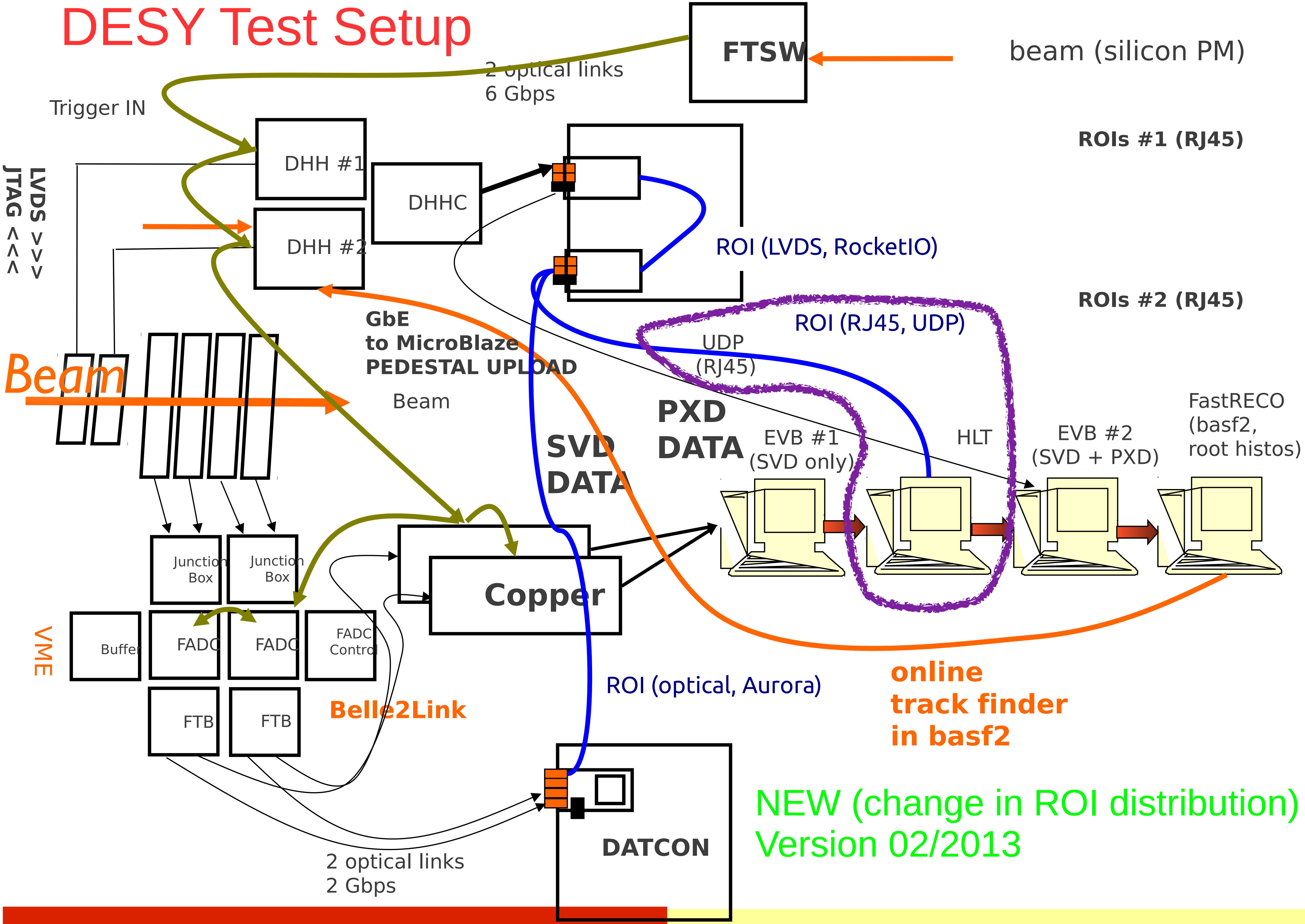
Conclusions

- Important changes in the software lead to:
  - unresolved Segmentation Violation problem to be solved ASAP
  - first (very) preliminary results show a deterioration of the efficiency and an important reduction of the execution speed
- Next weeks will be focused to solve the runtime problems and prepare for the TB (efficiency and speed are not an issue for the TB)
- On the long-term the focus will be on improving the efficiency and the speed of the Module

Thank You!

backup-slides

DESY Test Setup

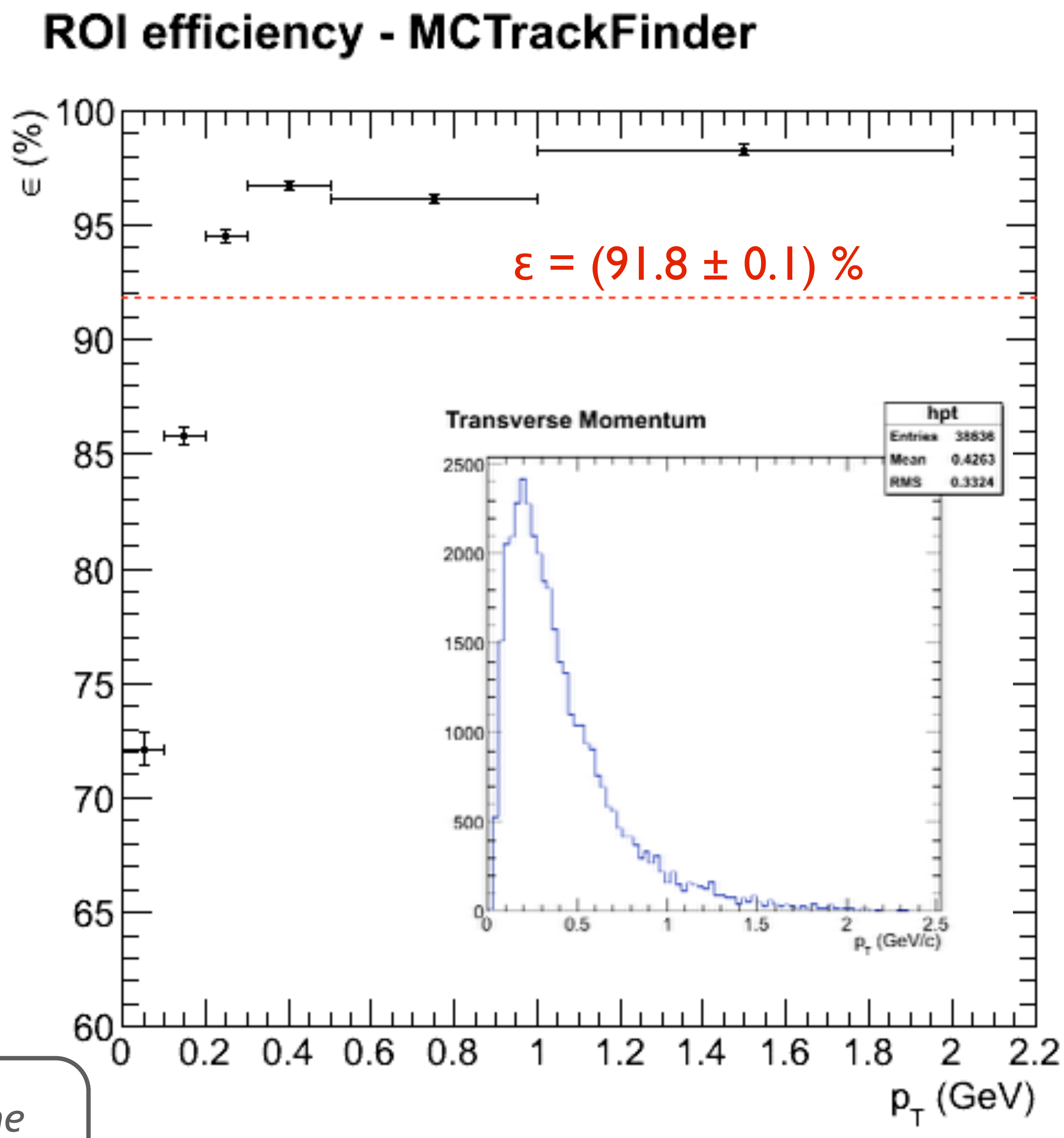


NEW (change in ROI distribution)
Version 02/2013

[genfit1 & MCTrackFinder]

ROI Efficiency, p_T dependence

- ➡ *Efficiency* = (91.8±0.1)% = 38636/42072 PXDDigits
    - strongly dependent on the *transverse momentum*
- ➡ *Inefficiency* mostly due to failures in fitting the track and finding an intercept with the sensor planes
    - 94% of the times no intercept is found
        - increasing the size of ROI will not have a significant impact on the efficiency
    - 6% of the times a ROI is defined, 95% of which the sensor is the wrong one.

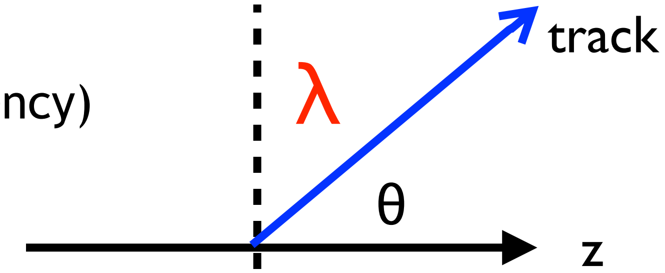


$$\varepsilon = \frac{\text{\# PXDDigits inside a ROI}}{\text{total \# PXDDigits of GFTrackCand}}$$

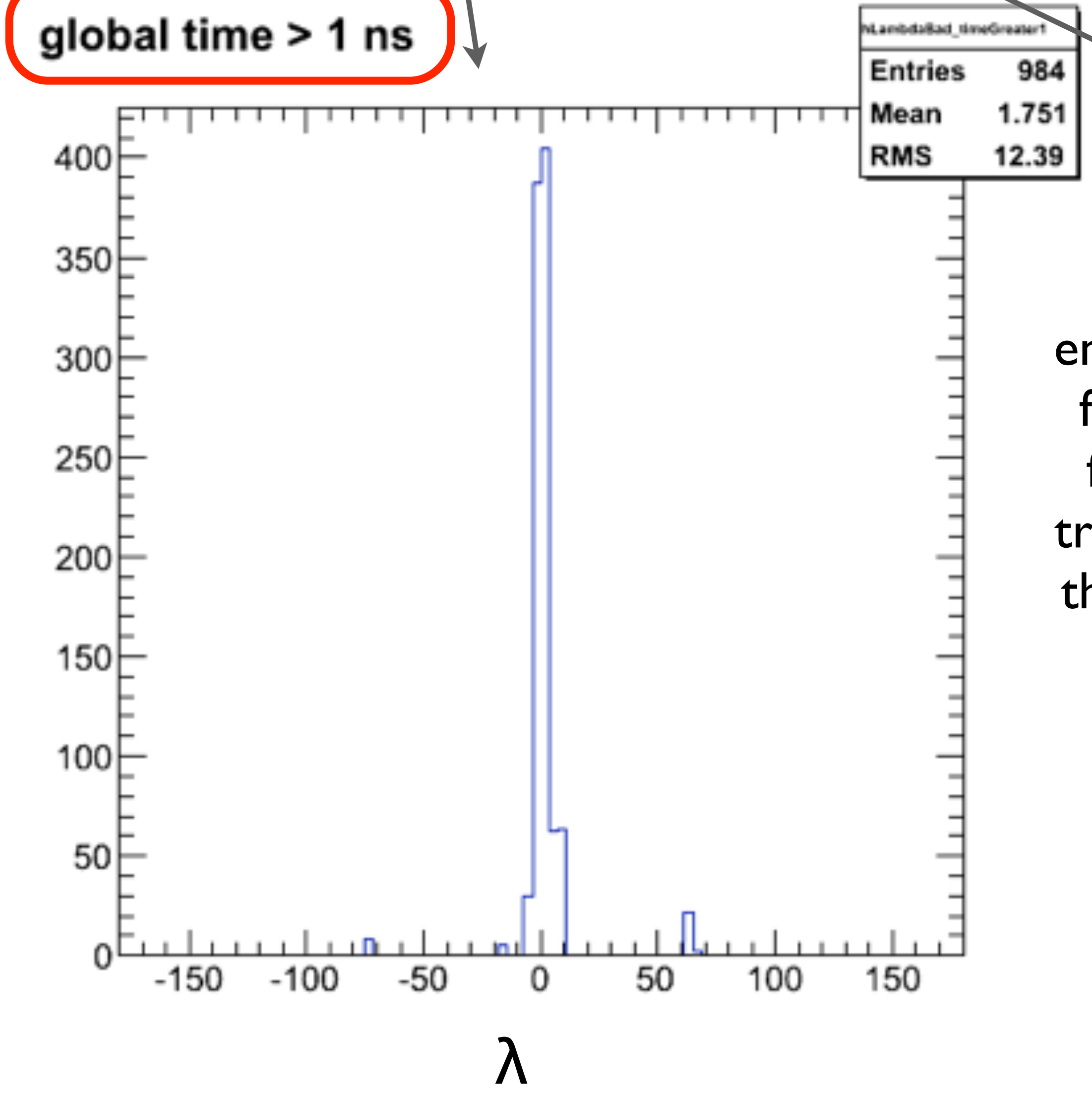
inefficiencies of the pattern recognition are factorized!!

What Hits are we Missing?

- ➡ mainly hits of low transverse-momentum tracks
- ➡ later hits of tracks looping on the plane z ≈ 0 (~30% of the inefficiency)
- ➡ first hits of tracks at λ≈0° and λ≈65° (~ 70% of the inefficiency)

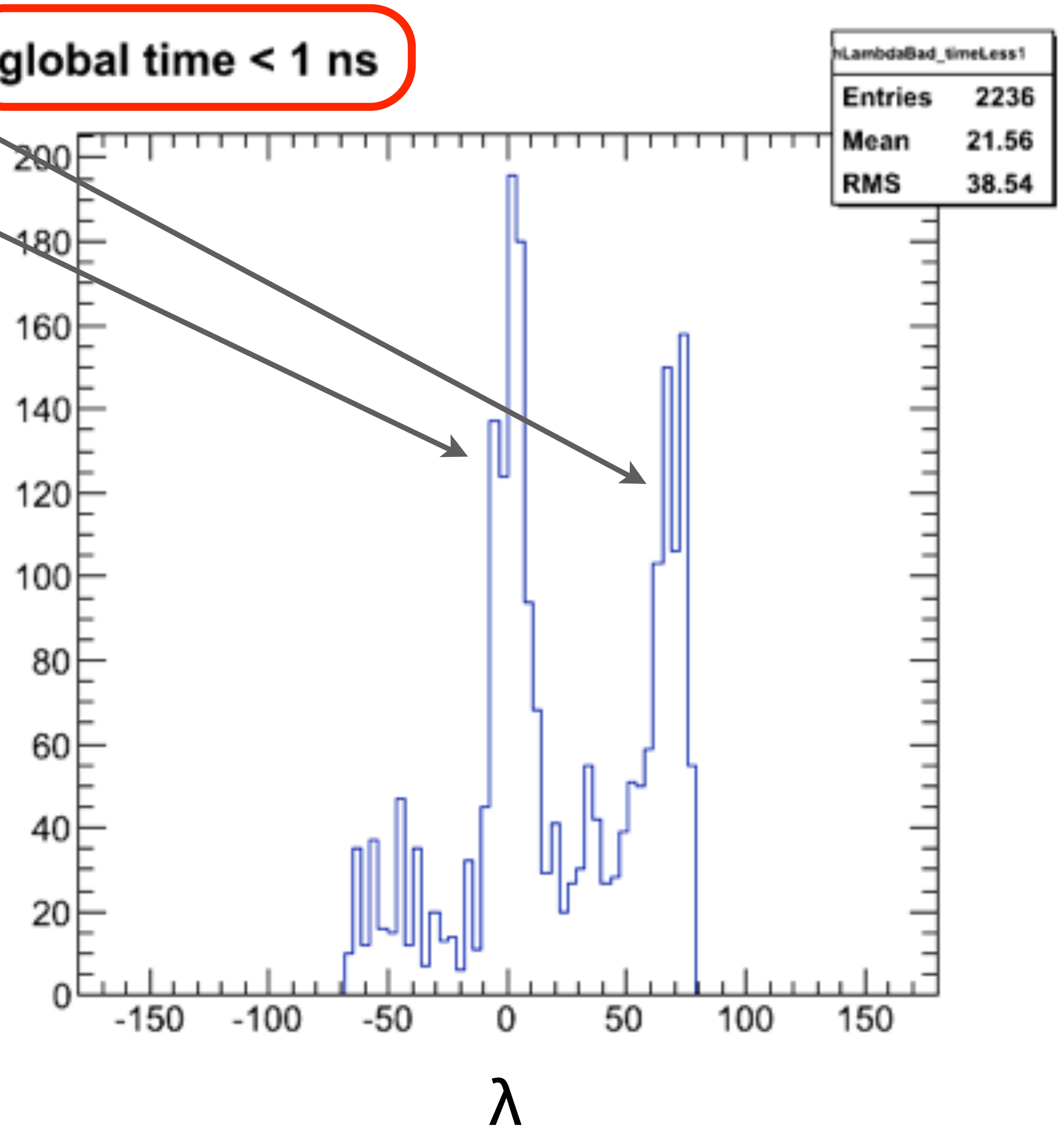


global time > 1 ns

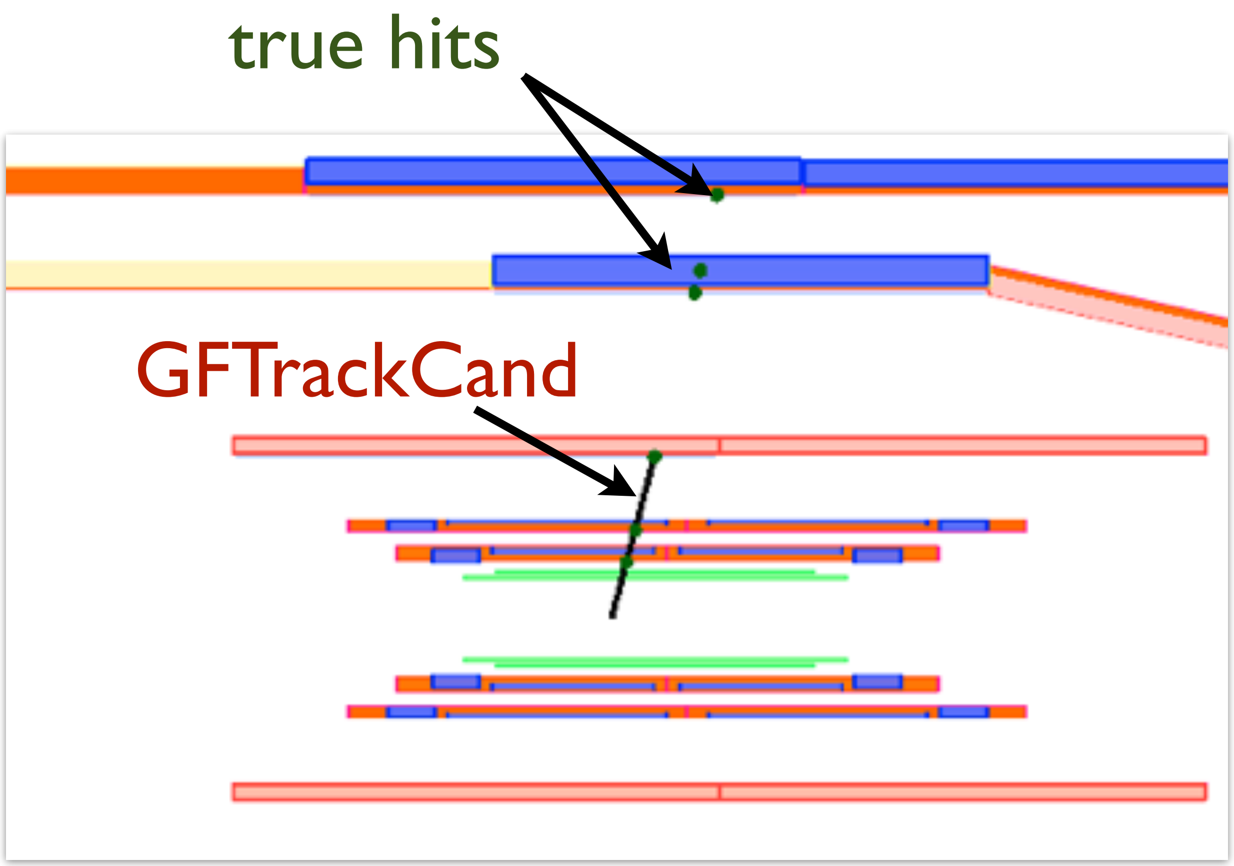


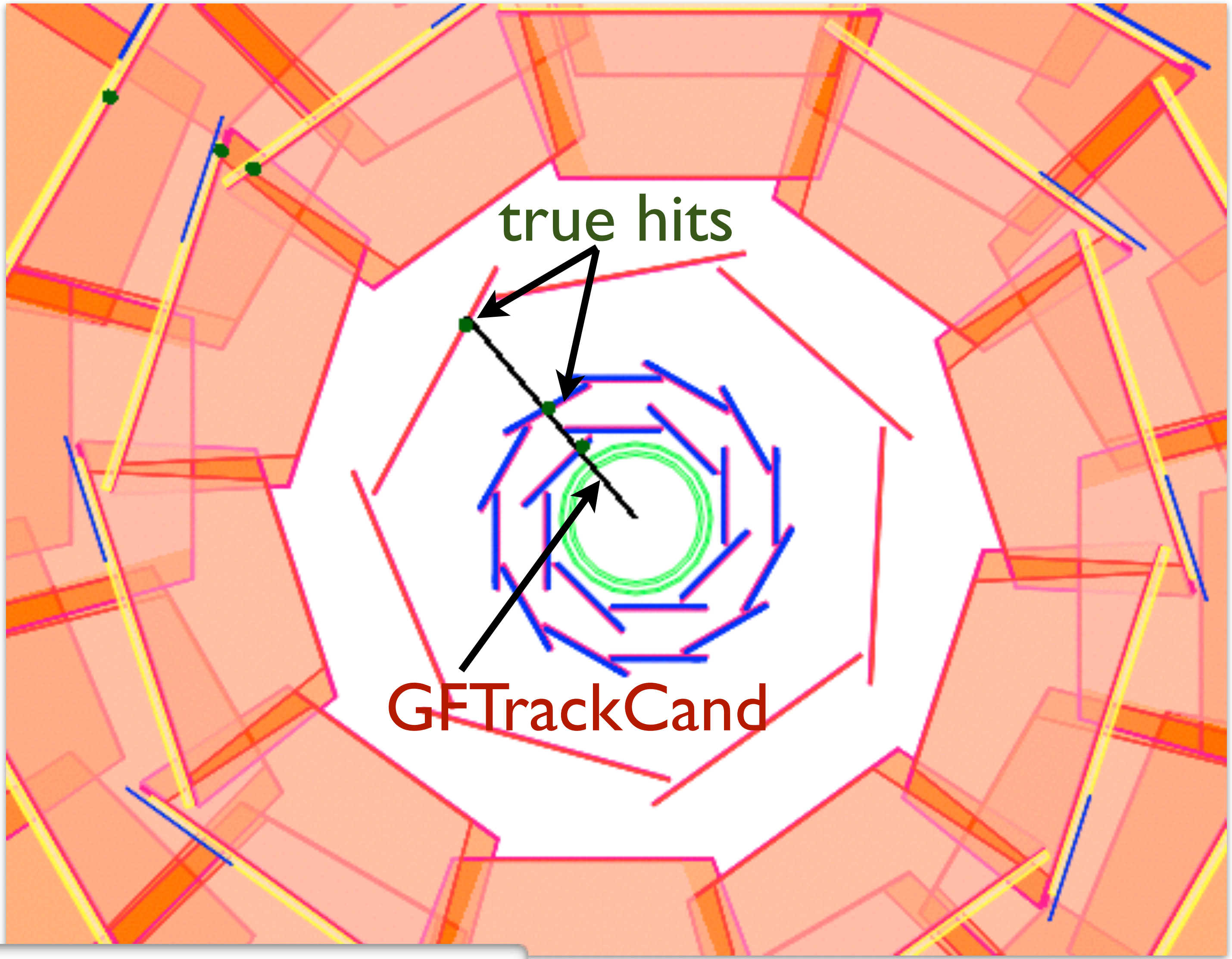
entries = tracks for which the fit has a “bad track status” or the intercept is not found

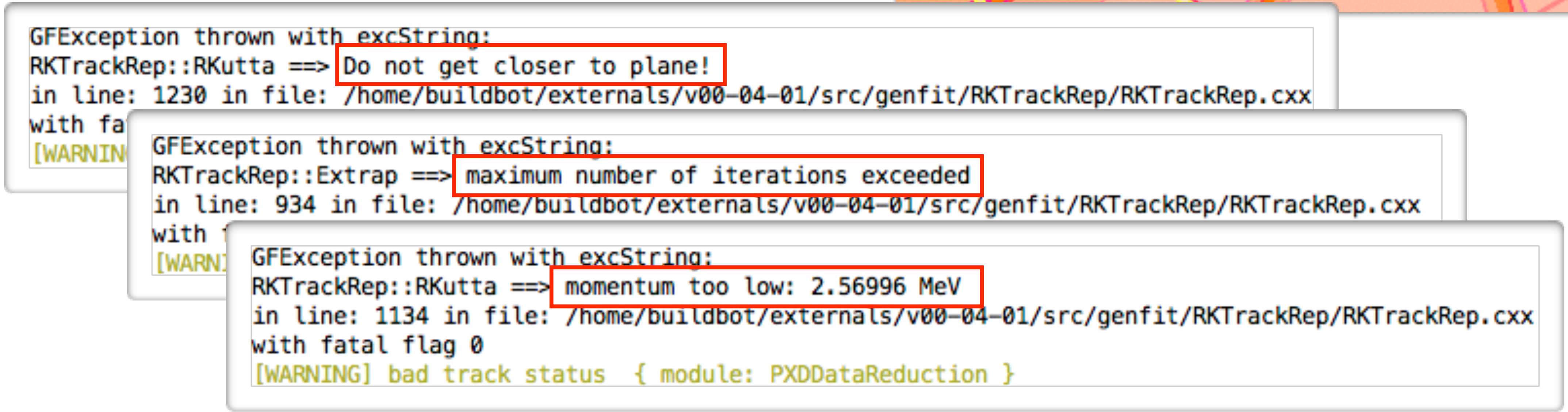
global time < 1 ns



Inefficiency due to *Bad Track Status*





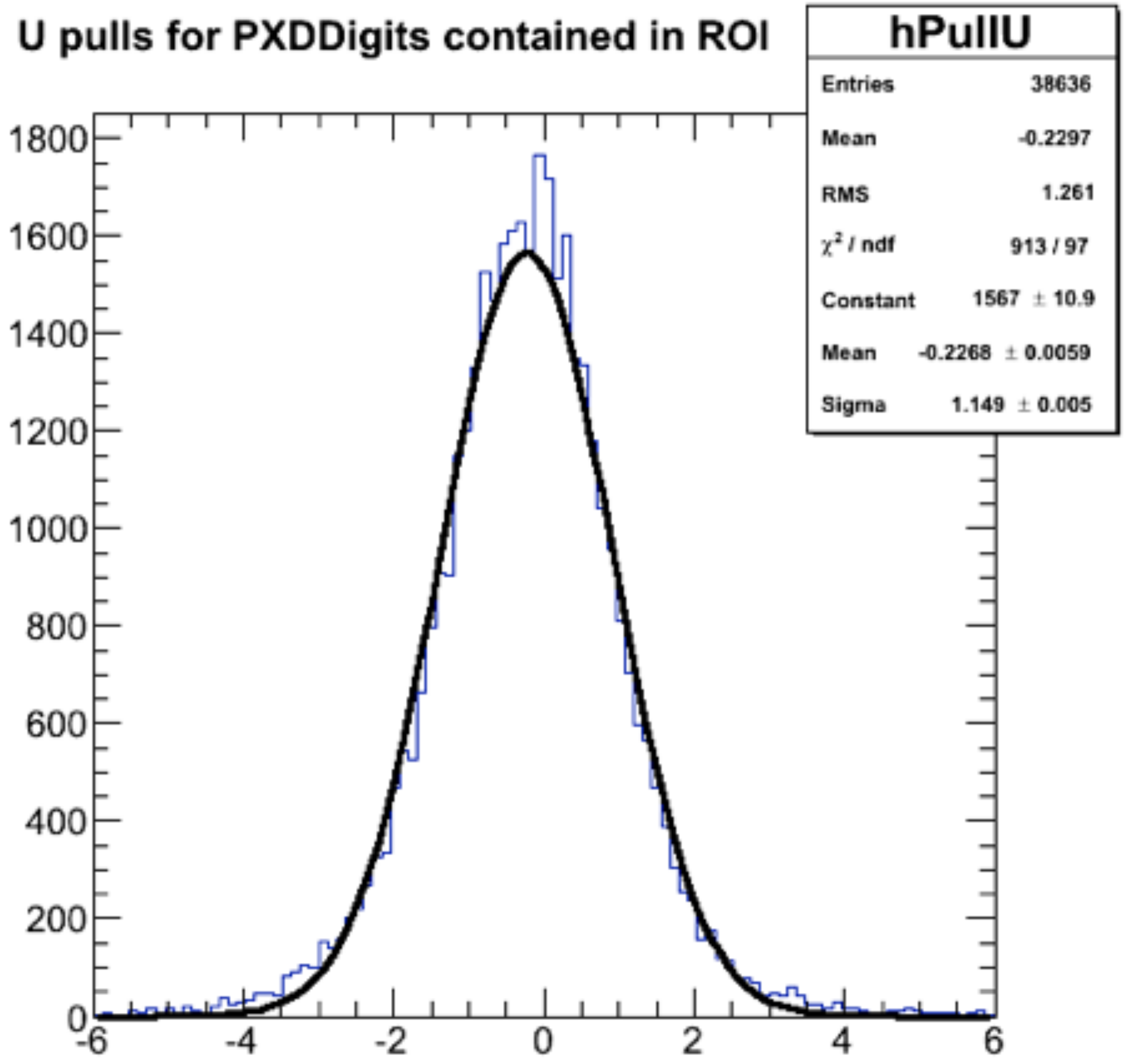


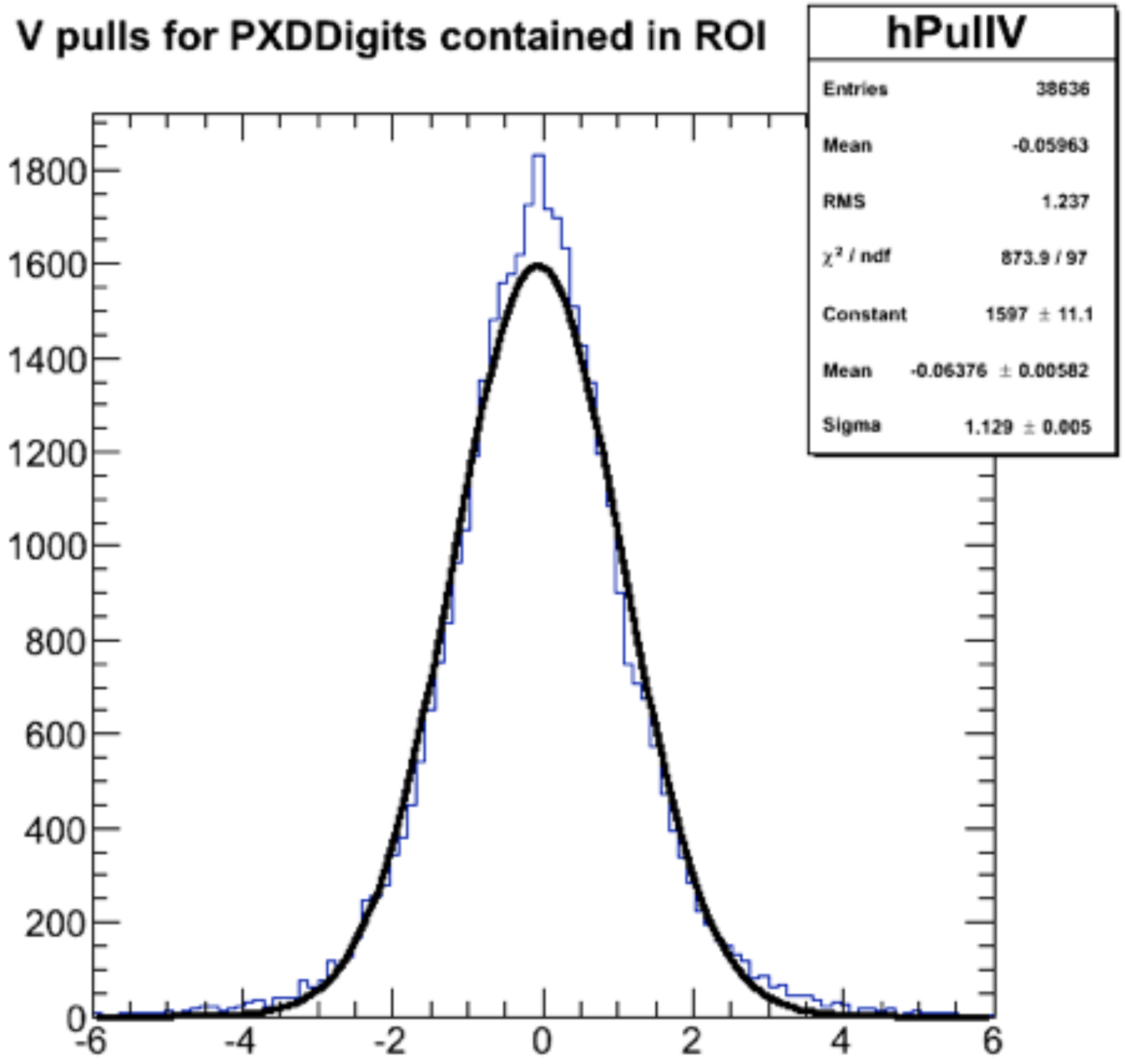
```
GFException thrown with excString:
RKTrackRep::RKutta ==> Do not get closer to plane!
in line: 1230 in file: /home/buildbot/externals/v00-04-01/src/genfit/RKTrackRep/RKTrackRep.cxx
with fa
[WARNIN
```

```
GFException thrown with excString:
RKTrackRep::Extrap ==> maximum number of iterations exceeded
in line: 934 in file: /home/buildbot/externals/v00-04-01/src/genfit/RKTrackRep/RKTrackRep.cxx
with f
[WARNI
```

```
GFException thrown with excString:
RKTrackRep::RKutta ==> momentum too low: 2.56996 MeV
in line: 1134 in file: /home/buildbot/externals/v00-04-01/src/genfit/RKTrackRep/RKTrackRep.cxx
with fatal flag 0
[WARNING] bad track status  { module: PXDDataReduction }
```

ROI Definition, the U and V pulls



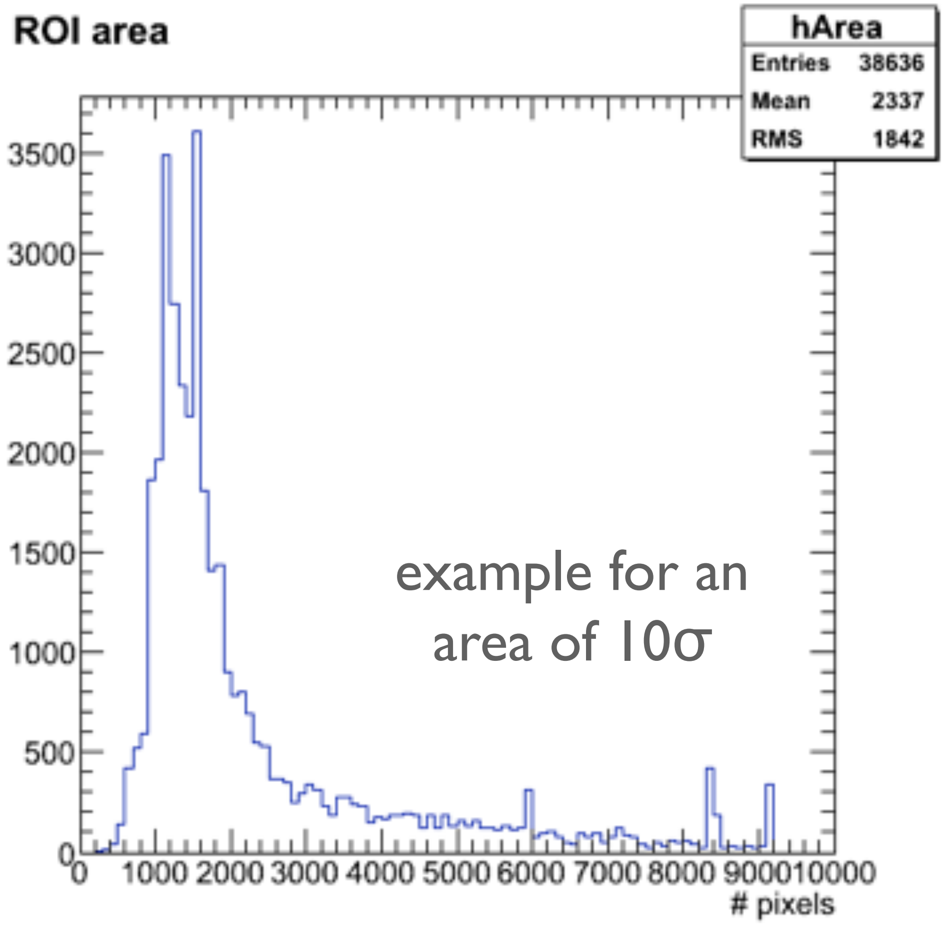


- pull = (intercept – center of the fired pixel)/(stat error on intercept)
- U (V) pull are slightly biased to negative values by 20% (5%) of the stat error
- U, V stat errors are underestimated by ~10-15%

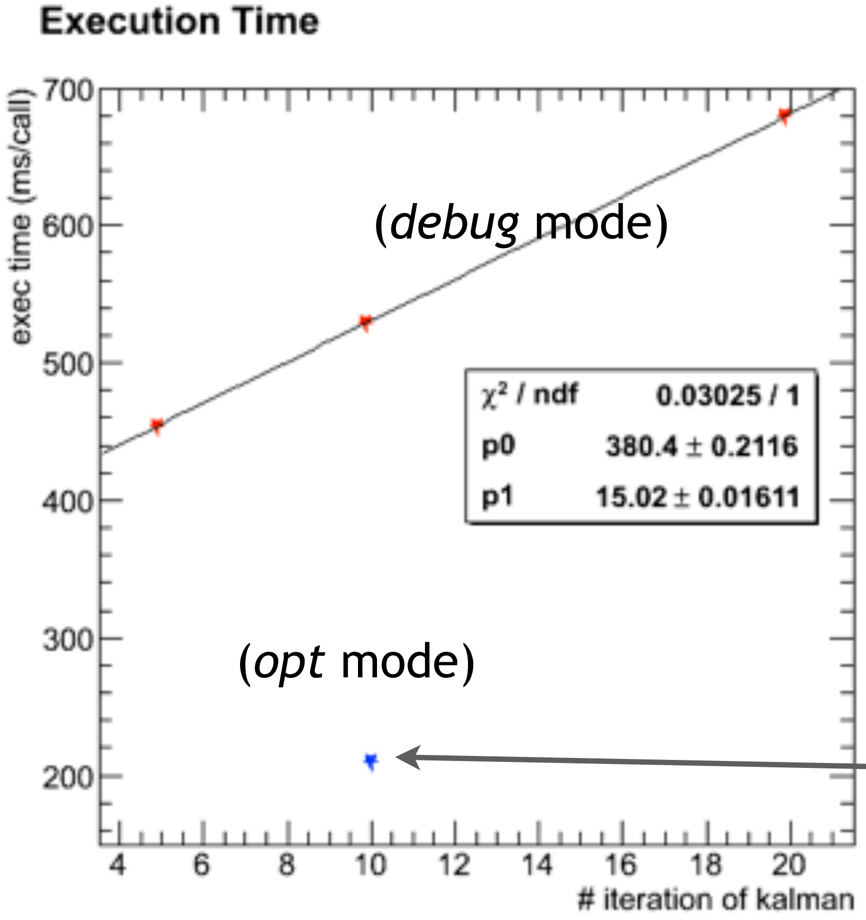
Data Reduction Factor & Execution Time

- Data Reduction Factor = $\dfrac{\text{< \# pixels in ROI/event >}}{\text{250*768 pixels/module * 40 modules}}$
  - 1.2% taking 10σ in each direction
  - 0.4% taking 5σ in each direction

(to be tested with background)



example for an area of 10σ



(*debug* mode)

(*opt* mode)

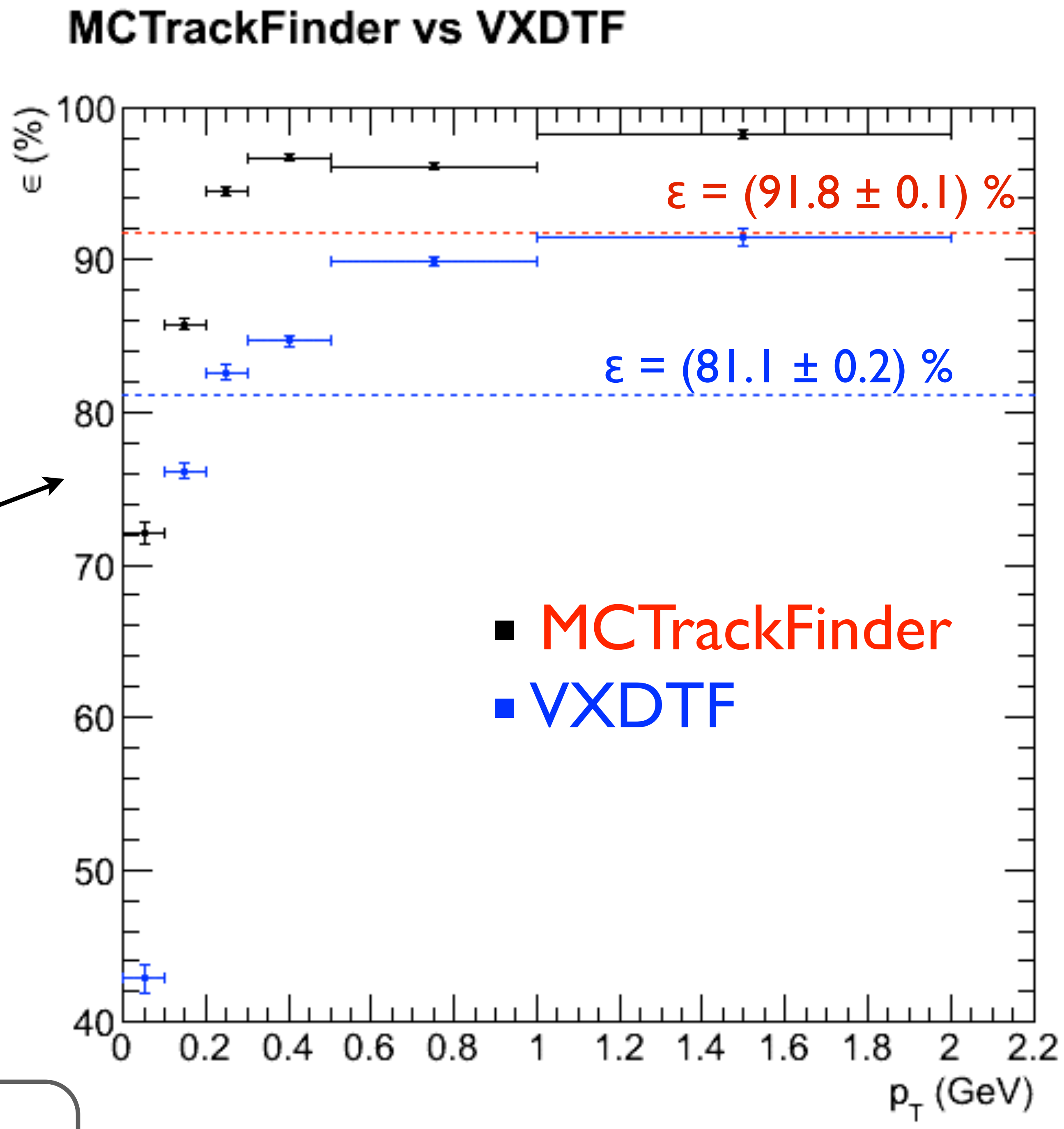
- Execution time (code compiled in *debug* mode):
  - ★ 45.5 ms/track with 5 iterations of the kalman filter
  - ★ 53 ms/track with 10 iterations of the kalman filter
    - ★ opt mode: 21 ms/track with 10 iterations
  - ★ 68 ms/track with 20 iterations of the kalman filter

MCTrackFinder vs VXDTF

track candidates are built with the true SVT hits

track candidates produced by real pattern recognition

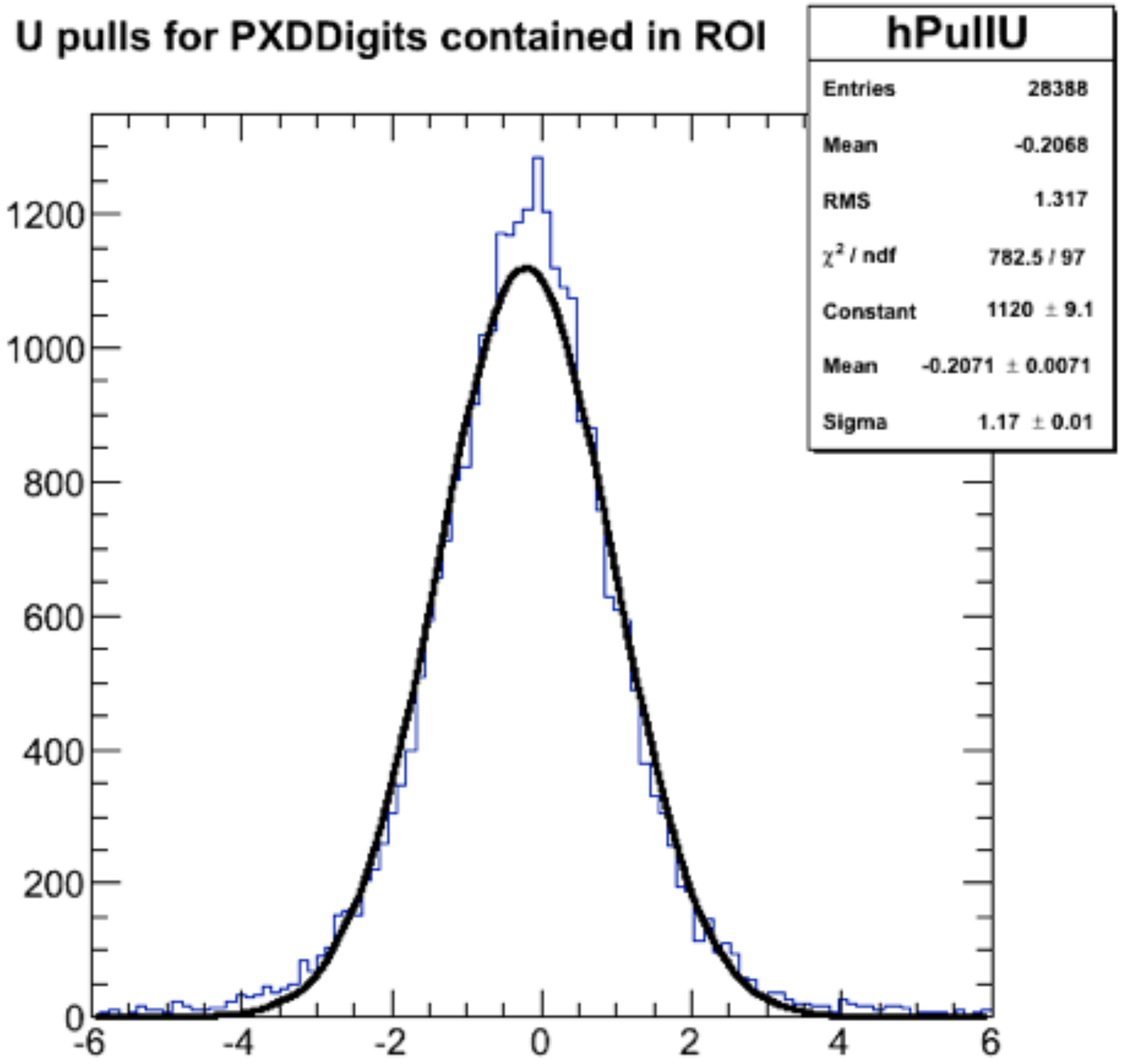
- We simulate 1k events using EvtGen and use the *VXDTF* as *pattern recognition*:
  - ~8.5 tracks/event
  - ~4.1 PXDDigits/track
- Efficiency = (81.1±0.2)% = 28388/34988 PXDDigits
  - similar dependence on transverse momentum observed with MCTrackFinder
- Inefficiency mostly due to failures in fitting the track and finding an intercept with the sensor planes
  - similar behavior observed with MCTrackFinder

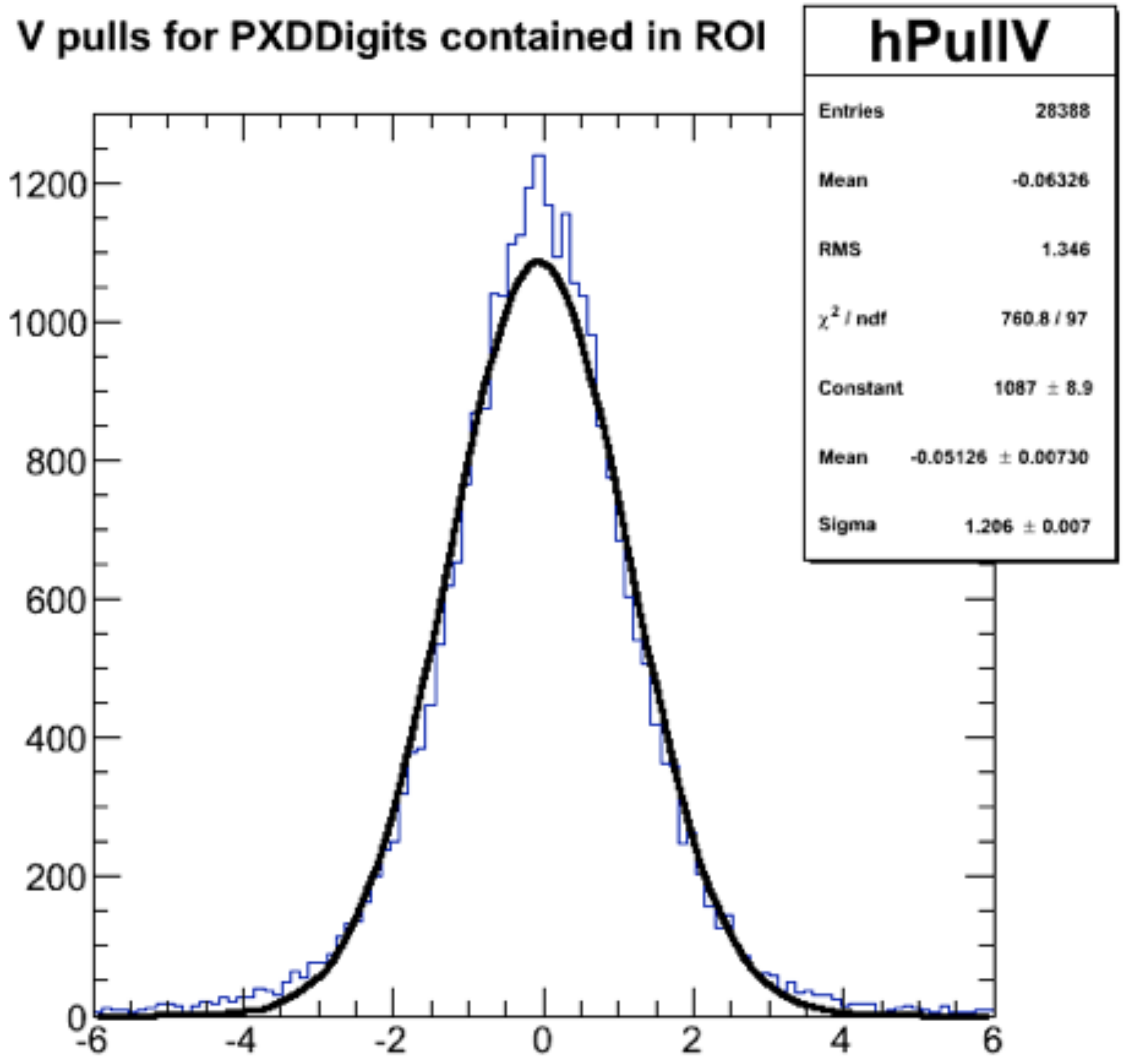


$$\varepsilon = \frac{\text{\# PXDDigits inside a ROI}}{\text{total \# PXDDigits of GFTrackCand}}$$

inefficiencies of the pattern recognition are factorized!!

ROIs with the VXDTF





➡ U (V) Pulls are negatively biased by 20% (5%) of the statistical error

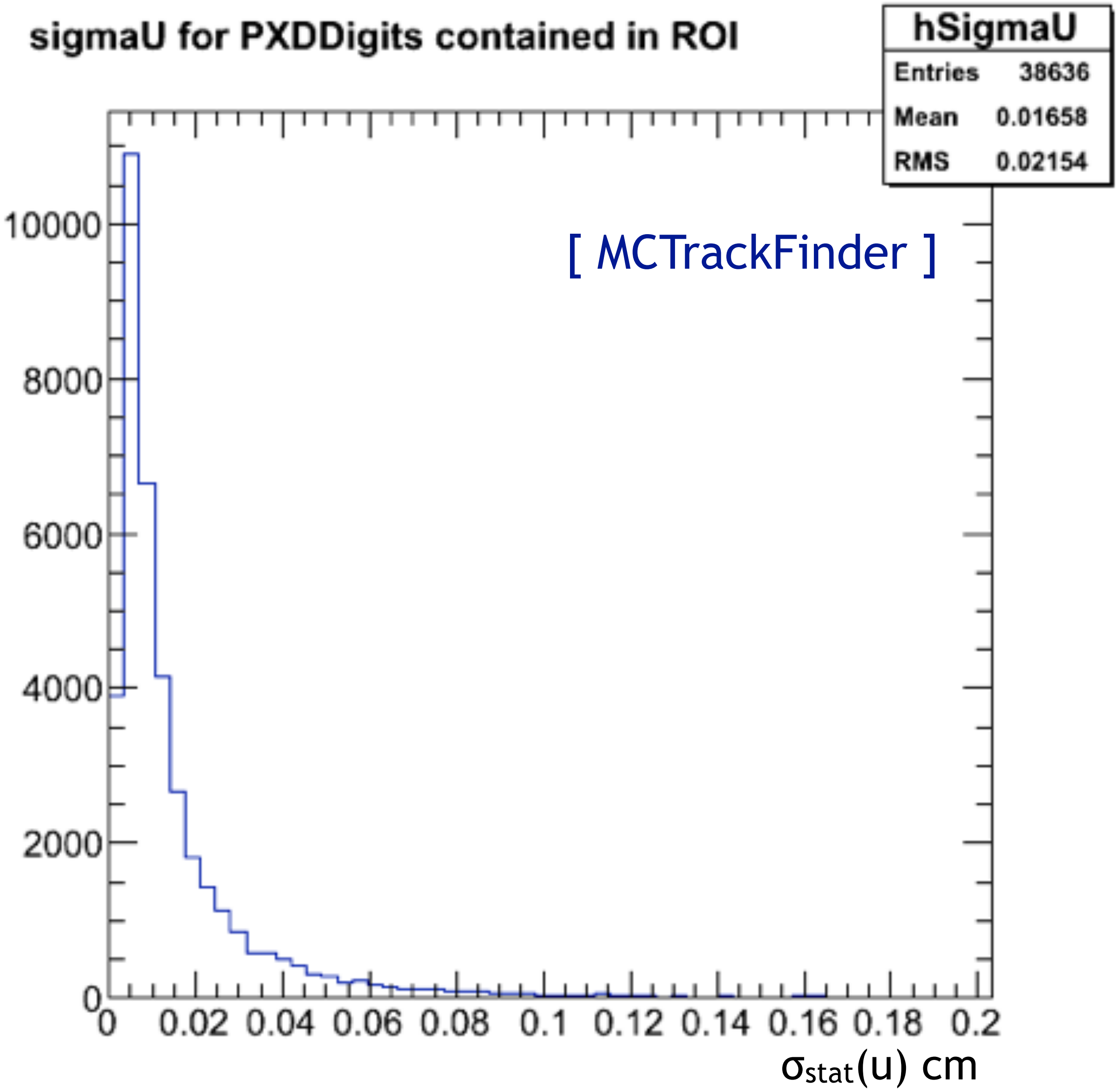
➡ the statistical errors are underestimated by ~10-15%

➡ Data Reduction Factor = 0.8%

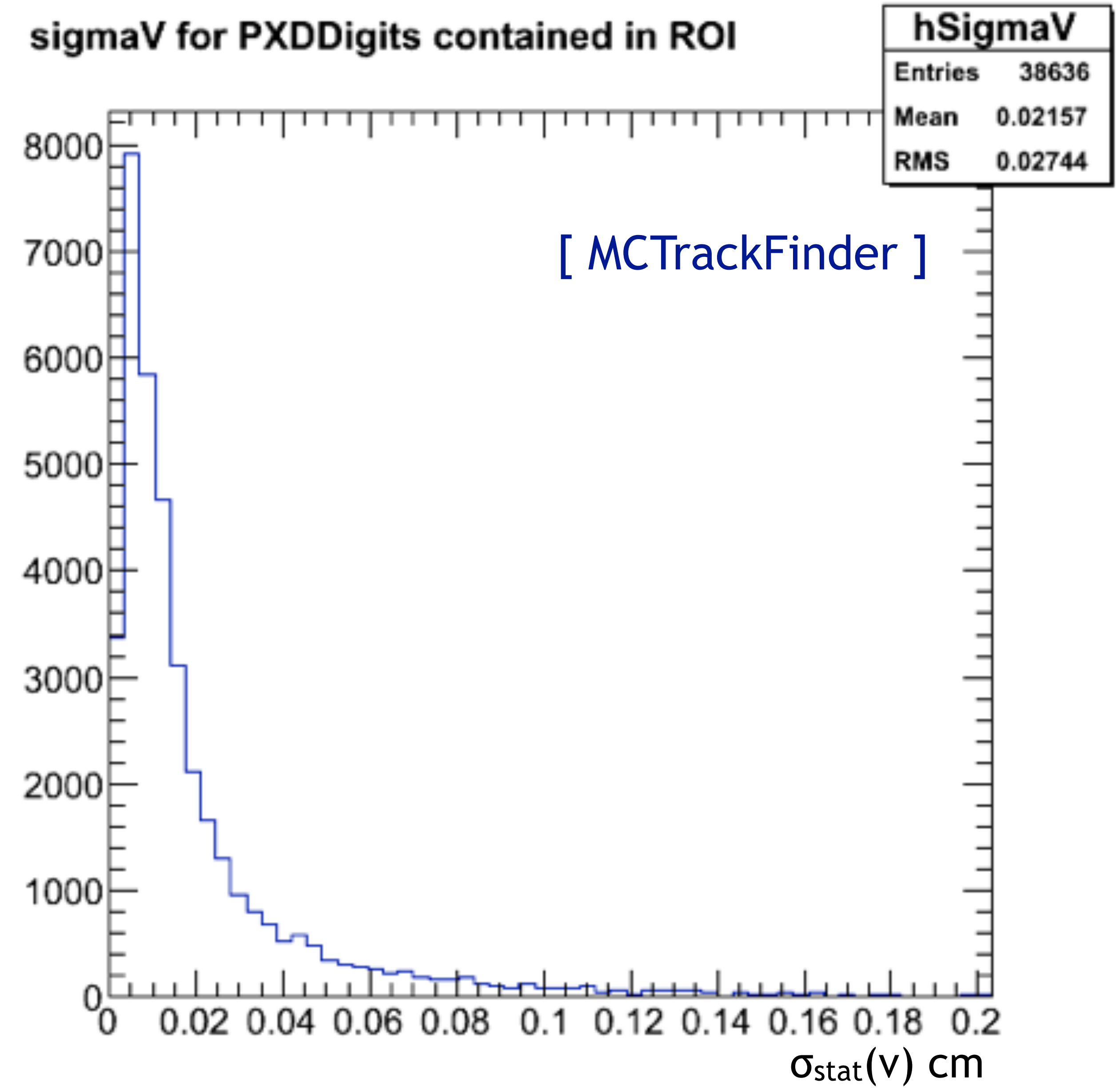
➡ Execution time = 35 ms/track (10 iterations of the kalman filter)

[genfit1]

Statistical Error of the Extrapolation



[MCTrackFinder]



[MCTrackFinder]

➡ with the VXDTF we observe similar statistical errors :

- in the U direction: mean = 0.015 cm (-10%), RMS = 0.021 cm (+13%)
- in the V direction: mean = 0.020 cm (-7%), RMS = 0.031 cm (-5%)