EUTelescope (for beam tests with Mimosa26 sensors)

final status

Igor Rubinskiy

+

Joerg Behr, Antonio Bulgehroni, Ingrid Gregor,
Tatjana Klimkovich, Slava Libov,
Philipp Roloff, Filip Zarnecki,
and the Telescope users (APIX PPS+3D, ALFA, DEPFET)

EUDET-JRA1 annual meeting
DESY/Hamburg

29 September 2010

Contents

EUTelecope releases

- EUTelescope releases
  - [Pro] Version v00-04-01
    - 1 year ago - [Old] v00-02-00, a.k.a. Better User Integration (BUI)
      - » Python based submission scripts introduced
    - 8 intermediate releases in this year
    - The EUTelescope analysis framework
      - » mostly final,
      - » but there are still things to add
    - significant performance improvements
      - » CPU
      - » Memory
      - » human intervention reduced to minimum (almost none)
    - documentation is kept up-to-date with every release
      - » How to run the EUTelescope step by step with python scripts: http://projects.hepforge.org/eudaq/Eutelescope/pythonScripts.html
      - » It is as easy to run analysis on GRID: http://projects.hepforge.org/eudaq/Eutelescope/gridtools.html

EUTelecope installation

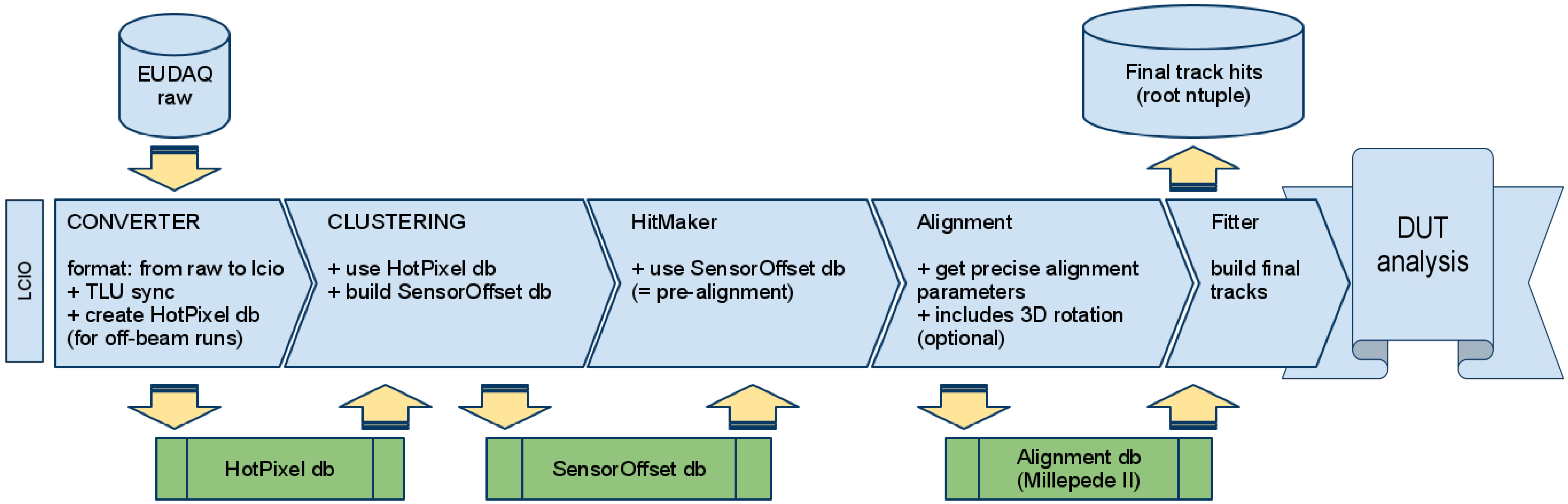
- Step by step (copy-paste style) instructions at http://projects.hepforge.org/eudaq/Eutelescope/ilcinstall.html
- installation on SL4/SL5 goes without problem (other OS problematic, e.d. Ubuntu)
- Can be a bit tricky
  - Due to mutual EUDAQ-EUTelescope dependencies
  - Follow strictly the instructions
    - Install EUDAQ
      - Compile with LCIO=0 and EUTELESCOPE=0 (Makefile flags)
    - Install full ilcsoft
      - With EUTelescope depending on the EUDAQ library
      - Recompile EUDAQ with LCIO=1 and EUTELESCOPE=1
      - Recompile the EUTelescope against the new libeudaq.so
    - Install Millepede II (the latest one from the svn)
- Before the EUTELESCOPE analysis can be started (every new terminal session) the environment must be loaded
  - %> source $EUTELESCOPE/build_env.csh
- Now can analyse beam test data
  - Go from pixels in RAW (EUDAQ format) to track parameters in LCIO (or ROOT)
  - And analyse the DUT features (see talk by Slava Libov)

Telescope users in 2010

user	data, GB	# runs	# DUTs	location	# events
FORTIS+TPAC	936	1523	1	DESY	~90 mln
TIMEPIX (INGRIDs)	-	-	-	“	-
APIX (3D)/RD42(SPIDER)	534	942	1/1'	CERN	~60 mln
NA62	15	288	0	“	~15 mln
APIX (Diamond)	20	221	1-2'	“	~10 mln
APIX (PPS)	85	908	8	“	~30 mln
ALFA	98	532	0	“	~98 mln
SPIDER	8	72	-	“	~7 mln
SILC	...	...	...	...	...
DEPFET	...	...	...	...	...

Total:	~300 mln
Analysis speed:(M26x6)	~10-50 ms/ev
(raw->Tracks ntuple)	[prev. ~1 s/evt]

EUTelescope data flow concept (for Telescope with Mimosa 26)



PyConverter
(python script processing of the processors in the template converter-tmp.xml)

- few changes
  - Producer based TLU id synchronization improved (Emlyn Corrin)
  - add one more processor – HotPixelKiller [to be run only on Off-Beam runs!]
    - **Define a “hot” pixel as firing more frequent then 1% of time** per run without beam, it looks like 10K events is enough. Default numbers: 1% and 10K events can be changed via steering template.
    - **Dump the HotPixel Collection into a DB file** with structure identical to a normal data run
    - The HotPixel Collection can be **loaded by the Clustering processor** and the **hot pixels skipped** during clustering
    - It's a good idea to take medium size runs (100K) without beam once in a while to see how the hot pixel distribution changes (if at all).

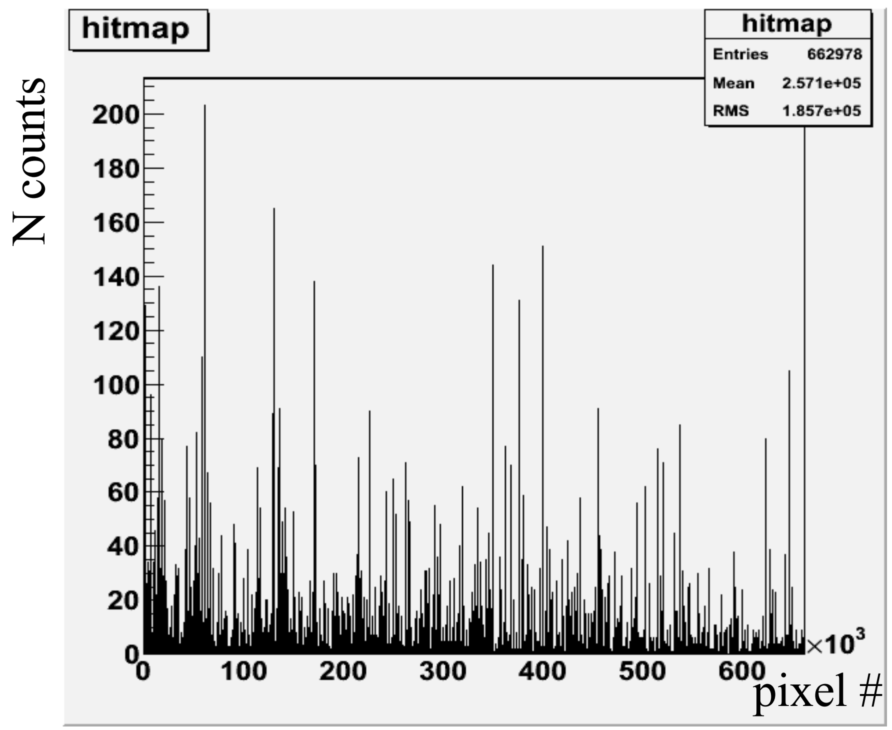
PyConverter
(python script processing of the processors in the template converter-tmp.xml)

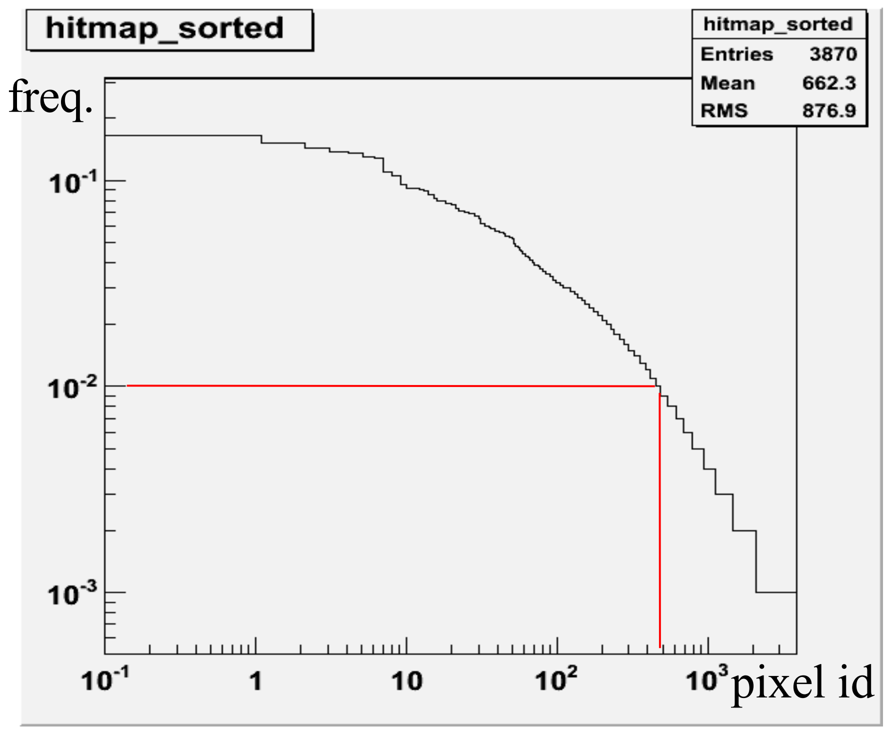
```
<execute>
   <processor name="UniversalNativeReader"/>
   <processor name="Mimosa26EUTelAutoPedestalNoiseProcessor"/>
   <processor name="HotPixelKiller"/>
</execute>

<processor name="UniversalNativeReader" type="EUTelNativeReader">
       <!--Resynchronize the events based on the TLU trigger ID-->
       <parameter name="SyncTriggerID" type="bool" value="false"/>
</processor>

<processor name="HotPixelKiller" type="EUTelHotPixelKiller">
      <parameter name="MaxAllowedFiringFreq" type="float" value="0.01"/>
      <parameter name="NoOfEventPerCycle" type="int" value="10000"/>
      <parameter name=”TotalNoOfCycle” type=”int” value=”0”/>
</processor>
```

new

new

PyConverter :: Hot Pixel Killer





- a run without beam
- ordered pixels
  - 3870 total, spikes are the “hot pixels”
  - define “hot pixels” as firing more often then 1% (600 pixels, 16% of all pixels)

PyClustering
(python script processing of the processors in the template clusearch-tmp.xml)

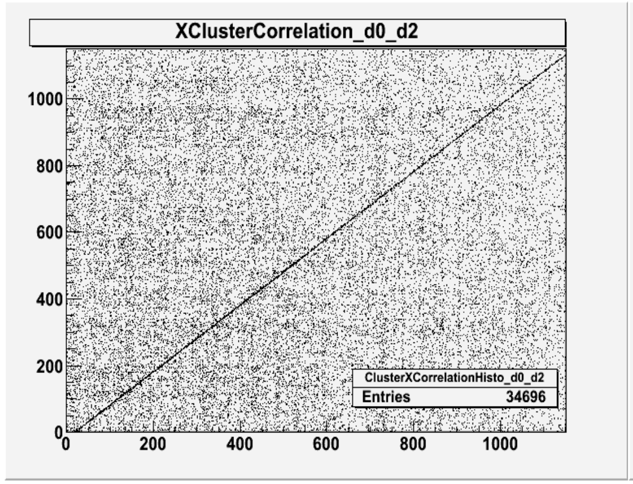
```
     <execute>
         <processor name="AIDA"/>
           <processor name="Mimosa26EUTelAutoPedestalNoiseProcessor"/>
new |      <processor name="LoadHotPixelDB"/>
           <processor name="Clustering"/>
           <processor name="Correlator"/>
         <processor name="Save"/>
       </execute>


    </processor>
         <parameter name="ZSClusteringAlgo" type="string" value="DFixedFrame"/>
         <!--parameter name="ZSClusteringAlgo" type="string" value="SparseCluster2"/-->
    </processor>
```

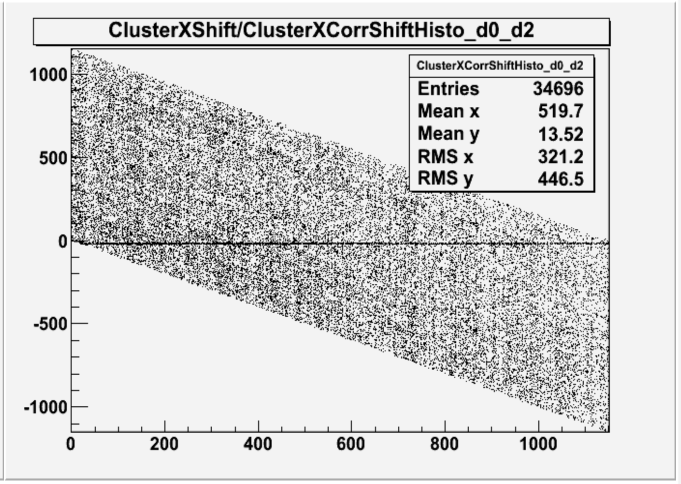
PyClustering:: Clustering

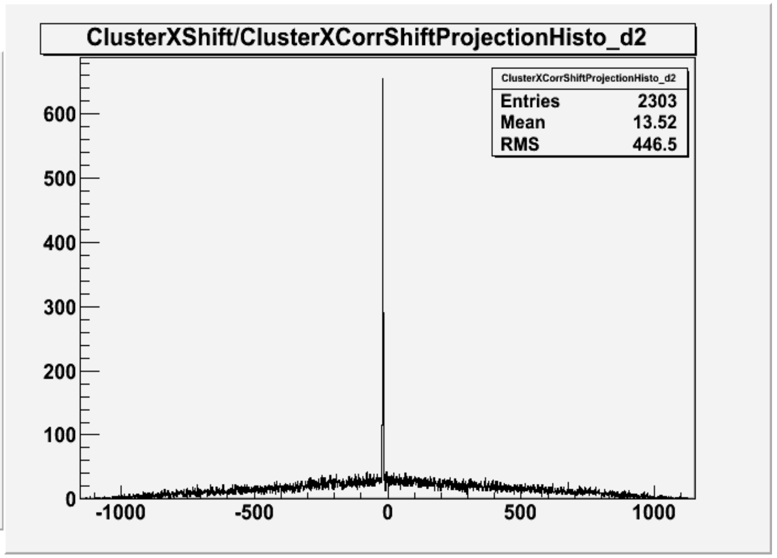
- Optimised clustering algorithms:
  - Digital Fixed Frame (DFF) – a version of the Zero Suppressed Fixed Frame (FF) [J.Behr]
    - Looking for clusters NxM size
    - The slowest part was the status collection update (to keep track of the hot pixels > 650K)
  - Sparse Clustering 2 (SP2) - a version of the Sparse Clustering (SP) [A.Bulgheroni]
  - In both cases the event-based-HotPixelKiller by default is turned off
    - The hot pixel collection is read only once in the beginning of the run
    - If in the real (beam) data run a pixel is found which is also in the HotPixel DB file, it's being ignored
      - » for hotpixel frequency 0.01 the cluster rate goes down by 50% (speedup ~ x2)
  - Some tests on one run (10602) of the clustering performance (first 10K events):
    - HotPixel freq 0.01:
      - » DFF(3x3): 238K (sensor 0), timing (for all): 31 ms/evt.
      - » SP2: 228K (sensor 0), timing (for all): 18 ms/evt.
    - SP2:
      - » HotPixel freq -0.05: 307/23, -0.10: 334/25, -0.15: 342/26 [K cluster/ ms/evt]
    - DFF(3x3):
      - » HotPixel freq -0.05: 317/36, -0.10: 344/28, -0.15: 352/39 [K cluster/ ms/evt]

PyClustering:: **Correlator**

- Correlator processor
  - Build 2D correlations between sensors (the first one #0 and all others)
    - sensor0(X).vs.sensor1(X), sensor0(Y).vs.sensor1(Y), etc.
    - If there was beam, one must see a clear diagonal line (“correlation band”)
    - By the shift of the “correlation band” from the real diagonal (from 0:0 to 1156:1156) one can calculate the relative sensor shift -> preAlignment
  - Build 2D “biased” correlation plot
    - sensor0(X).vs.[sensor1(X)-sensor0(X)], etc. [must take into account sensor flip!]
    - In this case the “correlation band“ goes horizontal
    - Make a 1D projection and the peak position gives the sensor offset value
    - Dump them into a db file with structure identical to the existing alignment-constants db file
    - Apply this preAlignment constants already at the Hitmaker level





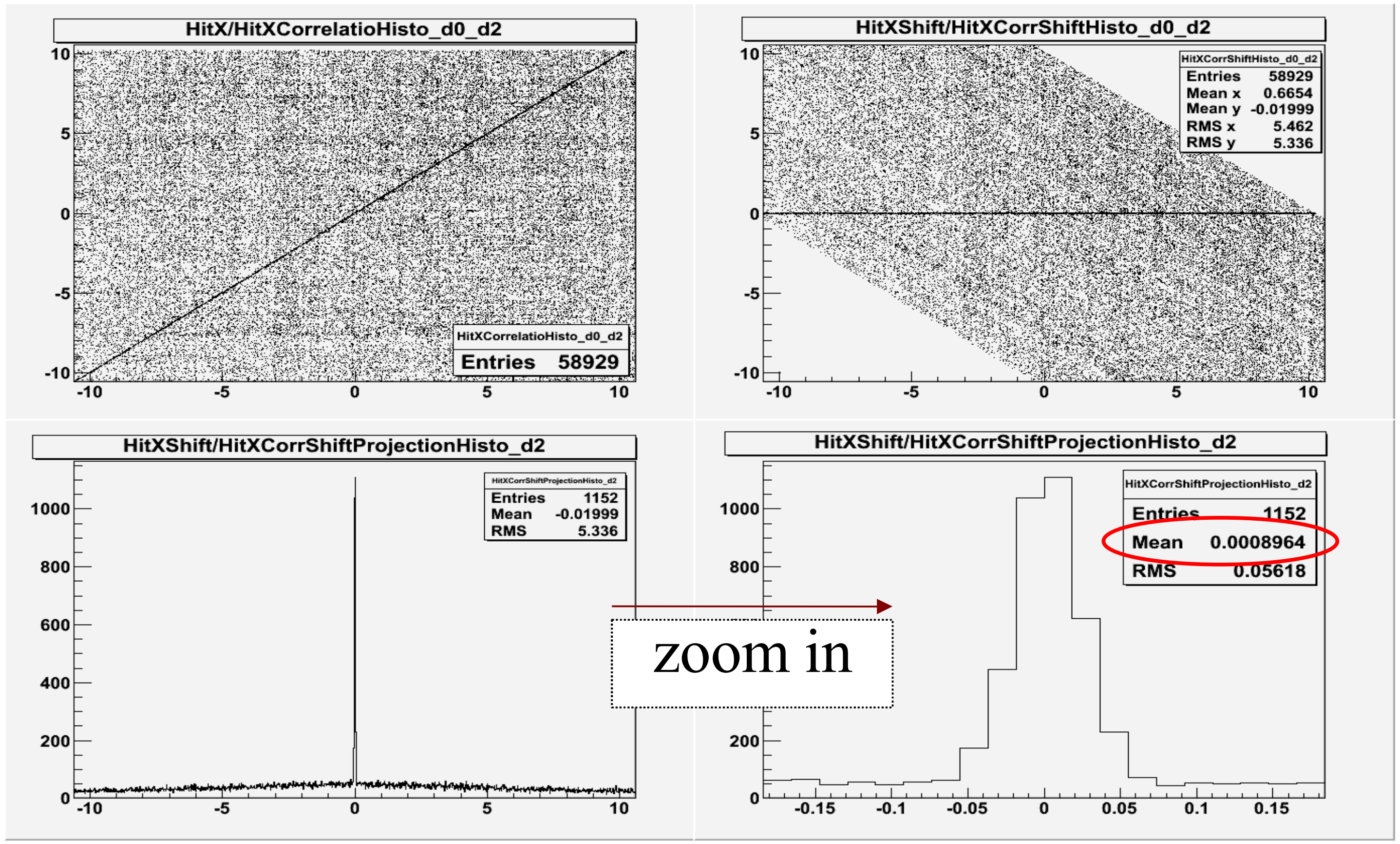


PyHitmaker
(python script processing of the processors in the template hitmaker-tmp.xml)

```
<execute>
     <processor name="AIDA"/>
     <processor name="LoadOffsetDB"/>
     <processor name="HitMaker"/>
     <processor name="Correlator"/>
     <processor name="Save"/>
</execute>

<processor name="LoadOffsetDB" type="ConditionsProcessor">
   <parameter name="SimpleFileHandler" type="StringVec"> preAlignment offset-db.slcio preAlignment    </parameter>
</processor>

<processor name="HitMaker" type="EUTelHitMaker">
   <parameter name="CoGAlgorithm" type="string" value="FULL"/>
</processor>
```

PyHitmaker :: HitMaker + Correlator

- main improvement – preAlignment!
- without preAlignment (previously) the sensors relative offset could be 0.1-2 mm, which make the real alignment (Millepede II) a bit problematic
- Now all sensors are preAligned at 0, with precision of better then pixel (half) pitch



PyAlignment
(python script processing of the processors in the template align-tmp.xml)

```
<execute>
        <processor name="AIDA"/>
        <processor name="Align"/>
</execute>
```

- The sensor X/Y offset values are calculated and applied in the PyHitmaker step, so we can **set new default Residual cut values** for the PyAlign step in the config/config.cfg file.
- This step required a lot of manual work previously, now the demand of babysitting significantly reduced

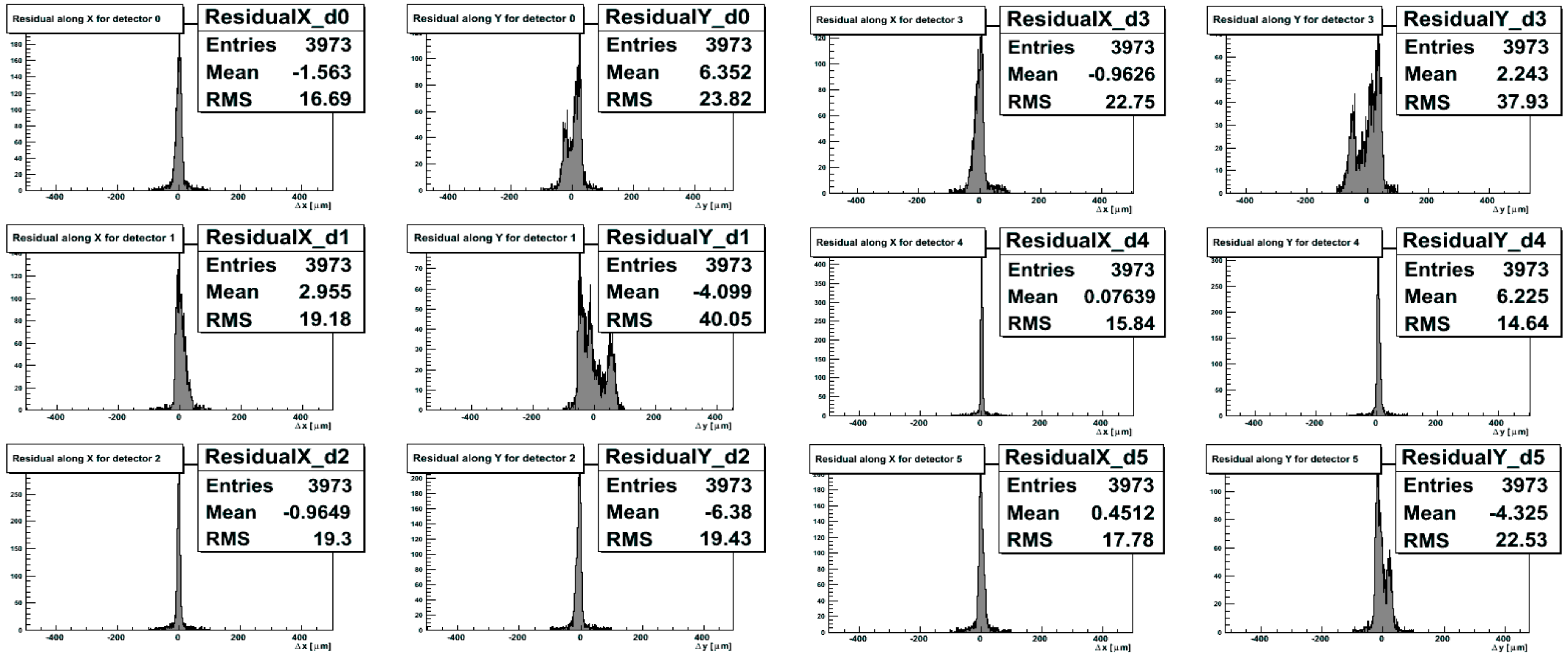
```
[AlignOptions]

    ResidualXMin      = -100  -100  -100  -100  -100  -100
    ResidualXMax      =  100   100   100   100   100   100
    ResidualYMin      = -100  -100  -100  -100  -100  -100
    ResidualYMax      =  100   100   100   100   100   100

    DistanceMax       =  2000
```

PyAlignment – Millepede II data format change

- The Millepede II output constants file format changed about a year ago
- For the time being all EUTelescope users were forced to use a modified version (by Joerg)
- Since v00-02-02 the EUTelescope format has been amended to the new official Millepede II
- The latest official Millepede II can be used now

PyAlignment – residual plots

- Main goal achieved – sensors initially aligned around '0'
- now at the final (fine) alignment (Millepede II) we can do um level alignment
- X direction is fine
- Y direction not always fine – artefact (previously seen in other runs)

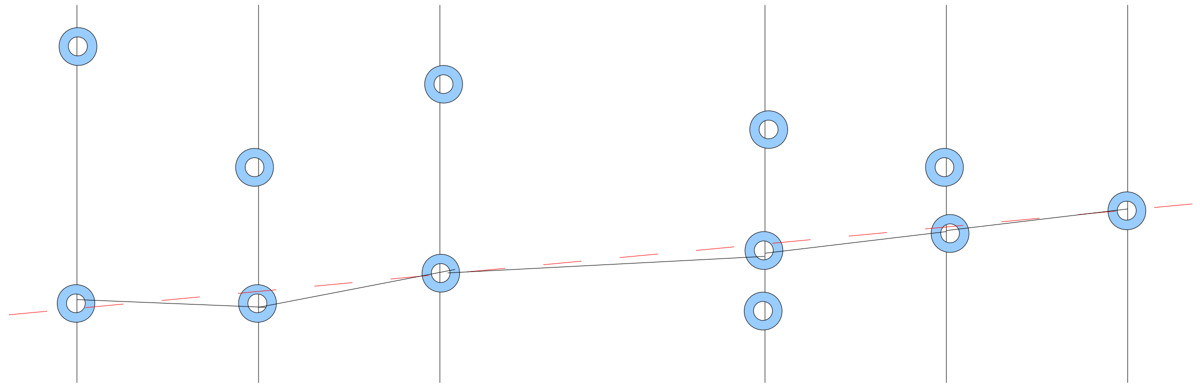


PyFitter
(python script processing of the processors in the template fitter-tmp.xml)

```
<execute>
          <processor name="LoadAlignment"/>
          <processor name="ApplyAlignment"/>
          <processor name="Fitter"/>
</execute>


<processor name="Fitter" type="EUTelTestFitter">
 <!--Decide now weather you want to rely on the track candidate slope permanence in X and Y, default=true -->
      <parameter name="UseSlope"   type="bool"  value="true"/>
      <!--Set the allowed maximum difference of the slope in X (from plane to plane), default = 0.01 -->
      <parameter name="SlopeXLimit"   type="float" value="0.0001"/>
      <!--Set the allowed maximum difference of the slope in Y (from plane to plane), default = 0.01 -->
      <parameter name="SlopeYLimit"   type="float" value="0.0001"/>
      <!--Maximal allowed (initial) distance between hits in the XY plane between the planes,default = 2. mm -->
      <parameter name="SlopeDistanceMax" type="float" value="@DistanceMax@"/>
</processor>
```

Most recent changes to the code are not discussed here
More in the talk by Filip Zarnecki

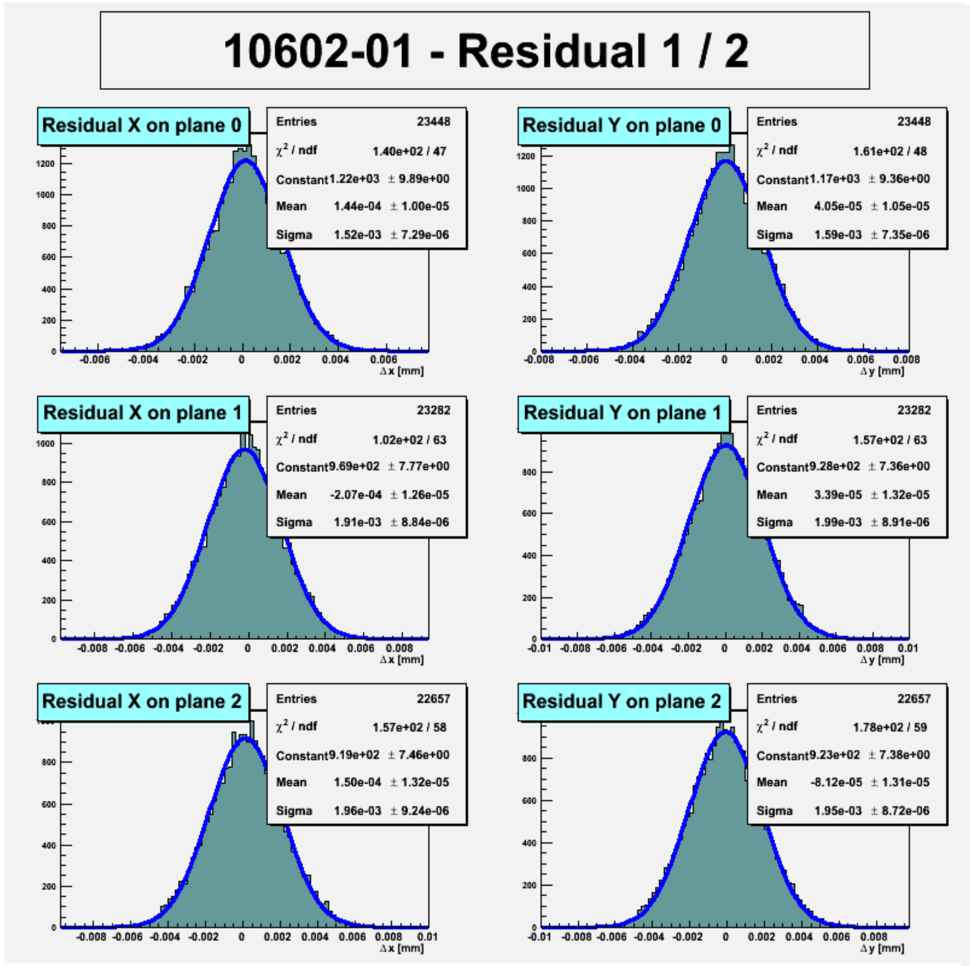
PyFitter – UseSlope (Beam slope constraint)

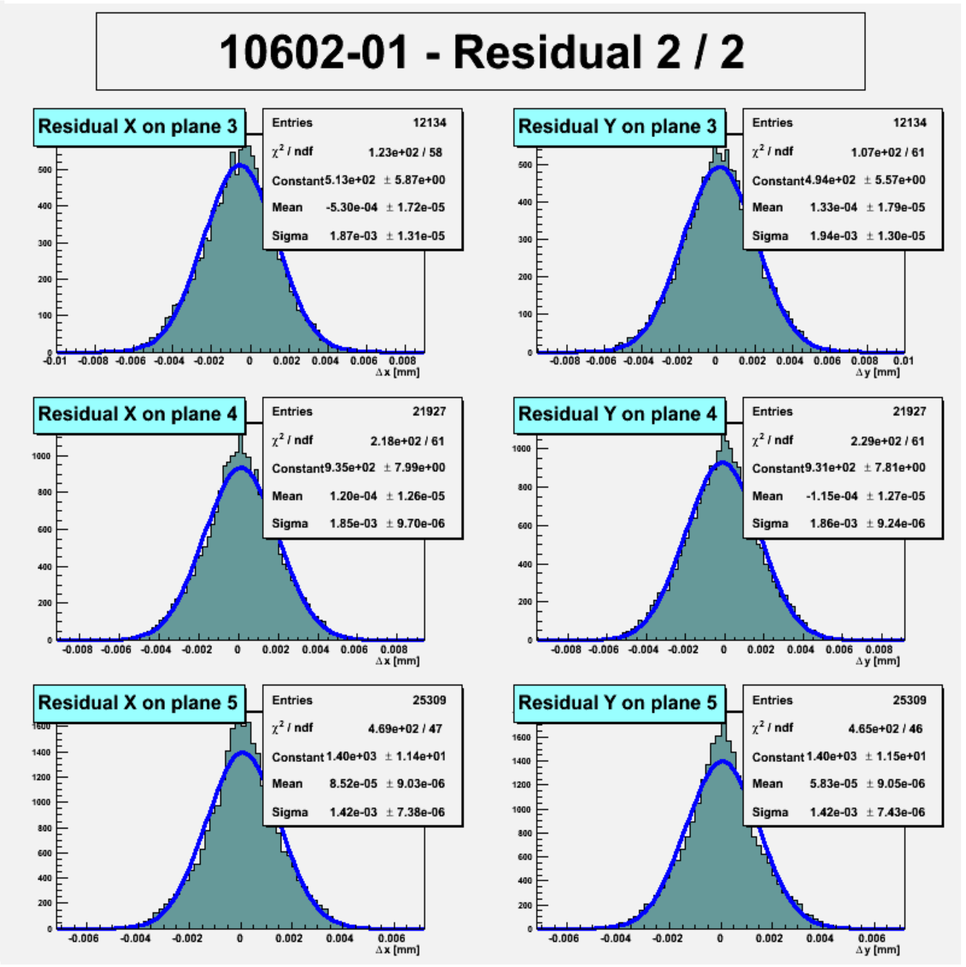


- UseSlope control card - reduces the hit selection combinatorics
- Each subsequent pair of hits is required to have only a very small difference in the slope,
    - default value is 0.01 for the (X and Y) difference of (x1-x0)/(z1-z0)
    - Since the slope has a meaning of the track angle it does not depend on the z-distance between the sensors

PyFitter – biased residuals

- Performance improvement – 1-2 order of magnitude
- Y-Artefacts from the alignment level are almost not seen (track chi2<10)





EUTelescope - Timing

- The benchamrking was done on run 10602 (July 2010 APIX data)
  - Using run 10494 without beam to prepare the HotPixel DB
  - Define a hot pixel as one with firing freq =1%
  - Only 6 Mimosa 26 sensors are considered (no DUT)
- Presently the slowest step is (still) the Clustering
  - Converter: **1 us/evt** [tot: 0.5 ms/evt, or 8 ms/evt HotPixelDB]
  - Clustering: **18 ms/evt** [tot: 20 ms/evt, excl. Correlator which runs on first 10K]
  - HitMaker: **4ms/evt** [tot: 5 ms/evt, also excl.Correlator]
  - Alignment: **1.3 ms/evt** [tot: 1.3 ms/evt]
  - Fitter: **14 ms/evt** [tot: 19 ms/evt], allowing 2 hits to be skipped (slow)
    - if none of the hits are to be skipped, the Fitter is ~x10 faster.
- Total 12 min per 10K events, but **scales not linearily with # events**

Summary

- The latest improvements to the EUTelescope library are discussed
  - Main focus in the last year developments have been given to performance improvements
  - Lots of effort put into it
  - Net estimation of the performance gain is at the level of at least factor x10 (some tests show even higher gain up to 2 orders of magnitude)
  - The slowest step is the Clustering
- The DUT analysis can be fully performed in the EUTelescope (talk by S.Libov)
- The library is in a very good shape now

- Still few things to do next:
  - Work on the sensor Geometry description in Gear to allow tilted sensors beam tests
    - 6D alignment with Millepede II have been already implemented
    - very urgent (!)
  - Need to better understand the Alignment artefacts (long standing question which seems to never have been an issue)
  - Continue on the B.U.I. (Better User Integration)
    - Can we do the installation fully automatic?