TPC space point distortion and calibration. Pass 0 and PassX calibration

marian ivanov for the TPC OFFLINE group

Outlook

TPC correction framework

Physical Models

Distortion/alignment calibration

Pass 0, Pass X calibration

TPC correction framework

(JimThomas, Stefan Rossegger, Magnus Mager, MI, ...)

Composed correction framework for modeling the TPC field distortions in AliRoot. Alice Internal Note. 2010

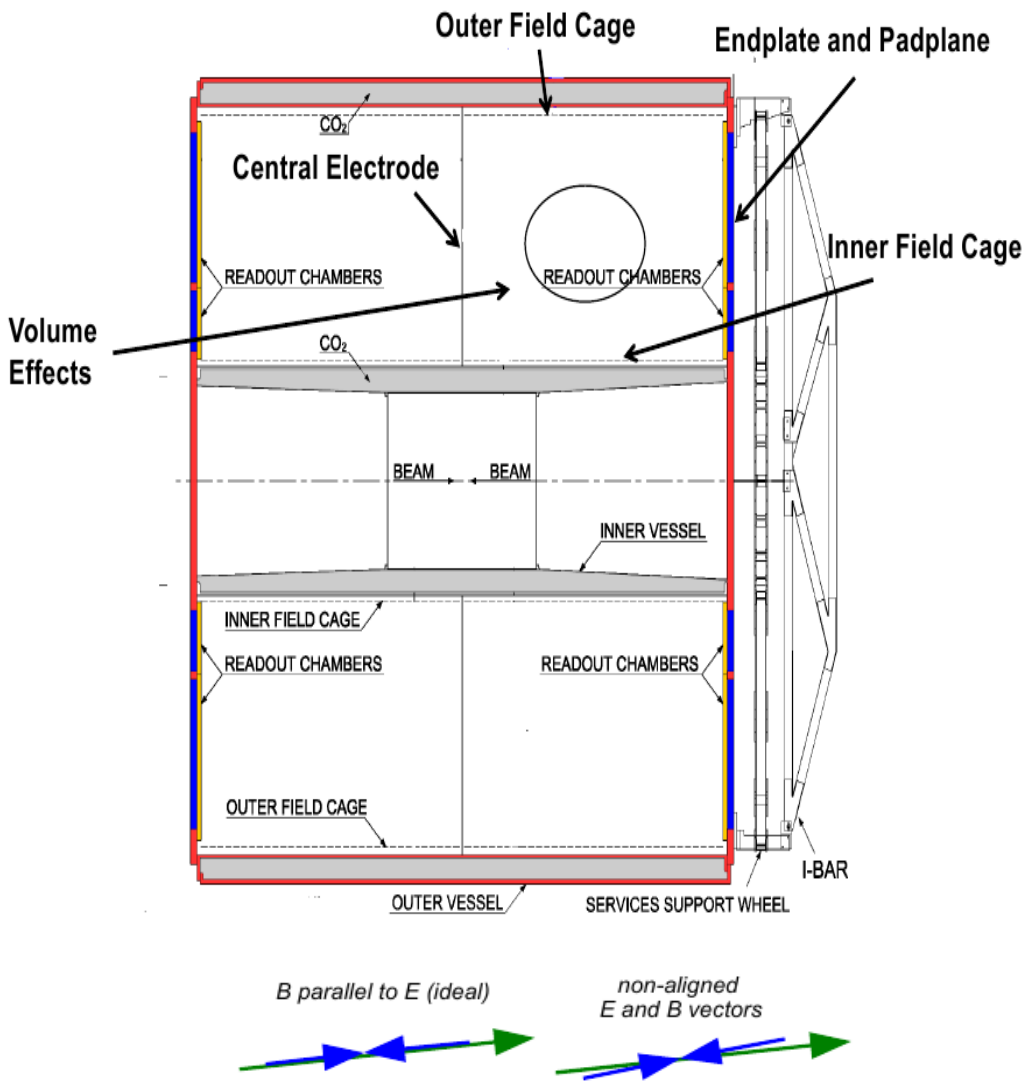
- This document provides a description of a group of C++ classes, which we have developed in order to model, distortions due to the different electric and magnetic field inhomogeneities in the ALICE TPC.
- The different models have been grouped together in a composed correction framework to simplify their usage and not to duplicate common code structures like the correction, distortion, print and plot functionality. The models were chosen to give a physical description of the distortions in order to achieve a step-by-step description of the geometrical details of the TPC itself.
- A complete list of field distortions and their resulting space point distortions can be formulated and the classes described here constitute a good description of the most important space-point distortions in the TPC.

[1] M. Mager, S. Rossegger and J. Thomas, Composed correction framework for modeling the TPC field distortions in AliRoot, Alice Internal Note, 2010, ALICE-INT-2010-018.
url: https://edms.cern.ch/file/1113105/1/ALICE-INT-2010-018.pdf

[2] S. Rossegger and J. Thomas, Space-charge effects in the ALICE TPC: a comparison between expected ALICE performance and current results from the STAR TPC, Alice Internal Note, 2010, ALICE-INT-2010-017.
url: https://edms.cern.ch/file/1113087/1/ALICE-INT-2010-017.pdf

[3] M. Mager, S. Rossegger and J. Thomas, The Langevin equation expanded to 2nd order and comments on using the equation to correct for space point distortions in a TPC, Alice Internal Note, 2010, ALICE-INT-2010-016.
url: https://edms.cern.ch/file/1108138/1/ALICE-INT-2010-016.pdf

Composed correction framework



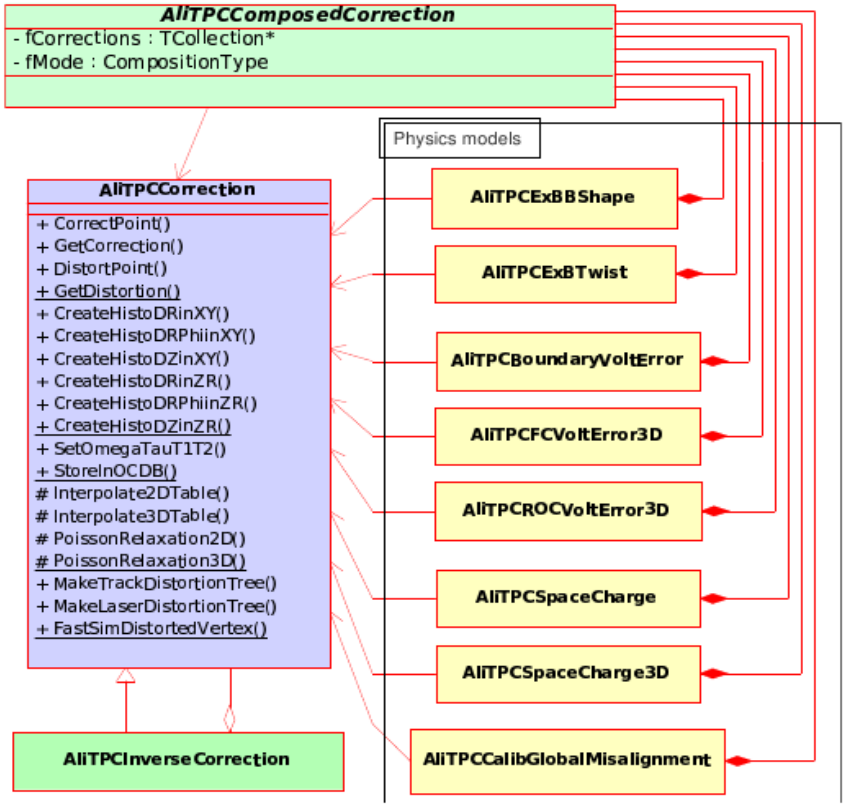
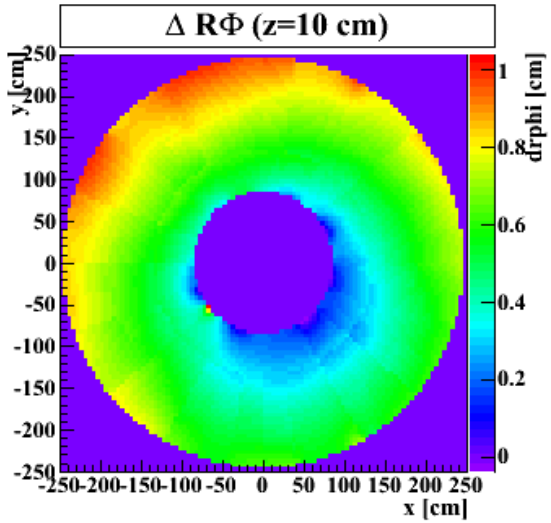
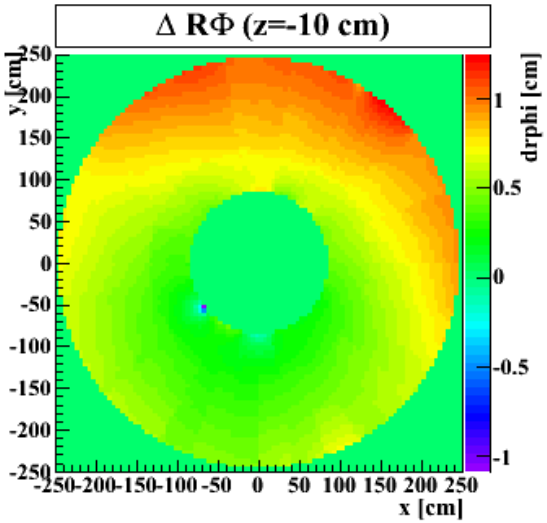
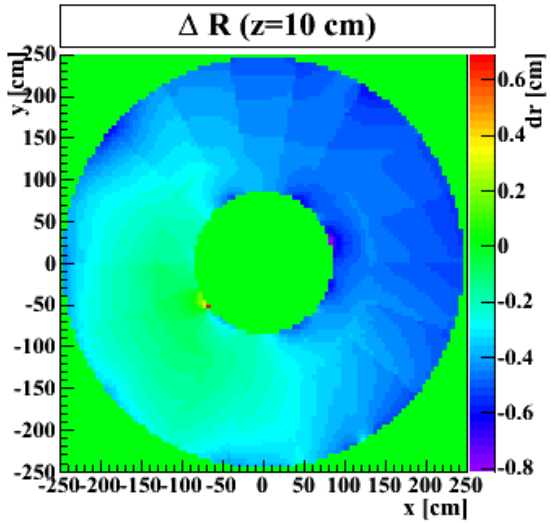


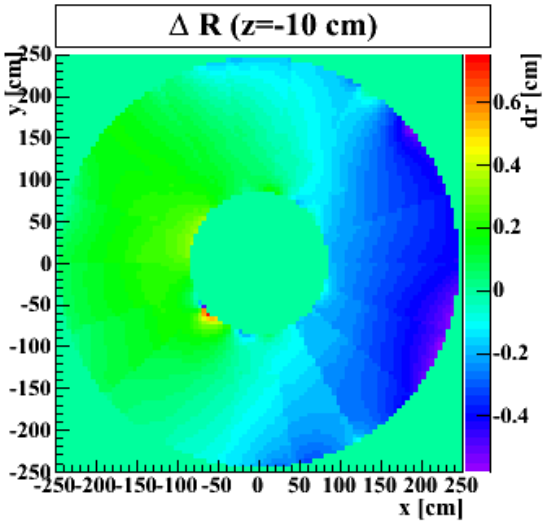
Figure 3: Simplified class diagram showing the inheritance structure and the most general functions

TPC composed correction





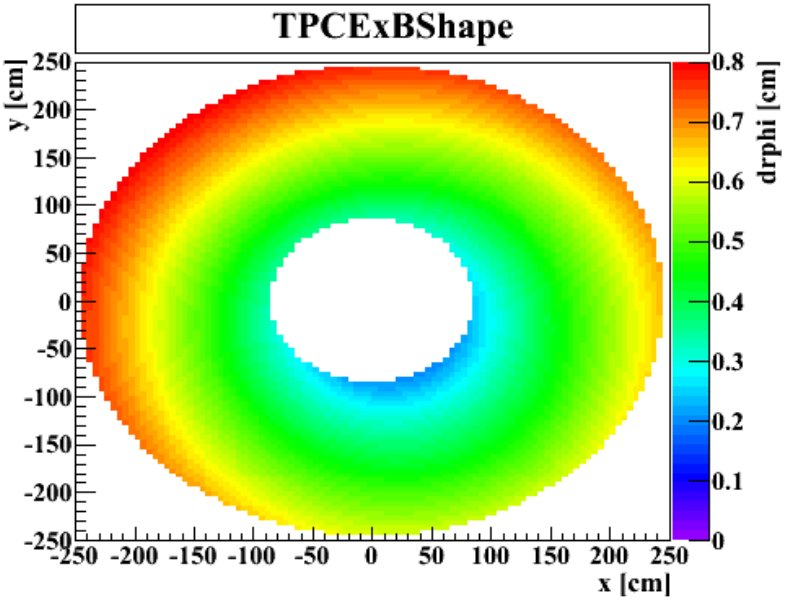


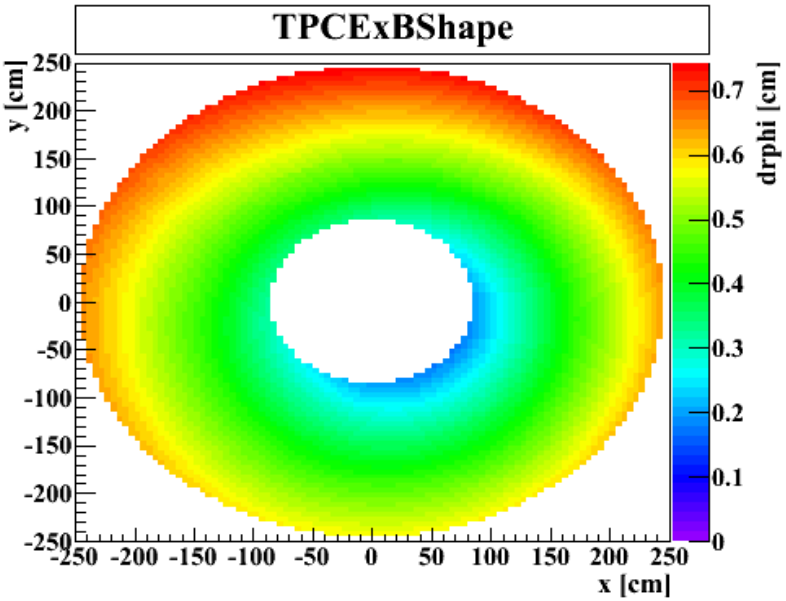


Composed of:

1. TPCExBShape
2. alignGlobal
3. alignLocal
4. alignQuadrant
5. FCVoltError3D
6. FitBoundary
7. FitExBTwist
8. FitAlignTPC
9. FitRocAlignZSum

Space point distortion due B field.





Input data – integral of B field map

Parameters to validate/fit:

- T1, T2, omega-tau
- Gas dependent

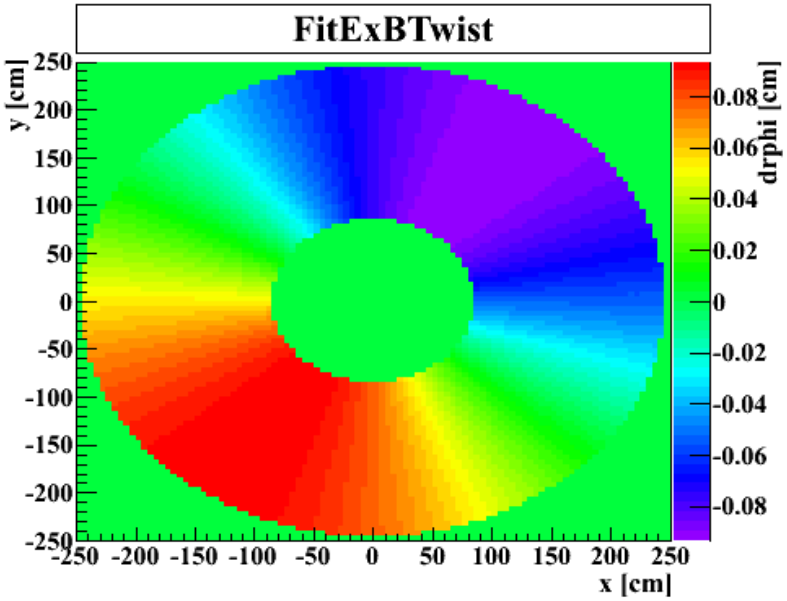
Special runs to validate models and to extract parameters:

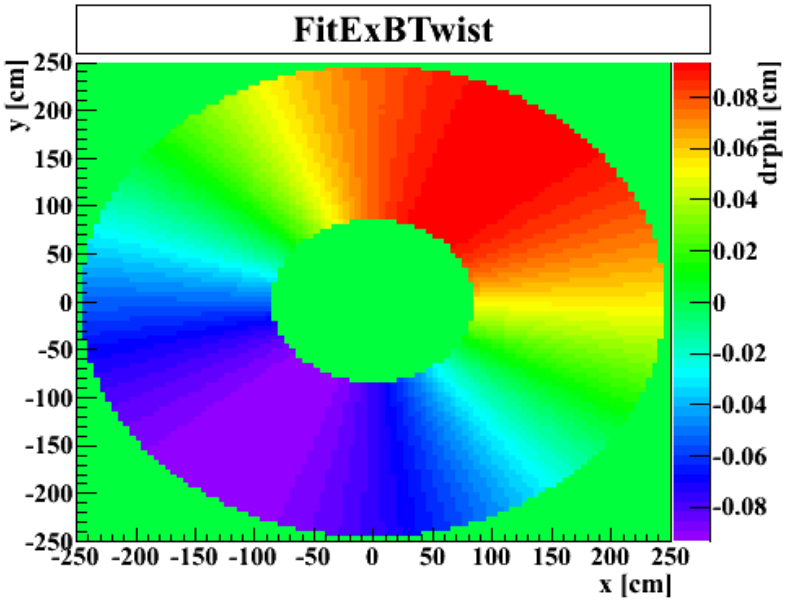
- B-field scan
- E-field scan

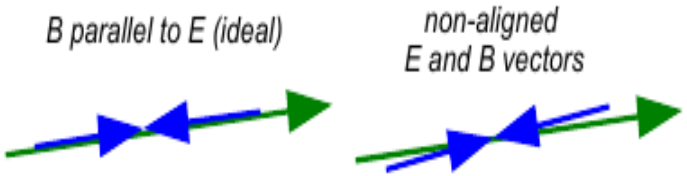
Time dependence:

- Omega-tau – depends on the time in similar way as the drift velocity

ExB twist







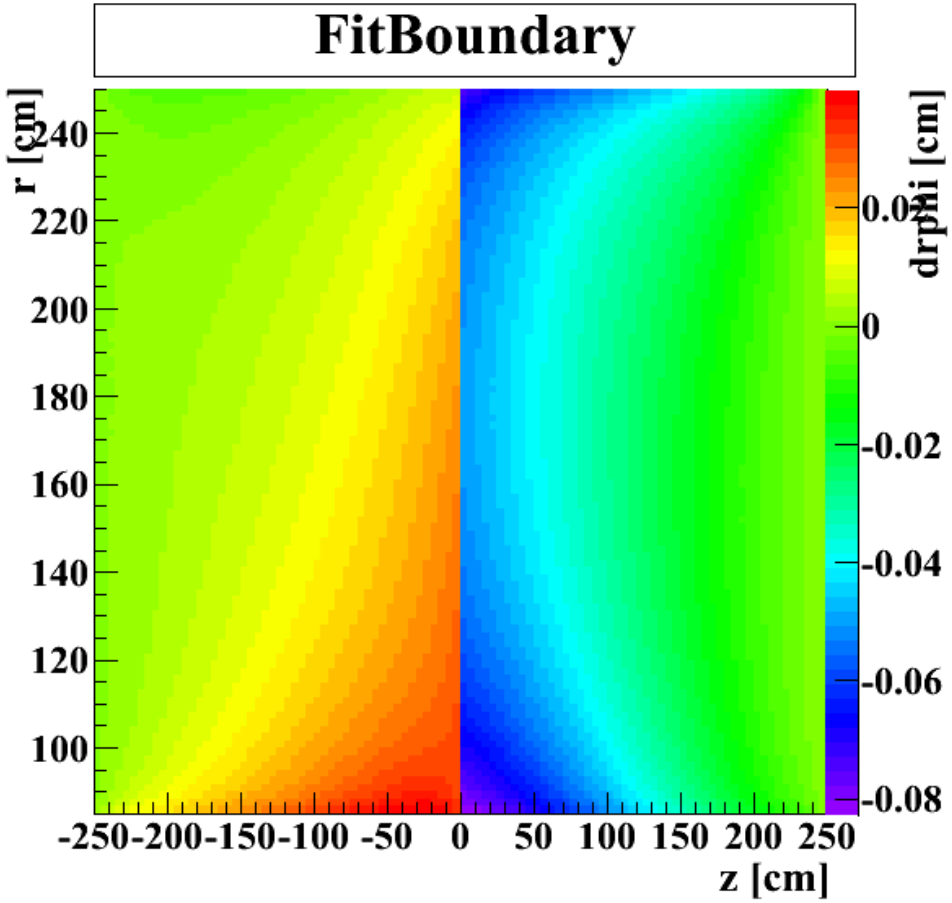
Space point distortion due global misalignment of the magnetic field and E-field in the TPC drift volume

2 parameters to fit

Time dependence:

- Following the aligment

Boundary voltage



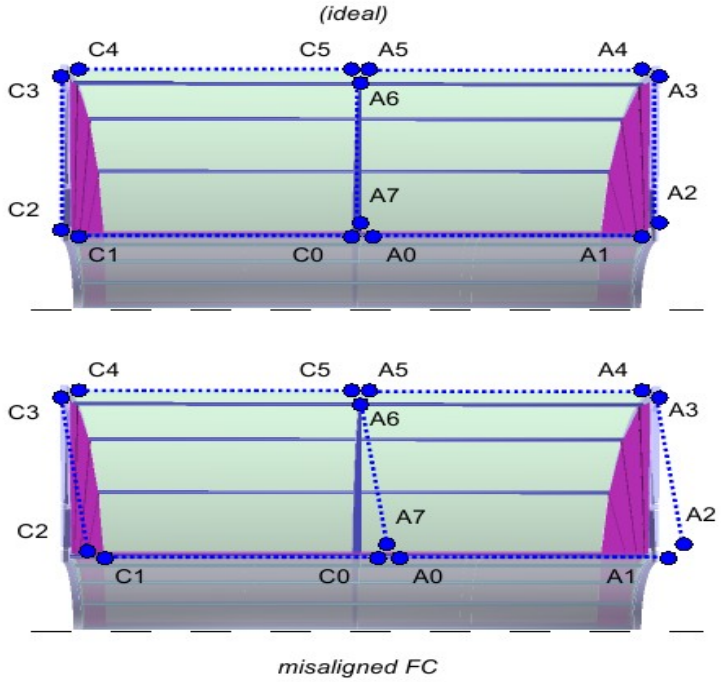
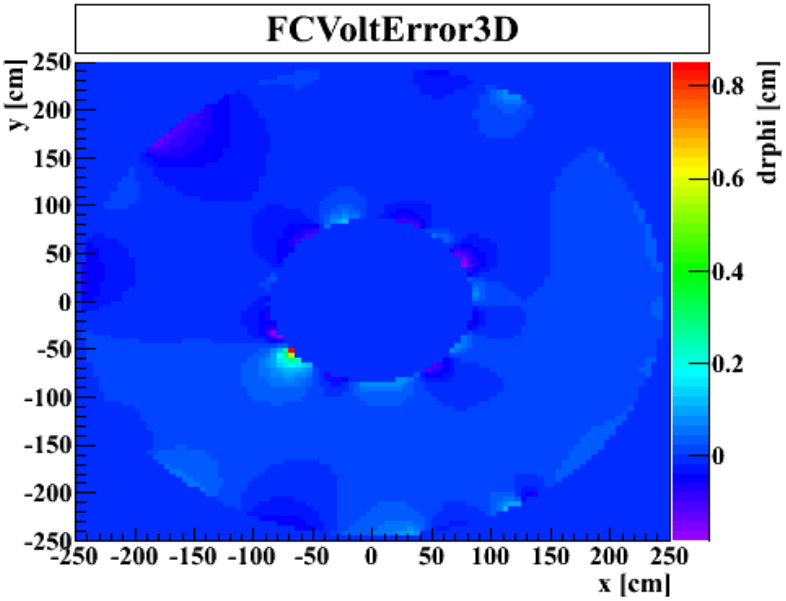
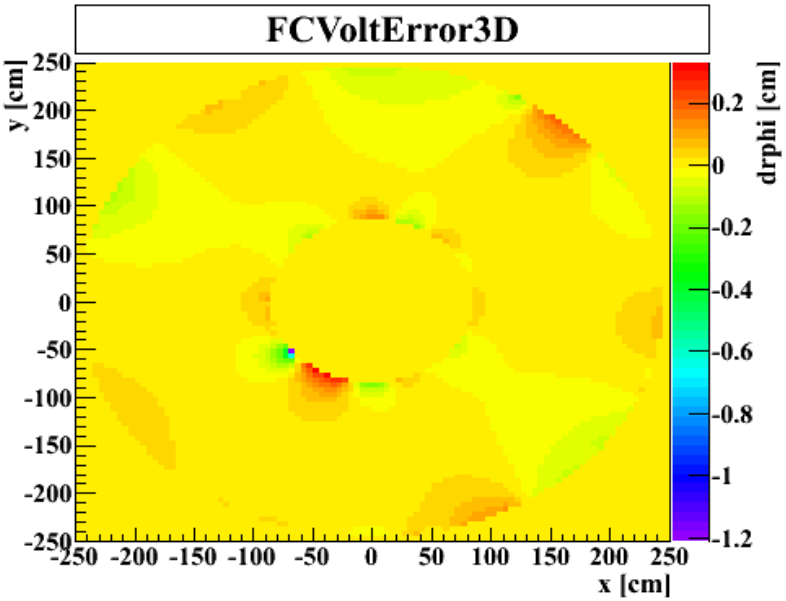


Figure 5: Numbering scheme of the vectors to set the Boundary conditions in `AliTPCBoundaryVoltError`. Top: ideal case, bottom: conical deformation of 1 mm at the IFC (e.g. $A0 = A1 = A2 = A7 = 40V$ and $C0, C1, C2 = -40V$)

Field cage and Rod alignment





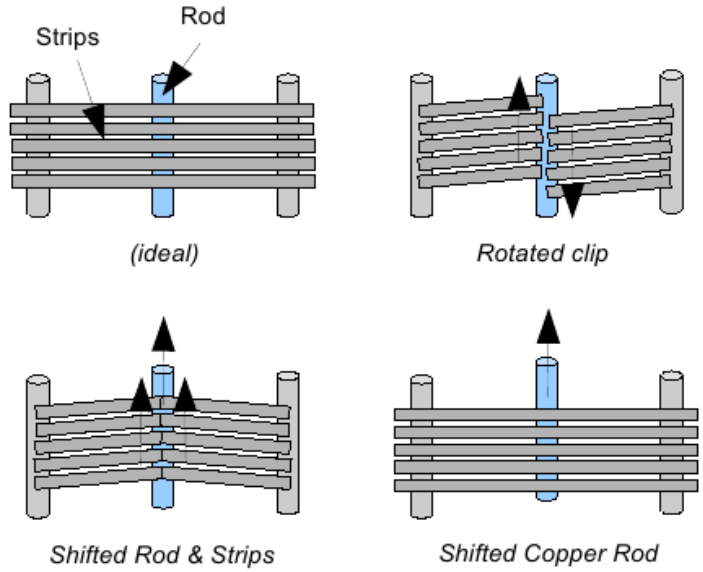


Figure 7: Misalignment scenarios of the FC components at each rod

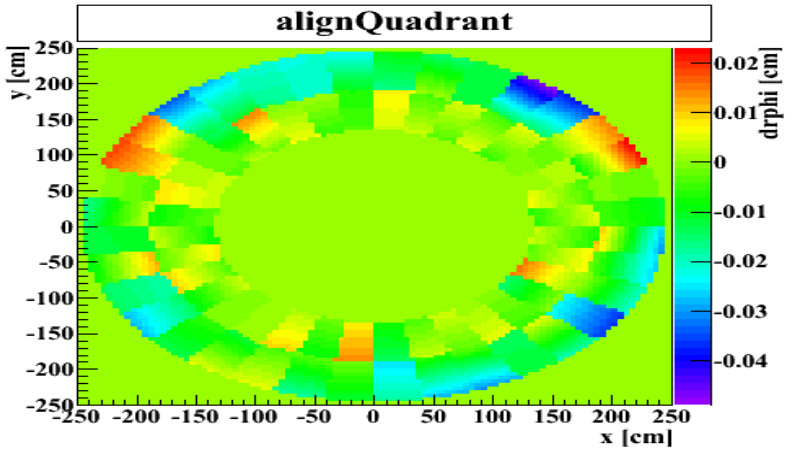
18 (rods) x 2 (IFC,OFC) x 2 (A side, C-side)

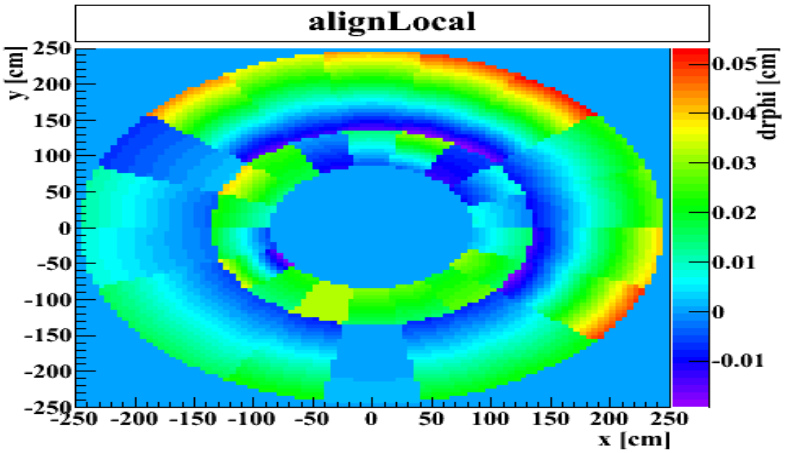
2 rotated clips x 2 (A side, C-side)

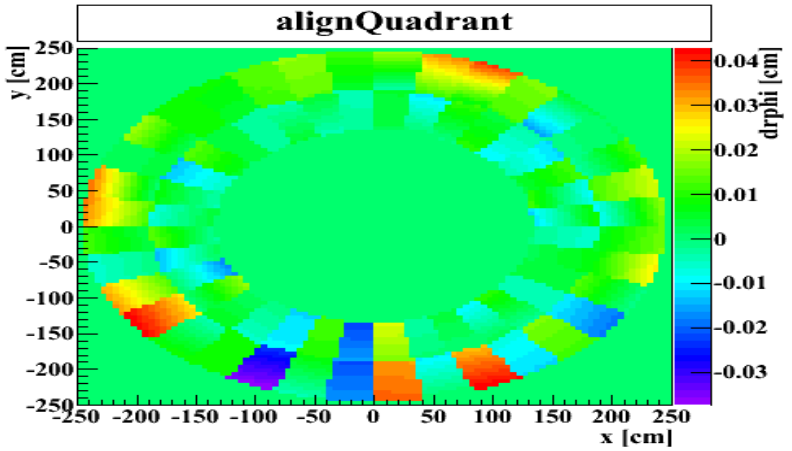
B field 0 data used for the alignment/calibration

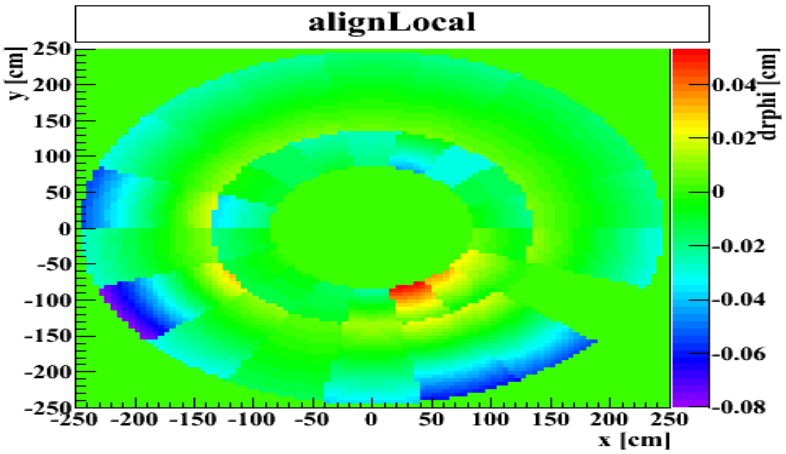
- Fitting of the distortion maps

Alignment

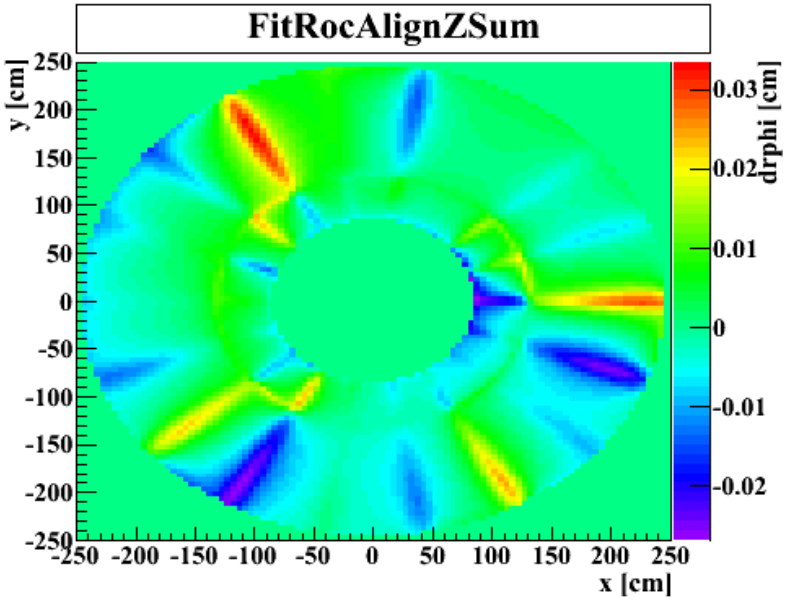

alignQuadrant
y [cm]
x [cm]
drphi [cm]

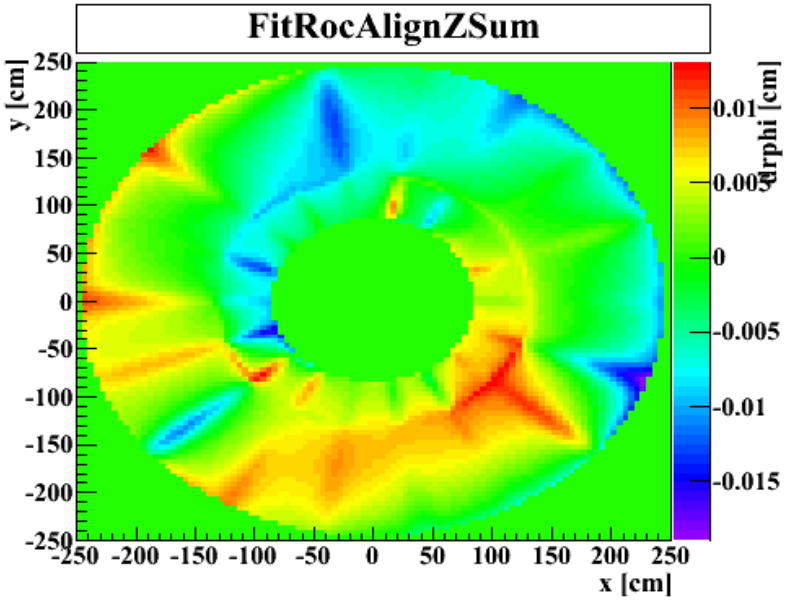

alignLocal
y [cm]
x [cm]
drphi [cm]


alignQuadrant
y [cm]
x [cm]
drphi [cm]


alignLocal
y [cm]
x [cm]
drphi [cm]

ROC z alignment





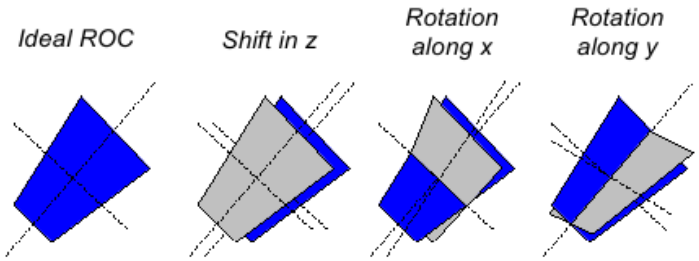


Figure 8: Misalignment scenarios of the ROCs in z

Time dependent alignment

The default calibration entry for the space point distortion extracted using the data from the LHC10b, LHC10c and LHC10e period

Additionally time dependent alignment correction applied on the top of mean correction

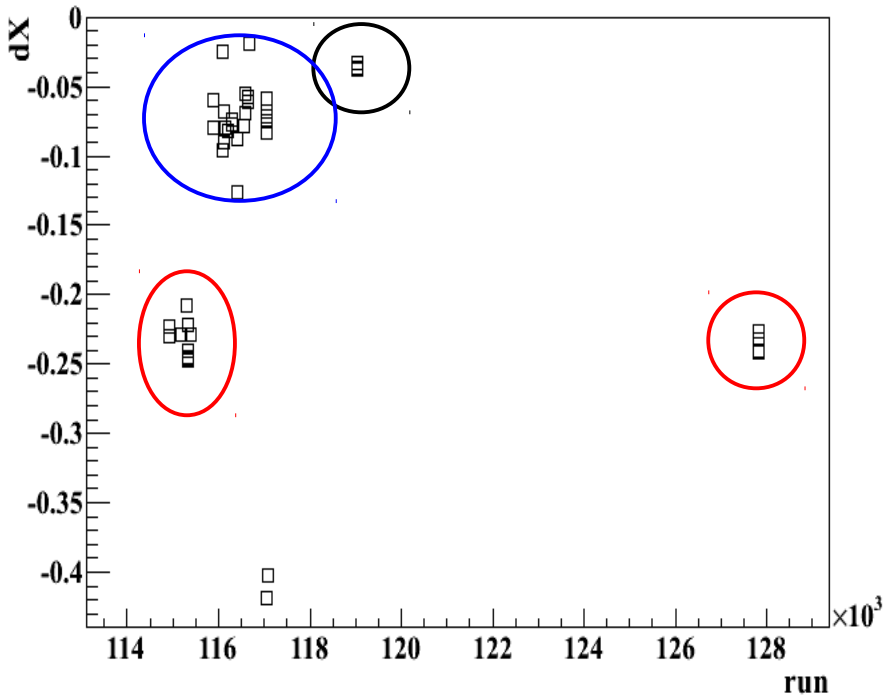
Calibration entry granularity - run by run

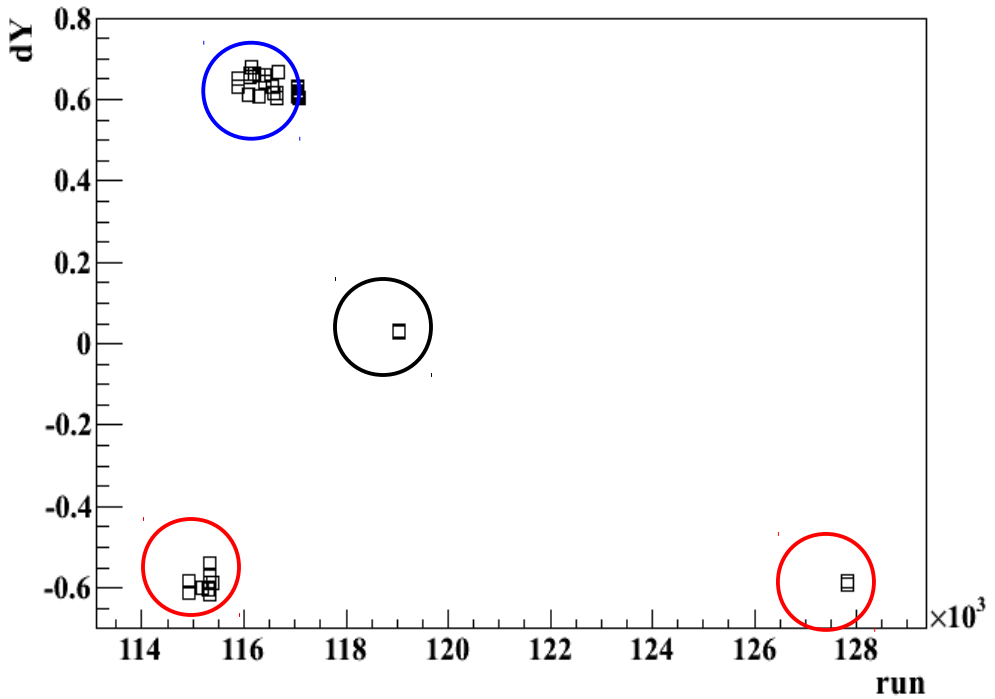
- Detector movement mainly switching the magnetic field polarity

Two models:

- Alignment in respect with other detectors
- Alignment in respect to the magnetic field (ExB Twist)

TPC-ITS alignment





B=-0.5 T

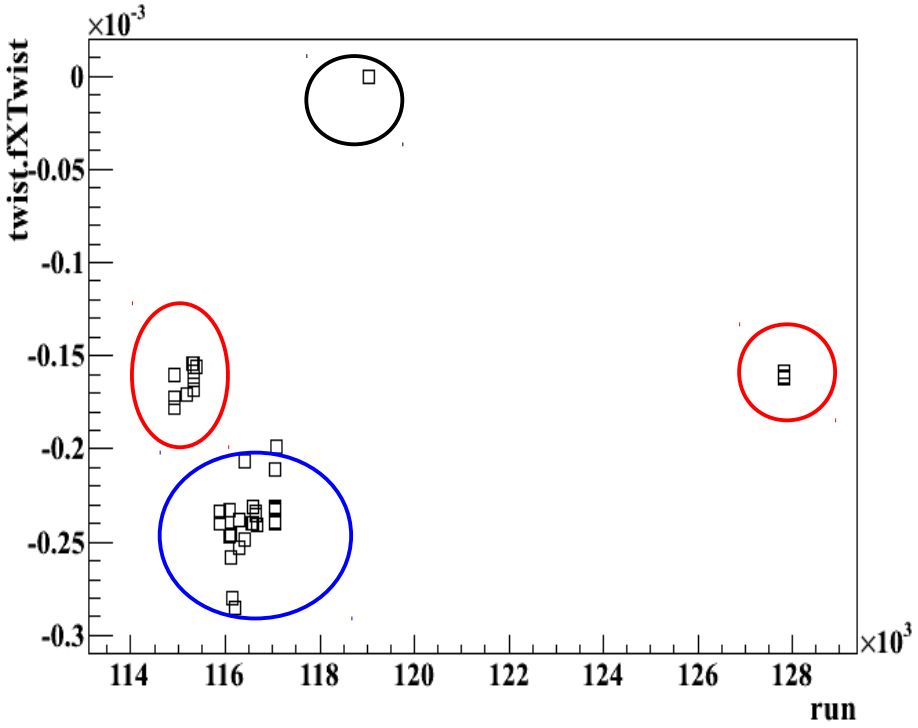
B=0

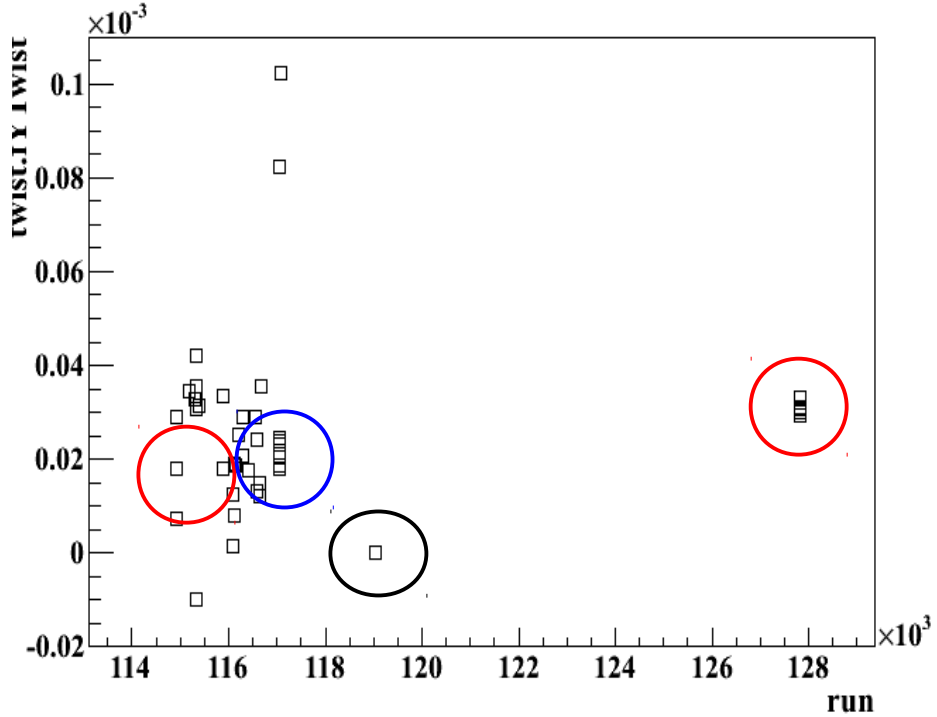
B=0.5 T

Delta X and delta Y of the TPC-ITS translation (mm)

Calibration part of the Pass 0. Used later in the reconstruction.

ExB twist





B=-0.5 T

B=0

B=0.5 T

ExB twist angle parameters (rad)

Calibration part of the Pass 0. Used later in the reconstruction.

Global fits and linear models
(MI, Ana Kreshuk)

Distortion/alignment calibration

Assumptions/Approximations:

Space point distortion commute (the order of applying of corrections is not important)

Space point distortion can be approximated as a linear combination of the "partial distortion" functions with given parameter:

- $\Delta = \Sigma k_i * E_i$

Space point distortion not directly observed. We define the set of observables O.

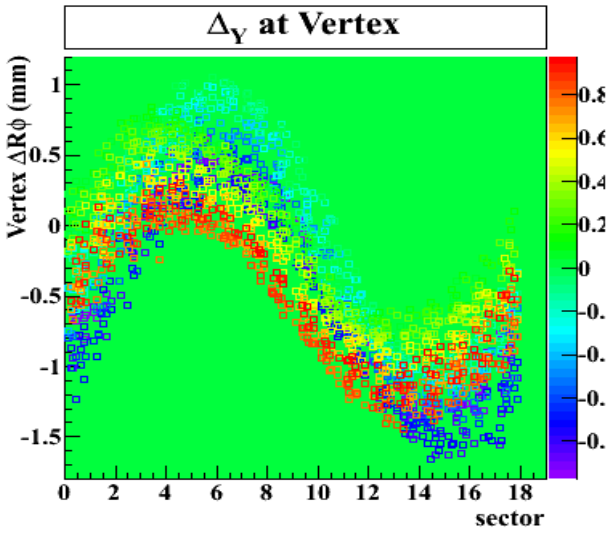
- $\Delta O = \Sigma k_i * Oe_i$

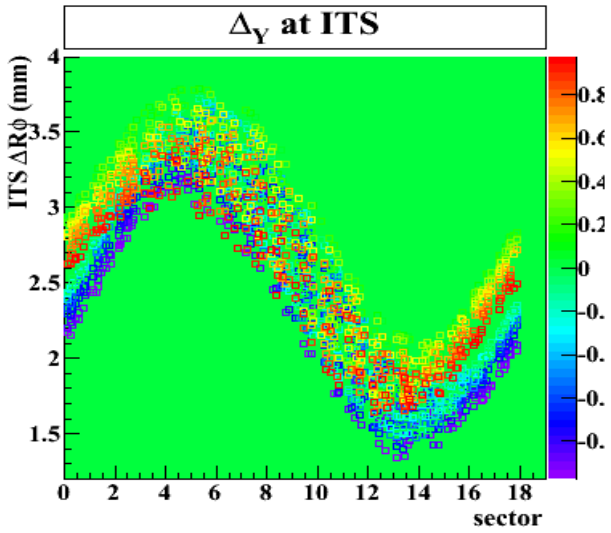
Under given assumption the analytical (not iterative) Global minimization can be performed solving the set of linear equations.

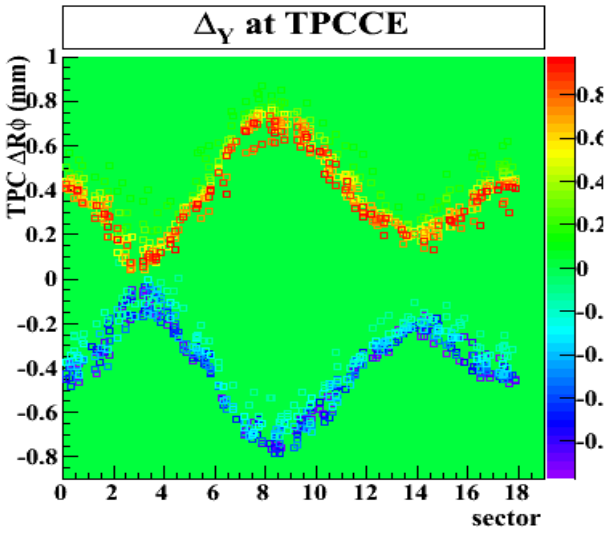
- Important to use unbiased set of observables

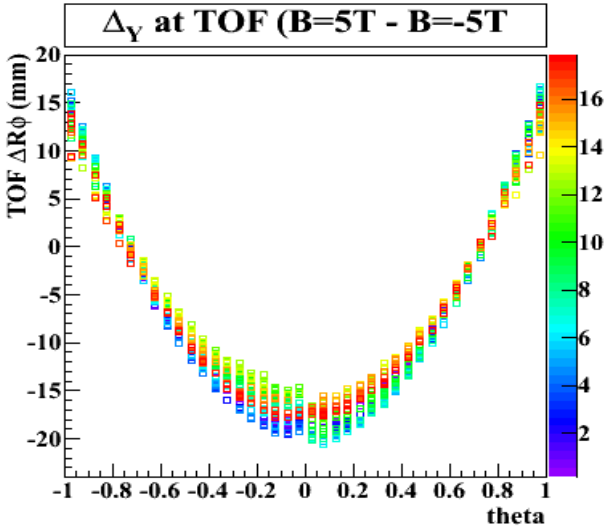
Assumptions were tested for the "typical" distortion in the TPC, moreover the assumption were tested also for the fitted parameters.

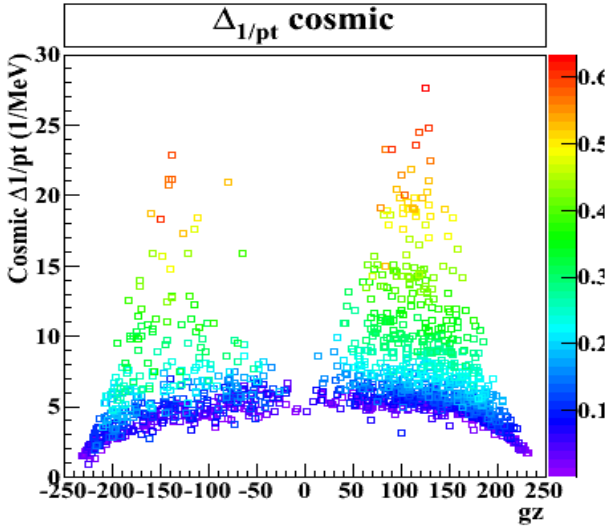
Observable models example: ExB effect due B field non linearities

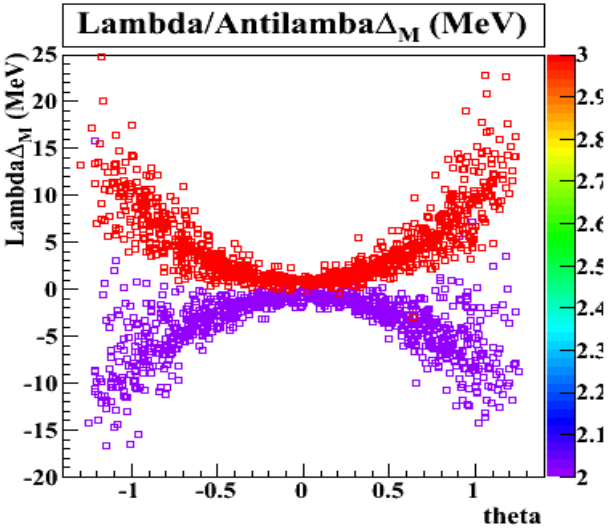












Observed distortion linear combination of partial distortions

Internal TPC calibration (2 parameters)

Fitting/minimization

```
   //
   //Linear fitter helper function
   //
   static TString* FitPlaneConstrain(TTree * tree, const char* drawCommand, const char* formula, const char*?
  cuts, Double_t & chi2, Int_t &npoints,  TVectorD &fitParam, TMatrixD &covMatrix, Float_t frac=-1, Int_t st?
art=0, Int_t stop=10000000, Double_t constrain=-1);
   static Int_t GetFitIndex(TString fString, TString subString);
  static TString FilterFit(TString &input, TString filter, TVectorD &vec, TMatrixD &covar);
  static void Update1D(Double_t delta, Double_t sigma, Int_t s1, TMatrixD &param, TMatrixD &covar);
   static void   Constrain1D(TString &input, TString filter, TVectorD &param, TMatrixD & covar, Double_t mea?
n, Double_t sigma);
   static TString  MakeFitString(TString &input, TVectorD &param, TMatrixD & covar);
```

Numerical part based on the linear fitting package implemented in the ROOT (TmatrixD, TlinearFitter)

Additional functionality implemented in the class TstatToolkit

- Input data observables and fit models from the tree
- Possibility to add constrains
- Possibility to check the the fit values (return value of the FitPlaneConstrain can be used as a alias in tree)
- Extraction of the partial fits

Pass 0 -Gobal fit example - AliTPCPreprocessorOffline

```
TString fstringFast="";
fstringFast+="FExBTwistX++";
fstringFast+="FExBTwistY++";
fstringFast+="FAlignRot0D++";
fstringFast+="FAlignTrans0D++";
fstringFast+="FAlignTrans1D++";
//
fstringFast+="hasITS*FAlignTrans0++";
fstringFast+="hasITS*FAlignTrans1++";
fstringFast+="hasITS*FAlignRot0++";
fstringFast+="hasITS*FAlignRot1++";
fstringFast+="hasITS*FAlignRot2++";
//
```

```
TString *strDeltaITS = TStatToolkit::FitPlaneConstrain(fAlignTree,"mean:err", fstringFast.Data(),cutFit,
hi2,npoints,param,covar,-1,0, npointsMax, 1);
strDeltaITS->Tokenize("++")->Print();
fAlignTree->SetAlias("fitYFast",strDeltaITS->Data());
''
```

```
''
TVectorD paramC= param;
TMatrixD covarC= covar;
TStatToolkit::Constrain1D(fstringFast,"Trans0D",paramC,covarC,0, 0.1);
TStatToolkit::Constrain1D(fstringFast,"Trans1D",paramC,covarC,0, 0.1);
TStatToolkit::Constrain1D(fstringFast,"TwistX",paramC,covarC,0, 0.1);
TStatToolkit::Constrain1D(fstringFast,"TwistY",paramC,covarC,0, 0.1);
TString strFitConst=TStatToolkit::MakeFitString(fstringFast, paramC,covar);
fAlignTree->SetAlias("fitYFastC",strFitConst.Data());
CreateAlignTime(fstringFast,paramC);
```

Define the model

↓

Fit model

↓

Additional constraints of particular parameters

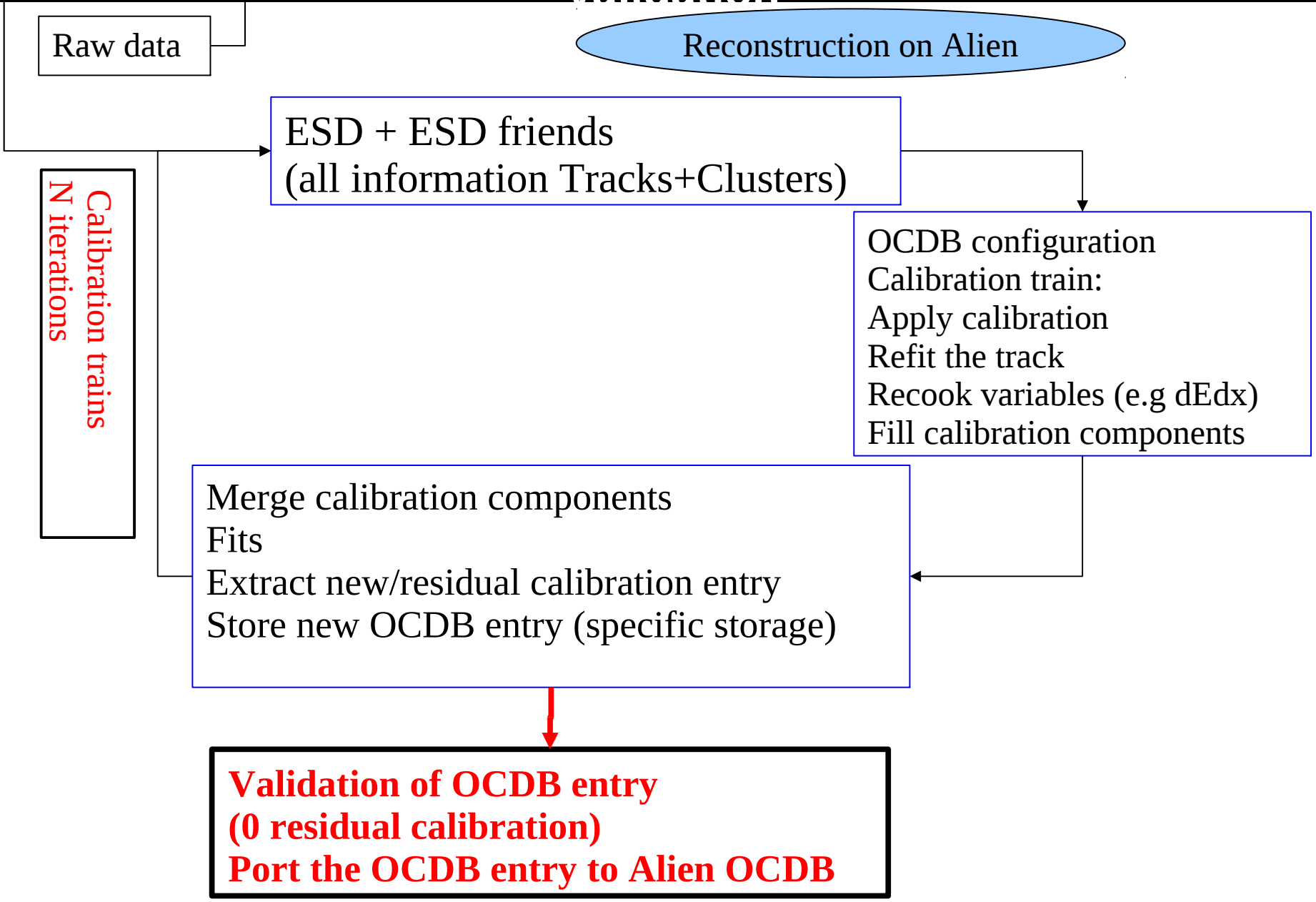
↓

Create calibration entry

Pass 0, Pass X calibration

(MI, Jacek Otwinowski, Alexander Kalweit, Jens Wiechula, ++)

TPC calibration train – calibration and calibration validation



TPC calibration components

AliTPCcalibCalib
Reapply transformations
According OCDB setting
Refit the track

AliTPCcalibTime
Drift velocity calibration
Inter-detector alignment
Filing of the detector matching residual
Histograms
Time granularity – 15 min

AliTPCcalibTimeGain
Gain equalization
in time and space (el. Attachment)
Time granularity – 15 min

Light calibration
OCDB entries per run

AliTPCcalibLaser
Laser residual histograms
Used later for global space point distortion calibration

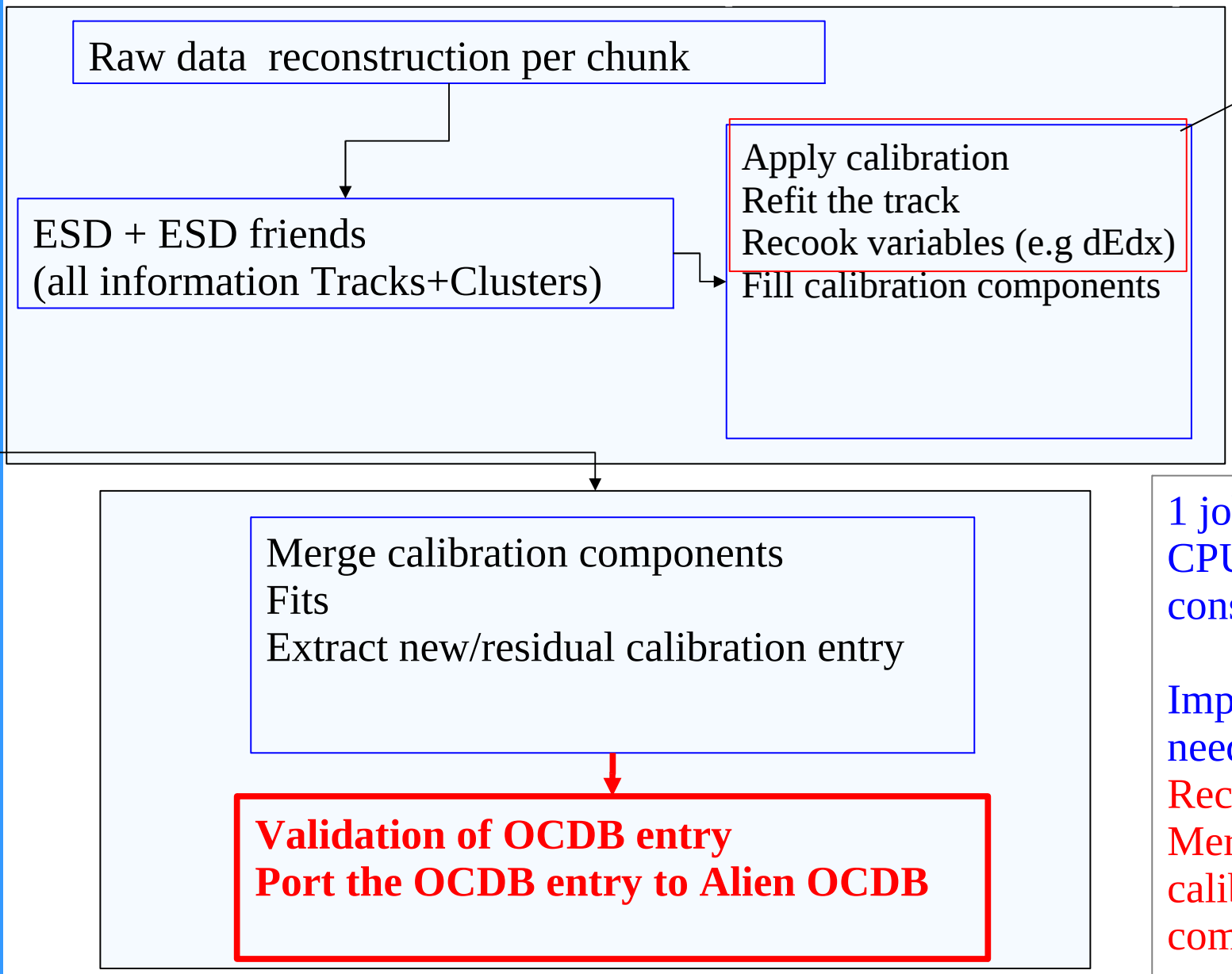
AliTPCcalibAlign
Cluster residual histograms
Tracklet residual histograms

AliTPCcalibV0
Filtering of the high pt V0

AliTPCcalibCosmic
Filtering of the cosmic track candidates

Used for space point calibration
OCDB entries per group of runs

Pass 0 / PassX (TPC oriented view)



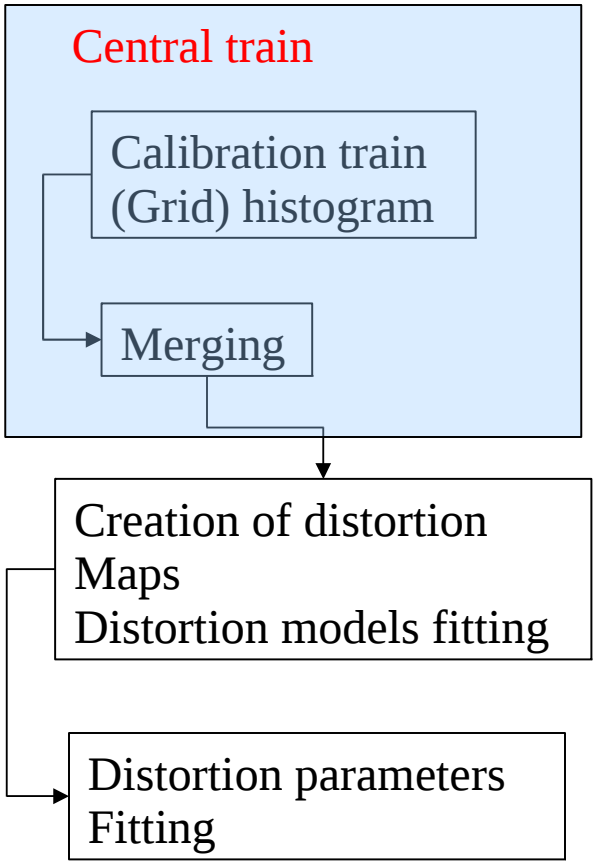
Binned Observables / Global alignment

Generalization of the algorithm for all outer outer detectors

- Global alignment

Possible to integrate the algorithm in the Pass0 or PassX calibration

- Detector matching – residual histogram
  - Differential Residual (4D) histogram in bins of phi,eta,1/pt
    - 20 000 bins
    - 10^6 events - 10^7 tracks ==>5000 entries in bin
- Creation of distortion maps
  - Very small Statistical error = rms/sqrt(entries) e.g
    - TPC-ITS ~ 20 microns
    - TPC-TOF ~ 300 microns
- Global fitting of distortion parameters



Steering macro

- runPass0.sh 2 aliroot processes
  - Reconstruction (recPass0.C)
  - Calibration (runCalibTrain.C)
- mergeMakeOCDB.sh - 2 processes
  - Merging – merge.C
  - OCDB extraction - makeOCDB.C
- validationMerging.sh
- validation.sh

Detector specific code

- Detector specific code should be implemented in the code loaded with AliRoot, the rest (ANLAYSIS dependent) in the libXXXcalib library e.g libT0calib.so custom par file are not allowed
  - Library needed for the pass0, PassX calibration loaded in the STEERING macro LoadLibraries
- Example compiled code to create detector calibration library::
  - https://alisoft.cern.ch/AliRoot/trunk/T0/CMakelibT0calib.pkg
  - https://alisoft.cern.ch/AliRoot/trunk/T0/T0calibLinkDef.h
  - https://alisoft.cern.ch/AliRoot/trunk/T0/AliT0PreprocessorOfflin e.cxx => Resonsible to create OCDB
  - https://alisoft.cern.ch/AliRoot/trunk/T0/AliT0CalibOffsetChannelsTas ==>Calibration task(s) itself

Detector specific configuration should be implemented inside of the AddTaskXXXCalib.C

- Example: https://alisoft.cern.ch/AliRoot/trunk/ANALYSIS/CalibMacros/Pass0/AddTaskTPCCalib.C