Table of Contents

8.7 Application of Alignment and Calibration Constants (AKA Miscalibration and Misalignment)

Complete:
Detailed Review status

Contents:

Goals of this page:

This page explains how to simulate the effect of misalignment and miscalibration in Monte Carlo studies. In particular:

- how to misalign the tracker geometry according to misalignment scenarios;
- how to apply custom (mis)alignment to the tracker geometry;
- how to miscalibrate the rechit energies in the ECAL starting from uncalibrated rechits;
- how to miscalibrate the rechit energies starting from already produced rechits;

Introduction

In order to simulate the imperfect calibration (resp. alignment) of the detector, one has to artificially miscalibrate the calorimeters (resp. misalign the geometry). The tools presented here have been developed for this purpose.

The tools described on this page were used to populate the offline conditions database with expected miscalibration and misalignment scenarios corresponding to startup, 1pb-1 and 10pb-1 of integrated luminosity for the CSA08 alignment and calibration exercises.

For more information about including conditions data from the offline conditions database in a CMSSW job, see SWGuideFrontierConditions.

Set up your Environment

Setup shown for %WBRELEASENEW%; use release noted in newsbox above.

- Full instructions
- Summary for every login session

If you already have a release based on CMSSW_%WBRELEASENEW%, set your runtime environment with
`cd CMSSW_%WBRELEASENEW%/src`
`eval `scramv1 runtime -csh`` or
`eval `scramv1 runtime -sh`, depending on your shell

Tracker Misalignment

Tracker Misalignment scenarios which are ready to be used for physics and other analysis are available in the database and can be accessed via Frontier. They are usually part of a Global Tag.

The following part of the section is out of date, will be adjusted to CMSSW_2_1_X "soon"...

NOTE All files used in this section can be found also in
`/afs/cern.ch/cms/Tutorials/september06/Misalignment/CMSSW_1_6_0_files`.

The misalignment is performed by applying random movements at the various levels of the tracker geometry (layers/disks, rods/strings/petals, modules). The average size of these movements represents the knowledge of the alignment at a given time. Different scenarios have been prepared, corresponding to different stages of the experiment. The scenarios are eventually stored in the offline database in the form of global positions and orientations of all the sensitive modules used by the reconstruction. In order to apply misalignment, one can:

- simply read a prepared scenario from the database;
- use the misalignment tool to read in a `cff` corresponding to a scenario (e.g. allowing to change the random movements);
- write a custom (mis)alignment file (can also be used for alignment "by hand").

In any case, at least part of the (global) reconstruction has to be re-done with the misaligned geometry.

Reading the scenario from the database

In this tutorial, we start with a sample of reconstructed events. We modify the configuration file used for the final CTF fit (baseline Kalman track fit) in order to refit the reconstructed tracks in the misaligned geometry. The transverse momentum of the tracks is then compared before and after misalignment.

We use a simple configuration file to refit (see attached file `misalign-fromDb.cfg`). Here are some **comments** on this file:

- *Database setup:* the `toGet` parameter tells which tag to retrieve. Here, we retrieve the global positions and errors on the global positions from the "100/pb scenario", representing the alignment knowledge after few months of physics run, or 100/pb. The data is read through the Frontier cache.

- *Refitting:* we include the configuration fragment performing the track refit, using the final CTF tracks as input, `RefitterWithMaterial.cff`.

- *Misalignment:* the object read from the database has to be applied to the geometry, which is built by the `TrackerDigiGeometryESModule`. So we ask this module to apply this object with **`replace TrackerDigiGeometryESModule.applyAlignment = true`**, overriding the default value.

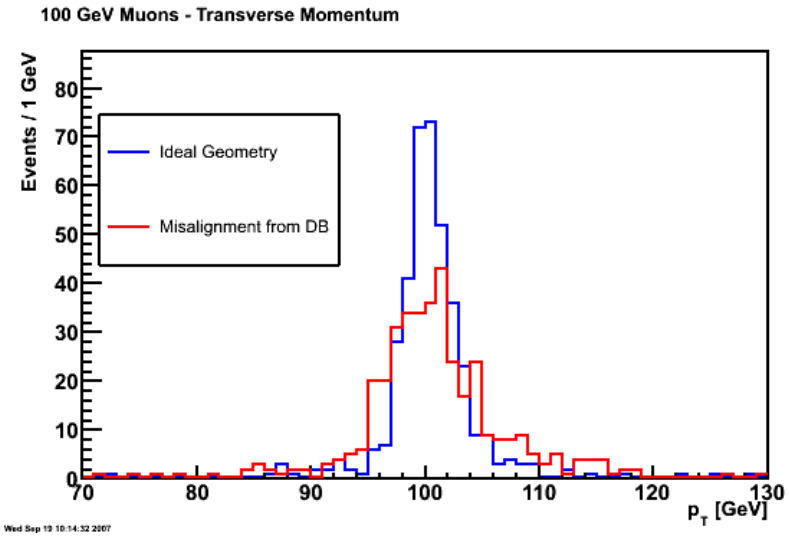
- *Input:* we read in a sample of 400 single muon events with a `pt` of 100 GeV/c. This already includes the full reconstruction. We are going to add in the output file the information on tracks refitted with a misaligned tracker.

- *Output:* we only store in the output file the branches we might be interested in, *i.e.* those filled by the CTF track finder (labelled as `ctfWithMaterialTracks`) and those filled by the refitter (labelled as `TrackRefitter`).

Now, run the configuration file:
`cmsRun misalign-fromDb.cfg`

It creates a root file we can now examine. The small attached macro will plot the transverse momentum distribution before and after misalignment for you:
`root -l compare-pt.C`

We now see the effect of misalignment on this distribution:



Note: More information on the available tracker misalignment scenarios can be found under TkMisalignmentScenariosCSA07 .

Applying custom misalignment

Custom misalignment can be applied using the **Misalignment Tools**. We do this using a second cfg file, the attached **`misalign-custom.cfg`**. This file uses the 10/pb misalignment scenario as a base and modifies it.

- The 10/pb scenario cfg file **`10pbScenario.cff`** is included.

- In addition we have to override the geometry so that the misaligned tracker geometry is used:

```
es_prefer MisalignedTracker = MisalignedTrackerESProducer {}
```

- We modify the short term scenario by applying scale factors for the misalignments of the tracker:

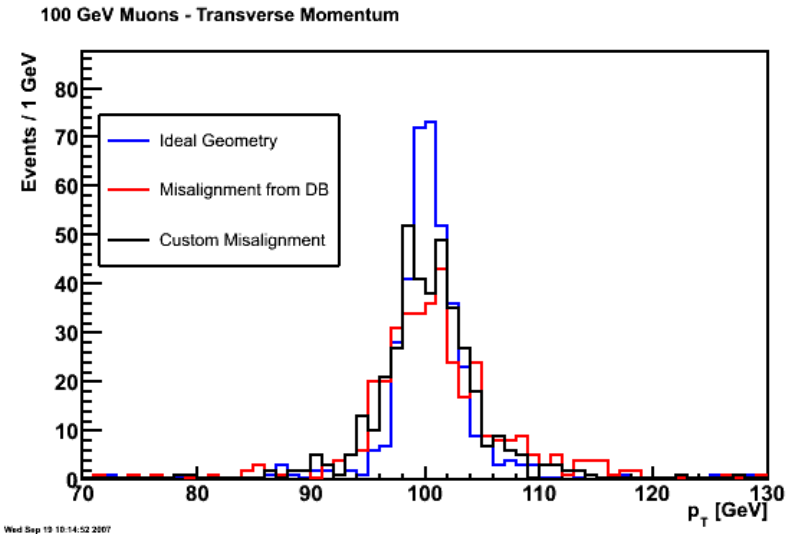
```
replace Tracker10pbScenario.TPBs.scale = 0.2
replace Tracker10pbScenario.TPEs.scale = 0.2
replace Tracker10pbScenario.TOBs.scale = 0.2
replace Tracker10pbScenario.TIDs.scale = 0.2
replace Tracker10pbScenario.TECs.scale = 0.2
```

- With these modifications the misalignment should be similar to the 100/pb scenario.

Now, run this configuration file:
`cmsRun misalign-custom.cfg`

If we run the root macro again:
`root -l compare-pt.C`

We should obtain the following result:



For more details on the configuration of the misalignment scenarios and the available parameters, please refer to SWGuideMisalignmentTools .

ECAL Miscalibration

Two different possibilities are provided to (mis)calibrate the ECAL rechit energies. The first option is to apply a miscalibration factor when the rechits are reconstructed from uncalibrated hits. The second option is useful in the case the rechits have been already reconstructed and one wants to apply a miscalibration. Both methods rely on the use of the package Calibcalorimetry/CaloMiscalibTools.

Reconstruction of miscalibrated rechits

There are only few modifications to the configuration file that are needed in order to perform ECAL rechit energy miscalibration. The only important thing to keep in mind in this case is that rechits reconstruction has to be included in the path since we are starting from a RAW input file. During reconstruction, condition data (ex. intercalibration constants) is retrieved from the DB. Including the ES source CaloMiscalibTools in the configuration file overrides the intercalibration constants used in the rechits reconstruction.

1-Checkout the Calibration/EcalCalibAlgos and CalibCalorimetry/CaloMiscalibTools using tag miscalibTutorial_216:

```
cvs co -r miscalibTutorial_216 Calibration/EcalCalibAlgos
cvs co -r miscalibTutorial_216 CalibCalorimetry/CaloMiscalibTools
```

2-go to the test directory in Calibration/EcalCalibAlgos, where you will find the following configuration file:

- miscalExample1_cfg.py

It takes as input a RAW RelVal sample containing single electrons (Pt=35 GeV). Conditions to be used in the reconstruction (which includes calibration constants) are defined by a Global Tag in the offline conditions database, set up through FrontierConditions_CMS.GlobalTag_cff.py (see SWGuideFrontierConditions. The following module is used here to override the Ecal calibration constants in the database, by providing an alternative ESSource for the cosntants, and using a prefer statement to indicate that these constants should be taken in preference to those in the database:

```
process.CaloMiscalibTools = cms.ESSource("CaloMiscalibTools",
    fileNameEndcap = cms.untracked.string('miscalib_endcap_0.05.xml'),
    fileNameBarrel = cms.untracked.string('miscalib_barrel_0.05.xml')
)
process.prefer("CaloMiscalibTools")
```

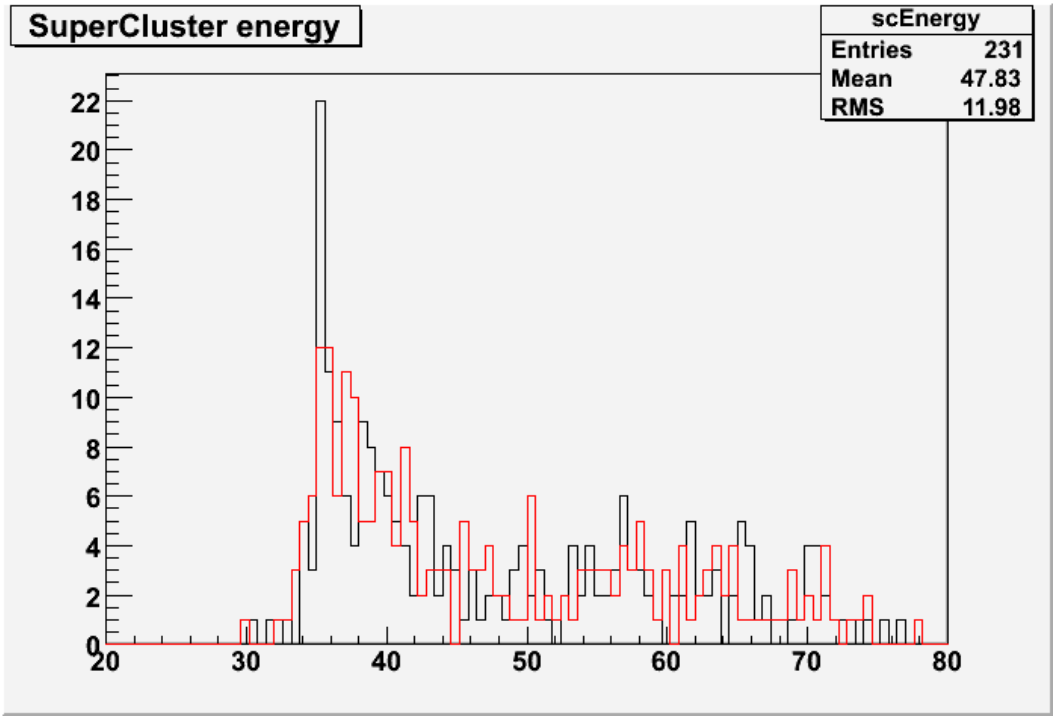
- The files *miscalib_barrel_0.05.xml* and *miscalib_endcap_0.05.xml* contain miscalibration constants associated to each crystal with a Gaussian spread of 5%. These files can be easily produced using the script Calibration/EcalCalibAlgos/test/make_miscalibration.py.

3-run the example:

`cmsRun nomiscalExample_cfg.py` without any miscalibration (to compare)

`cmsRun miscalExample1_cfg.py`

Starting from perfectly calibrated data, the comparison of the distributions of the energy reconstructed in the supercluster looks like:



Miscalibration of rechits (ECAL)

In the case where rechits have been already reconstructed, one can use EcalRecHitRecalib to miscalibrate them. Also in this case few additional lines in the configuration file are sufficient to perform the task. The exercise con be performed in the same way as the previous one, just run:

```
cmsRun miscalExample2_cfg.py
```

Miscalibration constants are taken from the xml file as in the previous example but an additional module is placed in the path before the creation of high level objects in order to re-scale the rechit energies:

```
process.CaloMiscalibTools = cms.ESSource("CaloMiscalibTools",
    fileNameEndcap = cms.untracked.string('miscalib_endcap_0.05.xml'),
    fileNameBarrel = cms.untracked.string('miscalib_barrel_0.05.xml')
)
process.prefer("CaloMiscalibTools")

process.miscalrechit = cms.EDFilter("EcalRecHitRecalib",
    barrelHitCollection = cms.string('EcalRecHitsEB'),
    endcapHitCollection = cms.string('EcalRecHitsEE'),
    ecalRecHitsProducer = cms.string('ecalRecHit'),
    RecalibBarrelHitCollection = cms.string('EcalRecHitsEB'),
    RecalibEndcapHitCollection = cms.string('EcalRecHitsEE')
)

process.hybridSuperClusters.ecalhitproducer = 'miscalrechit'
process.correctedHybridSuperClusters.recHitProducer = cms.InputTag("miscalrechit","EcalRecHitsEB"
```

The output file is *miscalibExample2.root*. The energy distribution should be the same as in *miscalibExample1.root*.

HCAL Miscalibration

The possibility to miscalibrate at the level of RecHit, as described above for ECAL, is also available.

Miscalibration of rechits (HCAL)

Quick instructions: 1-Checkout the CalibCalorimentry/CaloMiscalibTools using tag miscalibTutorial_216:

```
cvs co -r miscalibTutorial_216 CalibCalorimetry/CaloMiscalibTools
```

2-go to the test directory, you will find the following configuration files: hcalmiscal.cfg It takes as input a RECO RelVal sample for QCD di-jets (Pt=80-0 GeV). The following module is used miscalibrate the already reconstructed RecHits.

```
process.recalrechit = cms.EDFilter("HcalRecHitRecalib",
    hbheInput = cms.InputTag("hbhereco"),
    RecalibHFHitCollection = cms.string('RecalibHF'),
    hfInput = cms.InputTag("hfreco"),
    hoInput = cms.InputTag("horeco"),
    RecalibHBHEHitCollection = cms.string('RecalibHBHE'),
    Refactor_mean = cms.untracked.double(1.0),
    Refactor = cms.untracked.double(0.5),
    fileNameHcal = cms.untracked.string('hcalmiscalib_0.1.xml'),
    RecalibHOHitCollection = cms.string('RecalibHO')
)
```

The file *hcalmiscalib_0.1.xml* contains miscalibration constants associated to each HCAL cell with a Gaussian spread of 10%. This file can be easily produced using the script Calibration/EcalCalibAlgos/test/make_miscalibration.py.

The optional section:

```
process.load("Configuration.StandardSequences.GeometryPilot2_cff")
process.load("Configuration.StandardSequences.MagneticField_38T_cff")
process.load("Configuration.StandardSequences.Reconstruction_cff")
process.towerMaker.hbheInput = 'recalrechit:RecalibHBHE'
process.towerMaker.hfInput = 'recalrechit:RecalibHF'
process.towerMaker.hoInput = 'recalrechit:RecalibHO'

process.p = cms.Path(process.recalrechit*process.caloTowersRec)
```

shows how to re-reun the calotower maker using the recalibrated rechits. A similar procedure must be used to produce new objects (candidates, jets, etc.) based on re-calibrated rechits.

3-run the example:

=cmsRun hcalmiscal.cfg

It is also possible to apply low-level miscalibration using tools, provided by HCAL experts, which overwrite constants (pedestals,gains.etc) in the DB using txt files. For completeness, a detailed description is provided in the attached file but, given the technical details it is considered an "expert tool" and, as such, it is out of scope for this tutorial.

Generation of miscalibration files (XML format)

In the test directory of the CalibCalorimetry/CaloMiscalibTools you will find the make_miscalibration.py tool.

Usage:

```
./make_miscalibration.py <barrel|endcap|HCAL> <lumi> <filename> [MINRES=0.02] [SEED=random]
```

Barrel/endcap arguments refers to ECAL and are treated separately while HCAL tag will generate a XML file for the hadronic calorimeter. The second argument is the luminosity (in fb-1) and it's used in ECAL to simulate the expected accuracy vs statistics; by supplying lumi=0, precalibration constants are produced with a precision of 2% or whatever is specified in the optional MINRES argument. The last argument that must be specified is the XML file name. The optional SEED argument allows to set the random seed (for reproducibility)

Applying CSA08 Miscalibration and Misalignment in CMSSW_1_8_X

Miscalibrated and misaligned conditions data are not available in the conditions database for the 18X release series. Miscalibrated and misaligned scenarios are currently defined only for the 16X series for CSA07 (10pb-1 and 100pb-1) and 20X series for CSA08 (startup, 1pb-1 and 10pb-1).

It is possible use the miscalbrated/misaligned scenarios defined for 20X in a 18X release by applying miscalibration and misalignment on the fly using the tools described above. Some specific instructions are as follows:

Tracker misalignment:

1. Add to your cfg:

```
 include "Alignment/TrackerAlignment/data/Scenarios.cff"
 es_prefer MisalignedTracker = MisalignedTrackerESProducer {
     using TrackerSurveyLASCosmicsScenario
 }

# Refit tracks with misalignment
include "RecoTracker/TrackProducer/data/RefitterWithMaterial.cff"
include "RecoTracker/TransientTrackingRecHit/data/TransientTrackingRecHitBuilderWithoutRefit.
replace TrackRefitter.TTRHBuilder = "WithoutRefit"
```

2. Add the module TrackRefitter to your path of your cfg. Note that this (and the corresponding lines in 2.) is only needed if you are reading in events with already reconstructed tracks.

The startup and 1pb-1 scenarios are the same. For the 10pb-1 scenario, replace TrackerSurveyLASCosmicsScenario in the above with 10pbScenario.

Muon misalignment:

1. Check out the following packages:

```
cvs co -r V03-00-06-02 Alignment/CommonAlignment
cvs co -r V02-01-00-01 Alignment/CMS.MuonAlignment
scramv1 build
```

2. Add to your cfg:

```
     include "Alignment/CMS.MuonAlignment/data/Scenarios.cff"
     es_module mualign = MisalignedMuonESProducer {
         using Muon0inversePbScenario2008
     }
     es_prefer mualign = MisalignedMuonESProducer {}

    # Refit tracks with misalignment
    include "RecoTracker/TrackProducer/data/RefitterWithMaterial.cff"
    include "RecoTracker/TransientTrackingRecHit/data/TransientTrackingRecHitBuilderWithoutRefit.
    replace TrackRefitter.TTRHBuilder = "WithoutRefit"
```

3. Add the module TrackRefitter to your path of your cfg. Note that this (and the corresponding lines in 2.) is only needed if you are reading in events with already reconstructed tracks.

For 1pb-1 and 10pb-1 scenarios, replace Muon0inversePbScenario2008 in the above with Muon1inversePbScenario2008 and Muon10inversePbScenario2008 respectively.

ECAL Miscalibration:

1. Check out the following package:

```
cvs co -r V00-02-09 CalibCalorimetry/CaloMiscalibTools
```

2. Add to your cfg:

```
es_source = CaloMiscalibTools {
   untracked string fileNameBarrel = "miscalib_barrel_startup_csa08.xml"
   untracked string fileNameEndcap = "miscalib_endcap_startup_csa08.xml"
}
es_prefer = CaloMiscalibTools{}
```

The startup and 1pb-1 scenarios are the same. For the 10pb-1 scenario, replace miscalib_barrel_startup_csa08.xml in the above with miscalib_barrel_10pb_csa08.xml (same for endcap).

HCAL Miscalibration:

1. Check out the following package:

```
cvs co -r V00-02-09 CalibCalorimetry/CaloMiscalibTools
```

2. Add to your cfg:

```
 module recalrechit = HcalRecHitRecalib {
    InputTag hbheInput = hbhereco
    InputTag hoInput = horeco
    InputTag hfInput = hfreco

    string RecalibHBHEHitCollection = "RecalibHBHE"
    string RecalibHFHitCollection = "RecalibHF"
    string RecalibHOHitCollection = "RecalibHO"

    untracked string fileNameHcal = "hcalmiscalib_startup.xml"
    untracked double Refactor = 0.5
    untracked double Refactor_mean = 1.0
}
```

For 1pb-1 and 10pb-1 scenarios, replace hcalmiscalib_startup.xml in the above with hcalmiscalib_1pb.xml

and hcalmiscalib_10pb.xml respectively.

Review status

Reviewer/Editor and Date	Comments
FredericRonga - 19 Sep 2007	Updated for CMSSW_1_6_0
Luca Malgeri - 9 Nov 2006	Updated naming convention and added Recalib HCAL
AnneHeavey - 09 Oct 2006	copied over from Sept 06 tutorials
Luca Malgeri - 21 Sep 2006	HCAL moved to attachment
Lorenzo Agostino - 20 Sep 2006	Few changes in ECAL part
Luca Malgeri - 20 Sep 2006	Few changes in ECAL part
Fedor Ratnikov - 19 Sep 2006	Include HCAL miscalibrations
Lorenzo Agostino - 19 Sep 2006	Started miscalibration part
Frank-Peter Schilling - 19 Sep 2006	Extended misalignment part

Responsible: DavidFutyan
Last reviewed by: DavidFutyan - 26 Feb 2008

This topic: CMSPublic > SWGuideMisAlignCalib
Topic revision: r42 - 2009-11-23 - unknown