Table of Contents

EJTERM Exercise: Tracker Alignment

Introduction to Tracker Alignment

The goal of this EJTerm exercise is to familiarize the user with the task of tracker alignment and the effect it has on tracking and physics. The user is required to follow the EJTerm tracking exercise and to become familiar with tracking quantities. Within this exercise, we will familiarize the user with the alignment constants and how to apply misalignment/miscalibration in an analysis as well as where to find pertinent information. We will look at how the tracking quantities are affected by different alignment constants. Then, we will look at how different alignment constants can affect a fairly simple physics process like Z->mu+mu-.

First, we begin with a very short introduction to tracker alignment here.

Track-based validation of alignment constants

This part of the exercise is built upon the tracking exercise described here. We extend the exercise by adding in the hit residuals and examining quantities related to tracker alignment in the case of different alignment constants.

Including hit resduals

1. Include headers for accessing trajectories in `CMS.LhcTrackAnalyzer.cc`:

```
#include "TrackingTools/PatternTools/interface/Trajectory.h"
#include "TrackingTools/PatternTools/interface/TrajectoryMeasurement.h"
#include "TrackingTools/TrackFitters/interface/TrajectoryStateCombiner.h"
#include "TrackingTools/TrajectoryState/interface/TrajectoryStateOnSurface.h"
```

2. Create tree for residuals:

In `CMS.LhcTrackAnalyzer.h`:

```
TTree* alignmentTree_;
double residX_;
double subdet_;
```

In `CMS.LhcTrackAnalyzer.cc`, in `beginJob` method:

```
alignmentTree_= fs->make<TTree>("alignmentTree","residual tree");
alignmentTree_->Branch("residX", &residX_, "residX/D");
alignmentTree_->Branch("subdet", &subdet_, "subdet/D");
```

3. Add in this code snippet for calculating unbiased residuals to the `analyze` method:

Show... Hide

```
Handle<vector<Trajectory> > trajCollectionHandle;
iEvent.getByLabel(ctfTrackCollectionTag_ ,trajCollectionHandle);

for(std::vector<Trajectory>::const_iterator it = trajCollectionHandle->begin(), itEnd = trajColle
        const std::vector<TrajectoryMeasurement> &tmColl = it->measurements();
   for(std::vector<TrajectoryMeasurement>::const_iterator itTraj = tmColl.begin(), itTrajEnd = tm
      if(! itTraj->updatedState().isValid()) continue;

      TrajectoryStateOnSurface tsos = TrajectoryStateCombiner().combine( itTraj->forwardPredicted
          TransientTrackingRecHit::ConstRecHitPointer hit = itTraj->recHit();
```

```
       if (hit->isValid()){
          LocalPoint lPHit = hit->localPosition();
          LocalPoint lPTrk = tsos.localPosition();
          LocalVector res = lPTrk - lPHit;

                residX_ = res.x();
          subdet_ = hit->geographicalId().subdetId();
          alignmentTree_->Fill();
            }
      }
}
```

Offline conditions and global tags

The set of database tags which define the offline conditions are collected into a global tag. A list of the global tags to be used can be found here. A detailed list of the specific alignment constants are given here.

To include a different set of alignment constants over a given global tag, use the `ESPrefer` statement in the following way:

Show... Hide

```
from CondCore.DBCommon.CondDBSetup_cfi import *
process.trackerAlignment = cms.ESSource("PoolDBESSource",CondDBSetup,
                              connect = cms.string('sqlite_file:<YOUR PATH>'),
                              timetype = cms.string("runnumber"),
                              toGet = cms.VPSet(cms.PSet(
                                       record = cms.string('TrackerAlignmentRcd'),
                                       tag = cms.string('Alignments')
                              ))
)
process.trackerAlignmentErr = cms.ESSource("PoolDBESSource",CondDBSetup,
                              connect = cms.string('frontier://FrontierProd/CMS_COND_31X_ALIGNMEN
                              timetype = cms.string("runnumber"),
                              toGet = cms.VPSet(cms.PSet(
                                       record = cms.string('TrackerAlignmentErrorRcd'),
                                       tag = cms.string('TrackerCRAFTScenarioErrors310_mc')
                              ))
)
process.es_prefer_trackerAlignment = cms.ESPrefer("PoolDBESSource", "trackerAlignment")
process.es_prefer_trackerAlignmentErr = cms.ESPrefer("PoolDBESSource", "trackerAlignmentErr")
```

Run and Plot

1. Make sure the run with re-done tracking in the python configuration, `runTrackAna_cfg.py`.

```
process.p = cms.Path(process.re_tracking)
```

2. Run over 100 events **only**:

```
cmsRun runTrackAna_cfg.py
```

3. Run again, but this time using the design geometry. Use the code snippet below to add in the design geometry over the global tag.

```
from CondCore.DBCommon.CondDBSetup_cfi import *
process.trackerAlignment = cms.ESSource("PoolDBESSource",CondDBSetup,
                              connect = cms.string('frontier://FrontierProd/CMS_COND_31X_FROM21X'
                              timetype = cms.string("runnumber"),
                              toGet = cms.VPSet(cms.PSet(
                                       record = cms.string('TrackerAlignmentRcd'),
                                       tag = cms.string('TrackerIdealGeometry210_mc')
                              ))
)
process.es_prefer_trackerAlignment = cms.ESPrefer("PoolDBESSource", "trackerAlignment")
```

4. Copy this macro to your area and use it to plot results:

```
/uscms_data/d2/ntran/JTERM/public/plotAlignmentVariables.C
```

The effect of tracker alignment on physics processes

In this part of the of the exercise, we look at the effect of alignment on the Z->mumu resonance. We have already re-reconstructed simulated Z->mumu events under 3 different alignment conditions:

1. Ideal alignment geometry and errors
2. Startup alignment geometry and errors
3. Non-aligned geometry and startup errors

We have built a very simple analyzer which will reconstruct the Z resonance. Our task will be to look at the effect of the different alignments on Z.

Recipe for setting up the environment

- Open a release and set your environment

```
scramv1 p CMSSW CMSSW_3_3_3
cd CMSSW_3_3_3/src/
cmsenv
```

- Check out the following package

```
cvs co -d EJTERM UserCode/Bonato/EJTERM
```

- Compile

```
scramv1 b -r
```

Data to use

Data has already been prepared for the different alignment scenarios. The paths to the data are given here:

- Design geometry

```
dcap:///pnfs/cms/WAX/11/store/user/ntran/Zmumu/Zmumu_Design_fullReco/44f2df8e258da6c5ff569ddd0fd7
```

- Startup geometry

```
dcap:///pnfs/cms/WAX/11/store/user/ntran/Zmumu/Zmumu_Startup_fullReco/a10eb25f53013a16ca772c8a1b1
```

- Non-Aligned geometry

```
dcap:///pnfs/cms/WAX/11/store/user/ntran/Zmumu/Zmumu_MisalignedReal_fullReco/6676311ed793beffb024
```

Instructions for running the exercise

- Set up the input and output path in your config files. They are in the 'python/' subdirectory.

```
cd EJTERM/ZMuMuAnalysis/python
```

The main steering config is `myAnalyzer_cfg.py` The list of input files is defined here. The output file containing a ROOT TTree with some variables extracted by the analysis code is created by the TFileService defined in `ZMuMuAnalysis_cfg.py`. One can also put it in `myAnalyzer_cfg.py` using the line:

```
process.TFileService.fileName = "[yourfilename]"
```

- Re-compile again (needed because we changed ZMuMuAnalysis_cfg.py which is read by myAnalysis_cfg.py)

```
cd ..
scramv1 b -r
```

- Run CMSSW using myAnalyzer_cfg.py as config

```
cd python/
cmsRun myAnalyzer_cfg.py
```

After that it is done, in the output that you previously specified there will be a ROOT file containing a TTree. Open it and you will have plenty of Z->mumu (provided that you ran your analysis on a Z->mumu sample 😉)

- After running the analyzer of the 3 datasets, you can plot the results using the macro here:

```
/uscms_data/d2/ntran/JTERM/public/plotZs.C
```

-- NhanTran - 16-Dec-2009

This topic: CMSPublic > SWGuideEJTermTrackerAlignment
Topic revision: r19 - 2010-12-08 - SudhirMalik