Table of Contents

Heavy Ion Monte Carlo Track Matching

Complete:

Goal of this page

The aim of this page is to document the code and procedure for matching reconstructed tracks (reco::Track) to Monte Carlo truth (reco::GenParticle) in Heavy Ion events. Heavy Ion events are commonly simulated by mixing or embedding a signal event (corresponding to a high pt process) with a background event (corresponding to the underlying event). The existing track matching code, developed for pp events, only considers the case where the GenParticles belong to the signal event. This pages documents the modified code which can handle the background event as well—where the code lives, how to run it, the input and output, and details of the implementation.

Code and tags

Matching Tracks to GenParticles requires three modules:

- HiGenParticleProducer—produces the collection of GenParticles, and a map linking the event number and particle barcode to the index in the collection of GenParticles.
  - This currently lives in the package UserCode/CmsHi/Utilities.
- TrackingTruthProducer—creates TrackingParticles, which link SimTracks to generator-level tracks.
  - This lives in the package SimGeneral/TrackingAnalysis
- HiMCTrackMatcher—associates reco::Track to SimTracks using the association between TrackingRecHits and PSimHits, then uses the TrackingParticles to connect the SimTracks to generator-level particles, then uses the map produced by HiGenParticleProducer to link the reco::Track to the corresponding reco::GenParticle.
  - This currently lives in the package UserCode/CmsHi/Utilities.

(A more detailed description can be found in the implementation section below)

The code has for these modules has been committed to CVS, but is not yet part of an official CMSSW release. To check out the code type the following commands from the `src` directory of a CMSSW project area from CMSSW_3_3_0_pre4 or later.

```
cvs co UserCode/CmsHi/Utilities
mv UserCode/CmsHi .
cvs co -r CMSSW_3_3_0_pre4 SimGeneral/TrackingAnalysis
cvs co -r1.7 SimGeneral/TrackingAnalysis/src/classes.h
cvs co -r1.5 SimGeneral/TrackingAnalysis/src/classes_def.xml
scramv1 b
```

The `classes.h` and `classes_def.xml` files are not included in the 3_3_0_pre4 version of SimGeneral/TrackingAnalysis, because they contain unofficial data formats. They must therefore be checked out separately. This is not ideal; it is hoped that a new implementation of the MC track matching (which is needed to improve some other issues) will resolve this.

Configuration files

The settings for the track matching are contained in the following configuration files:

- HiGenParticleProducer

    - ♦ UserCode/CmsHi/Utilities/python/HiGenParticles_cfi.py
- TrackingTruthProducer
    - ♦ SimGeneral/TrackingAnalysis/python/trackingParticles_cfi.py
    - ♦ SimGeneral/TrackingAnalysis/python/HiTrackingParticles_cff.py
- HiMCTrackMatcher
    - ♦ UserCode/CmsHi/Utilities/python/HiTrackMCMatch_cfi.py

How to run the track matching

Do you need to run the track matching?

If your events already have an edm::Association<reco::GenParticleCollection> called `hiTrackMCMatch`, then the track matching has already been run, and you do not need to do it yourself. You can check this by either

- opening the file in ROOT, opening a TBrowser, clicking on the Root file, then Event, and looking for a branch called something like `recoGenParticlesedmAssociation_hiTrackMCMatch__RECO.`, or
- typing `edmDumpEventContent` *filename.root* at the command line, and searching for something like `edm::Association<vector<reco::GenParticle> > "hiTrackMCMatch" "" "RECO."` in the output.

Running the track matching

If however your events don't contain this Association (e.g. you are generating your own Monte Carlo events), follow the instructions below.

- In your configuration file, include the lines

```
process.load("CmsHi.Utilities.HiGenParticles_cfi")
process.load("SimGeneral.TrackingAnalysis.trackingParticles_cfi")
process.load("SimGeneral.TrackingAnalysis.HiTrackingParticles_cff")
process.load("SimTracker.TrackAssociation.TrackAssociatorByHits_cfi")
process.load("CmsHi.Utilities.HiTrackMCMatch_cfi")
```

- Then in the path in configuration file, include the modules `hiGenParticles, trackingParticles` and `hiTrackMCMatch`. The location within the path is dependent on the input they need. The module `trackingParticles` must be run in the same job that mixes or embeds the signal and background events, as it requires the CrossingFrame as input. The module `hiGenParticles` can be run using either the CrossingFrame, or the individual HepMCProducts as input (see the Parameters section below), but must be run before `hiTrackMCMatch`. The module `hiTrackMCMatch` must be run after the track reconstruction (so it has the tracks to match) and also after `hiGenParticles` (so it has the genParticles, and the map, to match to).

An example of the full path for a job which embeds a signal event into an existing background event, and also runs all three of the above modules is shown below.

```
process.sim = cms.Sequence(process.hiSignal*process.matchVtx*process.hiSignalG4SimHits*process.mi
process.gen = cms.Sequence(process.hiGenParticles * process.trackingParticles)
process.digi = cms.Sequence(process.doAllDigi*process.L1Emulator*process.DigiToRaw*process.RawToD
process.reco = cms.Sequence(process.reconstruct_CMS.PbPb * process.hiTrackMCMatch)

process.p = cms.Path(process.sim * process.gen * process.digi * process.reco)
```

- In order to retain the result of the track matching (the Association between tracks and GenParticles) in your output file, you should ensure that they are kept by the OutputModule. If your existing `drop` or `keep` statements don't cover this case , add the line

`process.output.outputCommands.append('keep *_hiTrackMCMatch_*_*')` to your configuration file, somewhere after

```
process.output = cms.OutputModule("PoolOutputModule",
 ...other settings...
)
```

Input

The objects required as input for the three modules used in the track matching are listed below. If they are not found in the event the reconstruction will crash. Note that the module `hiGenParticles` can be run using either the CrossingFrame, or the individual HepMCProducts as input (see the Parameters section below)

Product type	Module label(s)	Product instance label(s)
HiGenParticleProducer		
edm::CrossingFrame<HepMCProduct>	mix	generator
edm::HepMCProduct	hiSignal, generator	
TrackingTruthProducer		
edm::HepMCProduct	hiSignal, generator	
edm::CrossingFrame<PSimHit>	mix	g4SimHitsTrackerHitsPixelBarrelLowTof, g4SimHitsTrackerHitsPixelBarrelHighTof, g4SimHitsTrackerHitsPixelEndcapLowTof, g4SimHitsTrackerHitsPixelEndcapHighTof, g4SimHitsTrackerHitsTIBLowTof, g4SimHitsTrackerHitsTIBHighTof, g4SimHitsTrackerHitsTIDLowTof, g4SimHitsTrackerHitsTIDHighTof, g4SimHitsTrackerHitsTOBLowTof, g4SimHitsTrackerHitsTOBHighTof, g4SimHitsTrackerHitsTECLowTof, g4SimHitsTrackerHitsTECHighTof, g4SimHitsMuonDTHits, g4SimHitsMuonCSCHits, g4SimHitsMuonRPCHits
edm::CrossingFrame<SimTrack>	mix	g4SimHits
edm::CrossingFrame<SimVertex>	mix	g4SimVertex
HiMCTrackMatcher		
edm::HepMCProduct	hiSignal, generator	
TrackCollection	hiGlobalPrimTracks	
TrackingParticleCollection	hiMergedTruth	MergedTrackTruth
GenParticleCollection	hiGenParticles	
std::map<EncodedTruthId, unsigned int >	hiGenParticles	

Parameter settings

This section describes the configuration file parameters which the user is most likely to want to modify.

Parameter	Description
Configuration file: UserCode/CmsHi/Utilities/python/HiGenParticles_cfi.py	

`useCrossingFrame = cms.untracked.bool(False)`	GenParticles are produced from HepMCProducts. In both mixed and embedded events these are stored in a CrossingFrame<HepMCProduct>. In embedded events (i.e. one where the signal event is generated on-the-fly) both HepMCProducts are also stored separately in the event. If `useCrossingFrame` is True, the HepMCProducts are taken from the CrossingFrame (which is only present in the job which performs the mixing/embedding). If `useCrossingFrame` is False, the labels specified in the `src` parameter are used to retrieve the HepMCProducts.
`src = cms.vstring('hiSignal','generator')`	If `useCrossingFrame` is False these labels specify the HepMCProducts to be used to create the GenParticles. For the track matching to work the order of the labels must be the same as the order the events are mixed/embedded (i.e. signal event first, background event second). These labels are ignored if `useCrossingFrame` is True.
`saveBarCodes = cms.untracked.bool(True)`	This parameter must be True to produce the map connecting the event & particle number to the index in the hiGenParticles collection. (This map is required by HiMCTrackMatcher.)
Configuration file: SimGeneral/TrackingAnalysis/python/HiTrackingParticles_cff.py	
`useMultipleHepMCLabels = cms.bool(True)`	The default behaviour, with `useMultipleHepMCLabels` False, is to take the first valid label in the parameter `HepMCDataLabels` as corresponding to the HepMCProduct of the signal event, and to only create references from the TrackingParticle to the GenParticles for particles from this signal event. If `useMultipleHepMCLabels` is True, the module uses all valid labels, and creates TrackingParticle–GenParticle references for the background event as well. For the track matching to work on a mixed/embedded event, `useMultipleHepMCLabels` must be True.
`HepMCDataLabels = cms.vstring('hiSignal', 'generator')`	These labels specify the HepMCProducts to be used to create the references to the GenParticles. For the track matching to work the order of the labels must be the same as the order the events are mixed/embedded (i.e. signal event first, background event second).
Configuration file: UserCode/CmsHi/Utilities/python/HiTrackMCMatch_cfi.py	
`trackingParticles = cms.InputTag("mergedtruth","MergedTrackTruth")`	The name of the TrackingParticleCollection
`tracks = cms.InputTag("hiGlobalPrimTracks")`	The name of the TrackCollection. If you apply a track quality selection, e.g. as described in SWGuideHeavyIonTrackReco#Track quality cuts, you will need to use the name of the selected TrackCollection here instead, and also make sure that you run HiMCTrackMatcher after the track selection.
`genParticles = cms.InputTag("hiGenParticles")`	The name of the GenParticleCollection.

Output

The ultimate output of the track matching is an edm::Association<reco::GenParticleCollection> called `hiTrackMCMatch`, which maps the reconstructed tracks to the corresponding GenParticles. The intermediate steps also produce output, which may or may not be of interest to the user.

Product type	Module label	Description
HiGenParticleProducer		
GenParticleCollection	hiGenParticles	The collection of GenParticles, containing the MC truth information. This is what the reconstructed tracks are matched to.
std::vector	hiGenParticles	Vector of particle barcodes, used for standard pp track matching
SubEventMap	hiGenParticles	SubEventMap, used for Heavy Ion generator-level Jet finding
std::map<EncodedTruthId, unsigned int >	hiGenParticles	Map used by HiMCTrackMatcher to connect TrackingParticles to GenParticles. Not needed after the track matching is complete.
HiTrackingTruthProducer		
TrackingParticleCollection	mergedtruth:MergedTrackTruth	The collection of TrackingParticles. If your only interest in TrackingParticlesis as an intermediate step for the track matching, these can safely be dropped from the event after the matching is complete.
TrackingVertexCollection	mergedtruth:MergedTrackTruth	The collection of TrackingVertexes, objects which link the SimVertexes to their generator-level counterparts. Not used by the track matching.
TrackingParticleCollection	mergedtruth	Another collection of TrackingParticles, in which the multiple SimTracks produced by a bremsstrahlung electron are not merged into one TrackingParticle. (See SWGuideTrackingTruth for more information.) This version is not used in the track matching.
TrackingVertexCollection	mergedtruth	

		The corresponding additional collection of TrackingVertexes, Not used by the track matching.
HiMCTrackMatcher		
edm::Association<GenParticleCollection>	hiTrackMCMatch	The association map between the TrackCollection of reconstructed tracks and the GenParticleCollection.

Accessing the track match

Below is a short code fragment that indicates which header files (*.h) need to be included, and demonstrates how to access the genParticle corresponding to a reconstructed track from within the `analyze` function of an EDAnalyzer. You can find a tutorial about how to create an EDAnalyzer at WorkBookWriteFrameworkModule.

```
// user include files
#include "FWCore/Framework/interface/Event.h"
#include "DataFormats/TrackReco/interface/TrackFwd.h"
#include "DataFormats/TrackReco/interface/Track.h"
#include "DataFormats/HepMCCandidate/interface/GenParticle.h"
#include "DataFormats/HepMCCandidate/interface/GenParticleFwd.h"
#include "DataFormats/Common/interface/Association.h"

...

void DemoAnalyzer::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup)
{
   using namespace edm;
   using namespace std;
   using namespace reco;

   Handle<GenParticleCollection >  genParticlesHandle_;
   // Get the GenParticle collection
   iEvent.getByLabel("hiGenParticles",   genParticlesHandle_ );

   Handle<std::vector<reco::Track> >  trackHandle_;
   // Get the track collection
   iEvent.getByLabel("hiGlobalPrimTracks",   trackHandle_ );

   Handle<Association<GenParticleCollection> >  mcMatchHandle_;
   //Get the map from reconstructed tracks to genParticles
   iEvent.getByLabel("hiTrackMCMatch",   mcMatchHandle_ );

   cout <<  "Loop over the tracks\n";
   for(size_t i = 0; i < trackHandle_->size(); ++i) {
     TrackRef track (trackHandle_,i);
     cout << "Track parameters  " << i << " " << track->pt() << " " << track->eta() << " " << tra

     GenParticleRef mc = (*mcMatchHandle_)[track];
     if(mc.isNonnull()){
       cout << "\nMC parameters  " << mc.key() << " " << mc->pt() << " " << mc->eta() << " " << m
     }
     else{
       cout << "\t not matched\n";
     }
     cout << "--------------------\n";
   }
}
```

Implementation details

Before going into the details it should be pointed out that the method of MC track matching described on this page uses track association by hits, rather than the simpler track association by position (which would not require the use of TrackingParticles). This is because the track multiplicity in heavy ion events, especially the most central ones, is so much greater than for pp events that one is almost guaranteed to find a reconstructed track pointing in approximately the same direction as a given generator-level track.

As explained above, the code for matching Tracks to GenParticles in the existing pp software only considers the case where the GenParticles are all created from a single edm::HepMCProduct, even in pileup events, where this is not the case. The most logical way to address this would be to use the CrossingFrame<HepMCProduct> rather than just one HepMCProduct. However this would require changes to the data format of the TrackingParticle class, which is unlikely to happen as part of the official release. It would also require the use of a persistent CrossingFrame, and this class has now been made non-persistent (though a persistent version does exist). The code developed to allow matching of reconstructed tracks to generator-level information of all constituent events comprises three modules: HiGenParticleProducer, TrackingTruthProducer, and HiMCTrackMatcher. The main issue which needed to be addressed was how to transmit the (event number, particle number) information between the different modules. The method eventually adopted is just a generalisation of the pp architecture.

In the pp version of the track matching chain, GenParticleProducer takes one HepMCProduct, and produces a GenParticleCollection and a vector of particle barcodes. TrackingTruthProducer takes the SimTracks, which have a reference to the particle barcode and to the event number within the CrossingFrame, and produces TrackingParticles which have a GenParticleRef to the HepMCProduct (for signal particles only), and also contain an EncodedEventId. MCTrackMatcher then loops over the reconstructed tracks, uses the TrackingRecHit to PSimHit association to get the corresponding TrackingParticle, from the TrackingParticle gets the GenParticleRef to the HepMCProduct and consequently the particle barcode, then uses the vector of barcodes to get the index of the corresponding track within the GenParticleCollection.

In the modified version, we want to use a vector of HepMCProducts, thus to specify a particle uniquely we need the pair (event number, particle barcode). A class comprising these two objects already exists, EncodedTruthId. Thus we essentially replace "barcode" by "EncodedTruthId" and "HepMCProduct" by "vector of HepMCProducts" in the previous paragraph.

Therefore to perform the MC track matching in embedded events:

- HiGenParticleProducer takes a vector of HepMCProducts, and produces a GenParticleCollection and a map<EncodedTruthId, int> mapping (event number, particle barcode) to an index in the GenParticleCollection. The event number is taken as the index of the HepMCProduct within the vector. (the pp version uses a vector rather than a map as it uses the fact that the barcode is equal to the index. This is not actually correct, as barcodes start at 1 and the index at 0. This was discussed in this hypernews thread but no fix was ever implemented).
- TrackingTruthProducer takes the SimTracks, which have a reference to the particle barcode and to the event number within the CrossingFrame, and produces TrackingParticles which have a GenParticleRef to the corresponding HepMCProduct (for all particles). TrackingTruthProducer already took a vector of HepMCProduct labels as input; a flag was added to switch between using the first valid label and all of the labels. The correct HepMCProduct is chosen from the vector of input HepMCProducts by using the event id from the SimTrack as an index (The signal event has index 0 and the background event has index 1). This index is also stored an EncodedEventId in the TrackingParticle
- HiMCTrackMatcher then loops over the reconstructed tracks, uses the TrackingRecHit to PSimHit association to get the corresponding TrackingParticle, from the TrackingParticle gets the GenParticleRef to the HepMCProduct and consequently the particle barcode. It then constructs an EncodedTruthId from EncodedEventId of the TrackingParticle and the particle barcode from the

GenParticleRef, and uses the map of EncodedTruthIds to indices to get the index of the corresponding track within the GenParticleCollection.

Further information

- **Heavy Ions Physics Interest Group (PInG) with track based analyses**: ChargedSpectra
- **Group homepage**: HeavyIons

Contact

- **Hypernews fora**:
  - Software issues: https://hypernews.cern.ch/HyperNews/CMS/get/hiswDevelopment.html, (hn-cms-hiswDevelopment@cernNOSPAMPLEASE.ch)
  - General heavy ions issues https://hypernews.cern.ch/HyperNews/CMS/get/hi.html, (hn-cms-hi@cernNOSPAMPLEASE.ch)
- **Contacts/Developers**: Philip Allfrey

Review status

Reviewer/Editor and Date	Comments
PhilipAllfrey - 7 Aug 2009	created page

Responsible: PhilipAllfrey

This topic: CMSPublic > SWGuideHeavyIonMCTrackMatching
Topic revision: r4 - 2009-09-24 - PhilipAllfrey