Table of Contents

Transient Tracks

Complete:

Usage of TransientTracks objects

For most high-level algorithms, the **`reco::Track`** is unsuitable, since it does not have access to the magnetic field, which prevents one estimating the track parameters at arbitrary points along its trajectory. A **`TransientTracks`** has thus to be used. It has a pointer to the magnetic field, and can be given a pointer to the geometry as well. To easiest and safest way is to build the **`TransientTracks`** using the **`TransientTrackBuilder`**.

Link to Reference Manual (doxygen): TransientTracks , TransientTrackBuilder .

To use the TransientTrackBuilder, you have to add the following into your cfg:

```
process.load("TrackingTools/TransientTrack/TransientTrackBuilder_cfi")
process.load("Configuration.Geometry.GeometryIdeal_cff")
process.load("Configuration.StandardSequences.MagneticField_cff")
process.load('Configuration.StandardSequences.FrontierConditions_GlobalTag_cff')
process.GlobalTag.globaltag = 'GR_P_V42_AN3::All'  # or some other global tag depending on your C
```

You may need to include header files into your analyzer, look for example here: TransientTrackBuilderTest.cc.

To get the TransientTrackBuilder from the EventSetup:

```
edm::ESHandle<TransientTrackBuilder> theB;
setup.get<TransientTrackRecord>().get("TransientTrackBuilder",theB);
```

You can now build your TransientTracks from `Track`, `TrackRef`, or directly from the `edm::handle` to a `TrackCollection`:

```
reco::TransientTrack * build ( const reco::Track * p)  const;
reco::TransientTrack * build ( const reco::TrackRef * p)  const;
std::vector<reco::TransientTrack> build ( const edm::Handle<reco::TrackCollection> & trkColl)
```

You also have to include these packages into your `BuildFile.xml`: **`TrackingTools/TransientTrack`** and **`TrackingTools/Records`**.

Examples (including calculation of track impact parameters)

```
// get RECO tracks from the event
edm::Handle<reco::TrackCollection> tks;
iEvent.getByLabel(trackLabel(), tks);

//get the builder:
edm::ESHandle<TransientTrackBuilder> theB;
iSetup.get<TransientTrackRecord>().get("TransientTrackBuilder",theB);
//do the conversion:
vector<TransientTrack> t_tks = (*theB).build(tks);
```

One can use TransientTracks to estimate **track impact parameters** with respect to the beam line or primary vertex, taking into account the curvature of the track.

```
// with respect to beam-line
TransientTrack tk = ...;
TrajectoryStateClosestToBeamLine traj = tk.stateAtBeamLine();
Measurement1D meas = traj.transverseImpactParameter();
double d0 = meas.value();
double d0_error = meas.error();

// with respect to any specified vertex, such as primary vertex
GlobalPoint vert(pv.x(), pv.y(), pv.z());
TrajectoryStateClosestToPoint  traj = tk.trajectoryStateClosestToPoint(vert );
double d0 = traj.perigeeParameters().transverseImpactParameter();
double d0_error = traj.perigeeError().transverseImpactParameterError();
double z0 = traj.perigeeParameters().longitudinalImpactParameter()
double z0_error = traj.perigeeParameters().longitudinalImpactParameterError()
```

You can get more sophisticated impact parameters (e.g., 3D, signed ...) using the IPTools software, although it won't tell you z0. For example:

```
#include "TrackingTools/IPTools/interface/IPTools.h"
 double d03D = IPTools::absoluteImpactParameter3D(tk, pv).second;
```

where tk is a TransientTrack and pv is the PrimaryVertex.

A few more details

- The **`TransientTrack`** is reference counted.
- Different concrete implementation of the basic TransientTrack exist. This allows their use with different types of basic track objest (**`reco::GsfTrack`**, **`reco::Track`**, **`FreeTrajectoryState`** without persistent track, e.g. for refitted tracks). For a correct usage, the **`TransientTrack`** should be built using the TransientTrackBuilder, or, if you have a FreeTrajectoryState, using the TransientTrackFromFTSFactory.
- In case the **`TransientTrack`** has been created from a persistent track, the method **`track()`** allows to access the **`reco::Track`**. If not, a new **`reco::Track`** is created. This track can also be saved in the event.
- In addition, several **`reco::Track`** methods (e.g. the hit pattern) are implemented, and forward the call to the persistent track. An exception is thrown in case no persistent track is available.
- Since the TT is reference counted, if you request the builder a single TT, you do not get a pointer, and you do not need to delete at the and of your block. So use it like that:

```
const reco::TransientTrack transientTrack = theBuilder->build(&lepton);
```

Review Status

Editor/Reviewer and date	Comments
Main.speer - 17 Jul 2006	Page Author
JennyWilliams - 28 Mar 2007	moved page into SWGuide
ThomasSpeer - 27 Feb 2009	Review and update

Responsible: Main.speer (Thomas Speer)

This topic: CMSPublic > SWGuideTransientTracks
Topic revision: r20 - 2013-08-19 - IanTomalin

or Ideas, requests, problems regarding TWiki? use Discourse or Send feedback