Level 3 Tracking Tutorial

R. Beuselinck

7 February 2001

Contents:

- Current status of level 3 tracking interface
- User access methods and parameter definitions
- Simple examples of how to use tracking tool
- Sample performance data

Glossary of Tracking Tools and Classes

L3TTracker - Base class specifying common user interface for tracking tools.

L3TCFTTrack - CFT only track finder.

L3TGlobalTracker - CFT plus SMT tracker.

L3TrackFit - Basic track fit results, helix fit parameters. Contains an
L3TrackParams and L3TrackErrors object.
Contains methods for track propagation.

L3TrackParams - Contains helix fit information.

L3TrackErrors - Error matrix for L3TrackParams.

L3TTracker base class user interface

This is a common interface developed for L3 track finding tools.

By obtaining a pointer to an L3TTracker in your code the specific track finder used can be altered simply by editing the trigger list.

```
virtual L3Return DoThisTool(L3TrackFitVector &tracks, double PtMin) = 0;

virtual L3Return DoThisTool(L3ZRegion &region, L3TrackFitVector &tracks, double PtMin) = 0;

// Reconstruct the entire detector and store internally the found
//  tracks, to be retrieved with a call to "getFoundTracks()" below

virtual L3Return DoThisTool() = 0;

// Fills L3TrackFitVector with "const L3TrackFit *" corresponding to
//  *all* currently found tracks from all previous reconstruction
//  requests.
virtual void getFoundTracks(L3TrackFitVector &tracks)=0;
```

Tracks are returned to the user via an L3TrackFitVector which contains const pointers to L3TrackFit objects. In all cases the returned list is owned by the caller and is never cleared by the tracking tool. L3TrackFitVector is simply a typedef:

typedef l3vector<const L3TrackFit*> L3TrackFitVector;

Information access methods of L3TrackFit

```
////// Parameters and Errors
  const L3TrackParams& getConstParams() const;
  const L3TrackErrors& getConstErrors() const;
  float getParam(L3TrackParams::Index i) const;
  float getError(L3TrackParams::Index i) const;
  float getError(L3TrackParams::Index i,L3TrackParams::Index j) const;

 ////// Swimming the fit
  float phi_at_r(float r) const;
  float z_at_r(float r) const;
  void xy_at_z(float z,float &x,float &y) const;

  /////// Chi-squared
  float getChiSq() const;      // Total chi-square
  float getChiSqXY() const;    // chi-square for circle fit only.
  float getChiSqZ() const;     // chi-square for line fit only.

/////// Fit validity
  bool isValid() const;
  bool isValidZ() const;
  bool isValidXY() const;

  /////// Number of hits
  int nHitsZ() const;
  int nHitsXY() const;

 // standardized index to array of parameters
  enum Index { PTINV,PHI,Z,DCA,TANL,RINV };
```

NB. Indices for accessing *errors* are only valid for 1st 5 parameters listed in enum Index.

See

www-d0/~beuselin/d0_private/trackpar.ps

for definition of track parameters.

Definition of Track Parameters

RINV is S1/R where R is the radius of the circle in the (x,y) projection and S1 is the sign of the track rotation. +1 indicates the track turns anti-clockwise, -1 indicates the track turns clockwise.

PHI is the azimuthal angle of the track direction at the point of closest approach to the origin in the (x,y) plane (the DCA point).

TANL is dZ/dSxy, the pitch of the helix. Sxy refers to the distance traversed around the arc in the (x,y) plane measured from the DCA point.

DCA is S2*|DCA|. |DCA| is the distance of closest approach of the track helix to the origin in the (x,y) projection. S2 is the sign of the angular momentum of the track about the origin at the DCA point.

Z is the Z coordinate of the helix at the DCA point.

PTINV is the signed inverse transverse momentum, computed from RINV on the assumption of a uniform 2 Tesla magnetic field, parallel to the Z axis.

Example RCP file for tests

L3fCFTTester.rcp

```
string InterfaceName = "process"
string Interfaces = "generator decide builder process dump runInit jobSummary"
string Flow = "generator decide builder process dump”

string Packages = "read  ftunp pack l3cft"

RCP read = <l3ftrack_cft ReadEvent>
RCP ftunp = <ft_unpdata ftunpdata>
RCP pack = <l3fcftunpack PackReco_FT>
RCP l3cft = <l3ftrack_cft l3fCFTTestPkg>

int DumpPeriod = 1
```

L3TCFTTrack uses raw data unpack tool.

In order to read older MC production files a raw data chunk can be created on the fly by using the packing processes shown in red.

These processes should be removed if data already contains raw data chunk.

Unfortunately ftunp has been broken for many releases now!

Example Trigger List Entries

Trigger list entry for L3TCFTTrack

```
CFTUnp L3TCFTUnpack(tooltype=unpack)
CFTTRACK L3TCFTTrack(tooltype=data, UNPACK=CFTUnp, ptcut=0.5)
```

The only user variable trigger list parameter is ptcut which determines the minimum pt tracks that will be found.

Higher values may be specified in calls of DoThisTool methods for trackers.

Trigger list entry for L3TGlobalTracker

```
SMTUnpack L3TSmtUnpack
CFTUnpack L3TCFTUnpack
GlobalTracker L3TGlobalTracker(CFTUnpack=CFTUnpack,SMTUnpack=SMTUnpack,tooltype=data)
```

Example Code for Tool Use

For an example of using the available trackers look at the integrated test code in

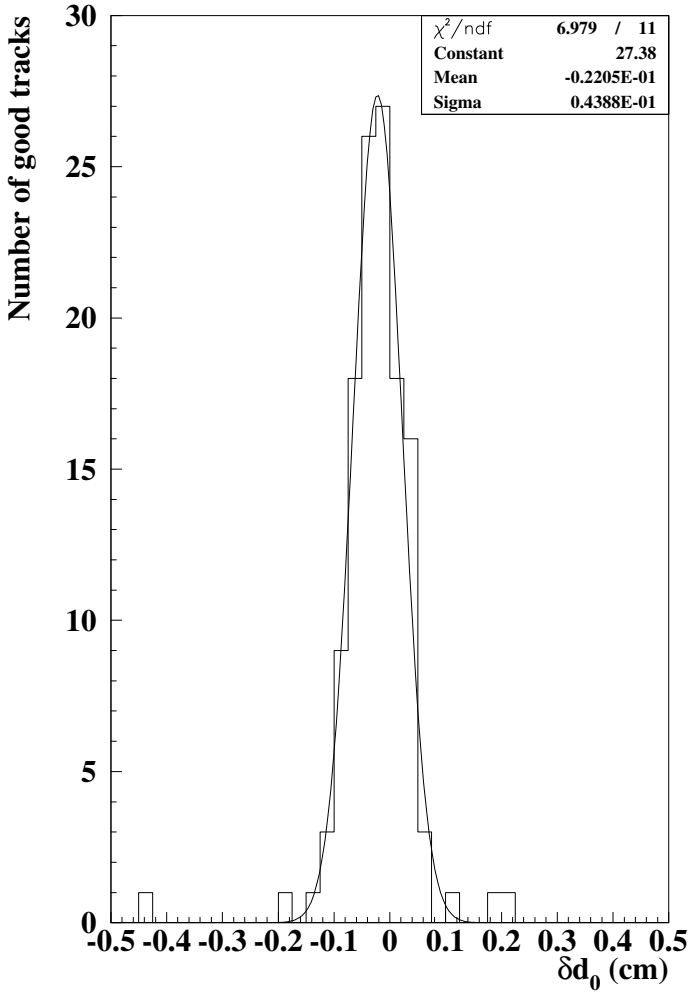
l3ftrack_cft/test/l3fCFTTester.cpp for L3TCFTTrack or

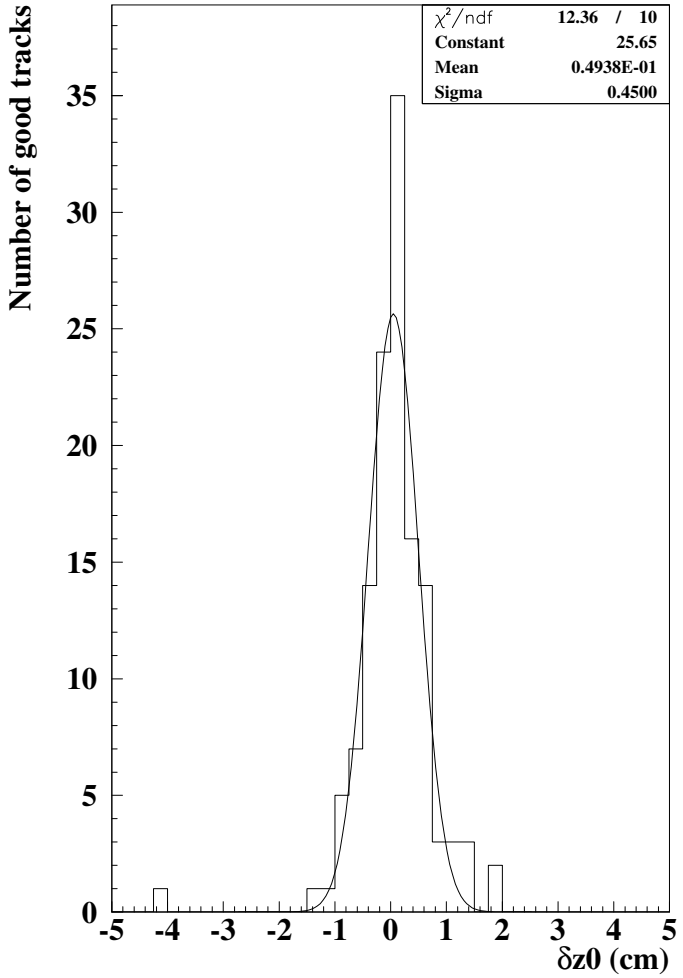
l3ftrack_global/test/l3fGlobalTrackerTester.cpp for L3TGlobalTracker.

```
Result l3fCFTTestPkg::processEvent(Event& evt)
{
  unpackTool::setEvent(&evt);

  L3TrackFitVector tracks;
  _cftTrack->DoThisTool();
  _cftTrack->getFoundTracks(tracks);

  L3TrackFitVector::const_iterator l3itr;
  for (l3itr=tracks.begin() ; l3itr!=tracks.end(); ++l3itr)
  {
    cout << **l3itr << endl;
  }

  //ScriptRunner normally handles the resets
  ((bTool *)_cftUnpack)->Reset();
  ((bTool *)_cftTrack)->Reset();

  return Result();
}
```

Very abbreviated code snippet from l3ftrack_cft integrated test.

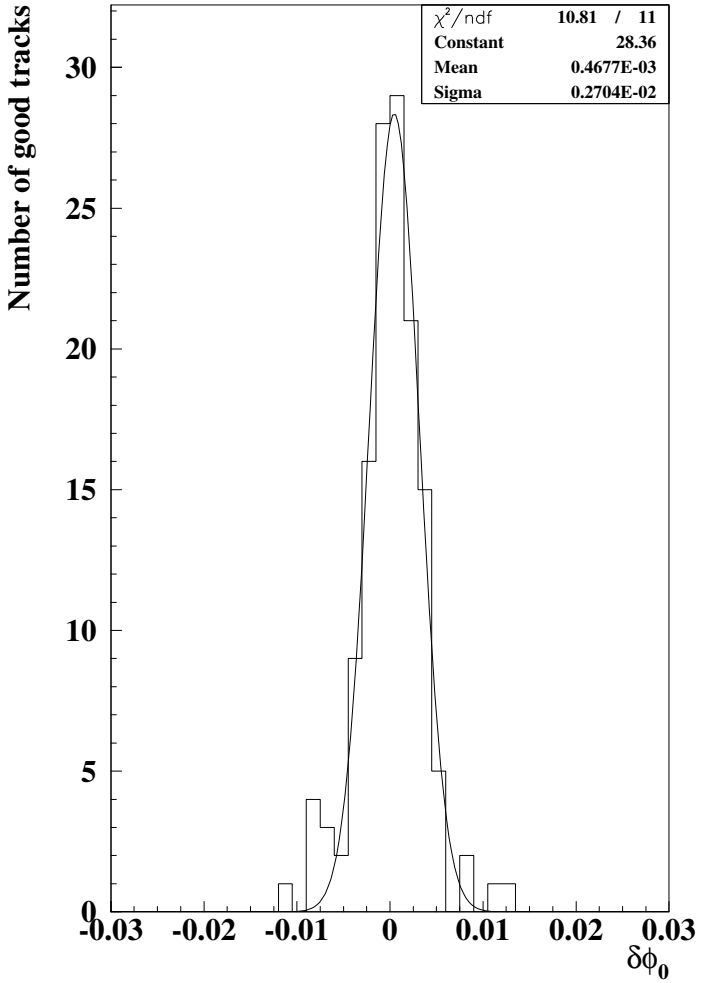
Example Resolution plots for Z->μμ

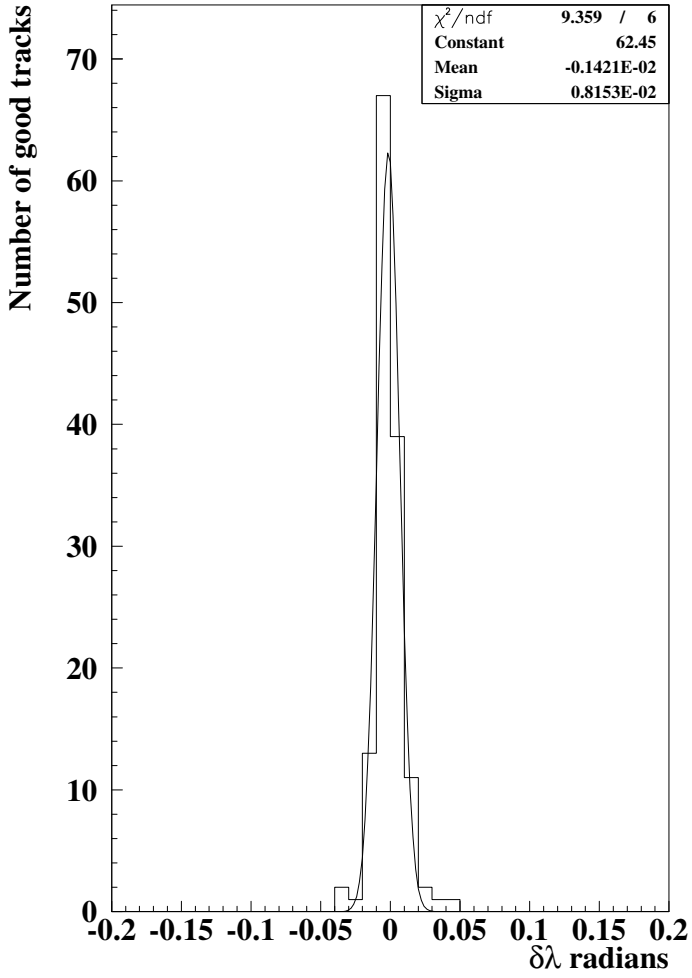


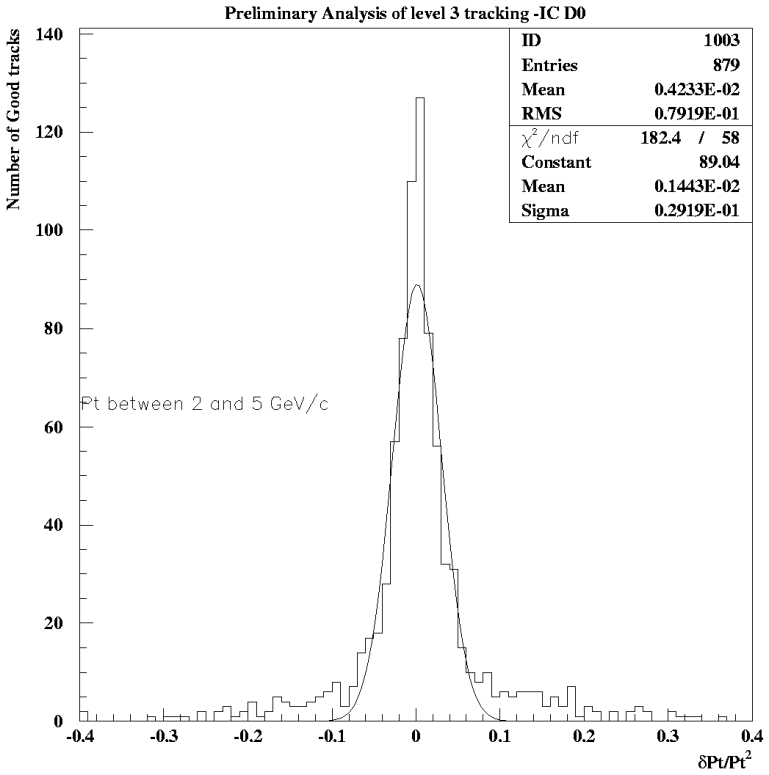
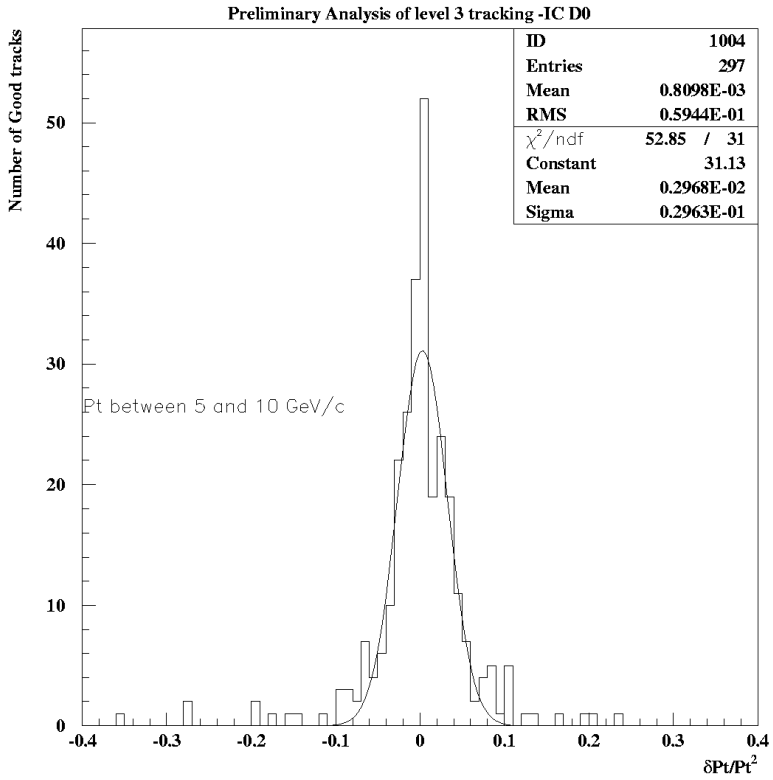


L3TCFTTrack

Example Resolution plots for Z->μμ





L3TCFTTrack

Top events (no min bias)
100 events