Offline vertex reconstruction

T. Speer

Brown University

Data and analysis flow tutorials Session 3: Distributed data

Introduction

- Vertex Reconstruction:
  - Vertex Finding: Identification of vertices and assignment of tracks to vertices, with possible estimate of vertex position
    - Primary vertex reconstruction
    - Vertex finding in Jets
  - Vertex Fitting: Most precise estimate of the vertex position and track parameters at vertex from a set of tracks
- Only primary vertex search is part of Standard Sequence and stored
- Further searches (secondary) have to be done by the user, selecting tracks of interest
  - It could be part of other sequences, e.g. conversions, V0 search, etc
- Documentation:
  - SW guide on Vertex reconstruction: SWGuideVertexReco
  - Workbook on Vertex reconstruction: WorkBookVertexReco
- HyperNews: hn-cms-btag

The vertex objects

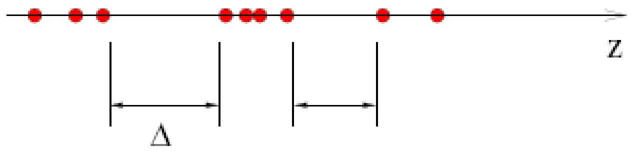
- The basic, persistent, object as seen from the used is the reco::Vertex:
  - Documentation: dOxygen, cvs, lxr
  - Position, uncertainty
  - chi2, ndof of the fit
  - Reference to tracks used in the fit
  - Weight of the tracks ($0 \leq w \leq 1$)
  - Refitted tracks, if refitting done
  - Flags on the validity/type of vertex:
    - isValid
    - isFake: whether it is made from tracks, or from BeamSpot
- If the user performs a vertex fit/search himself, he may use also the TransientVertex, which provides more information
  - It can be casted automatically to a reco::Vertex
  - Documentation: dOxygen, cvs, lxr

Primary vertex reconstruction

- Result of offline primary vertex reconstruction stored in both AOD and RECO:
  - Search using all tracks reconstructed in the tracker: generalTracks
  - BeamSpot: offlineBeamSpot
- Two collections:
  - OfflinePrimaryVertices: Default primary vertex reconstructions
  - OfflinePrimaryVerticesWithBS: Primary vertex reconstructed, imposing the offline beam spot as a constraint in the fit of the vertex position.
- We advise for now to use the OfflinePrimaryVertices collection
- If no reconstructed vertex is found in an event:
  - A vertex based on the beam-spot is put into the event
  - flag isFake() is set to true
  - Contains no tracks, chi^2 =0, ndof = 0, and the
- Documentation: AOD page, SWGuide, cvs

Primary vertex reconstruction

- Track selection
  - tracks compatible with the beam-line: distance of closest approach $< 5\sigma$
  - Minimum number of hits (7, of which 2 in pixel)
- Cluster tracks according to their z-coordinate at the point of closest approach



  - split clusters where $\Delta > 1$mm
- Fit tracks of a cluster to a common vertex (default: Adaptive Vertex Fit), with or without beam-line constraint
- Final cleaning
  - distance of vertex to beam-line < 500 (fit w/o BeamSpot) /200μm (fit w. BeamSpot)
  - vertex fit χ^2 probability > 1%
- Sort vertices by Sum (p_T^2)

Further vertex reconstruction

- Further vertex reconstruction can be done by user
  - Analysis dependent
  - Selection of tracks
  - Search or fit of vertex
- Several algorithms available
- Various tools available:
  - Distance between tracks, PCA, etc: (dOxygen)
  - Vertex-compatibility, 3D or 2D (dOxygen): Distance, compatibility, signed distance

Vertex reconstruction algorithms

- Several algorithms available:
- VertexFitters: SWGuide
  - Kalman Filter: LSM fitter
  - Adaptive Vertex Fitter: soft-assignment, iterative, re-weighted LS fit
  - TrimmedKalmanVertex Fitter: hard-assignment, iterative LS fit
  - Gaussian-Sum Filter: Gaussian mixture of *pdf*s
  - Adaptive Gaussian-Sum Filter
- Vertex finders: SWGuide
  - AdaptiveVertexReconstructor
  - TrimmedKalmanVertexFinder
  - MultiVertexFit: Concurent Multi-Vertex Fit
  - TertiaryTracksVertexFinder
- Kinematic fit: fit with constraints (SWGuide)

TransientTrack

- Default reco::Track not suitable for most higher-level algorithms (e.g. vertex, *b/τ* -tagging)
  - no access to magnetic field (no propagation!)
- Use Tracks through reco::TransientTrack
  - Doc: SWGuide, dOxygen
- In your application, build TT through TransientTrackBuilder:

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

- Gives access to different states, magnetic field, etc
- ReferenceCounted (à la TSOS)
  - Different concrete classes (TrackTransientTrack, GsfTransientTrack, TransientTrackFromFTS)
  - Same interface, done through the builder

Vertex Fitting and finding

- The object with which the user interacts is a VertexFitter or a VertexReconstructor:

```
KalmanVertexFitter fitter;
TransientVertex myVertex = fitter.vertex (vectorOfTransientTrack)
```

```
KalmanTrimmedVertexFinder finder;
vector< TransientVertex > vertices = finder.vertices (vectorOfTransientTrack)
```

- VertexFitter/VertexReconstructor:
  - controls all the steps of the vertex fit from the input of the initial information to the output of the estimated quantities.
  - different objects which perform the different steps are either hard-coded or have to be given at construction time.
- All algorithms have a default VertexFitter/VertexReconstructor which has reasonable defaults (components, parameters)
- Can also be used through ConfigurableVertexFitter / Reconstructor

Track refit

- Constraint of the Track parameters with fitted vertex (*smoothing*):
  - Only parameters/covariance at vertex are estimated, not along the track
  - Full track-to-track covariance matrix
  - Constraint done as part of vertex fit, at end of fit
  - Can be done by all fitting algorithms (see doc)

- Stand-alone class: SingleTrackVertexConstraint

```
class SingleTrackVertexConstraint {
  pair<TransientTrack, float> constrain
           (const reco::TransientTrack & track,
       const GlobalPoint& priorPos, const GlobalError& priorError) const;
}
```

  - The float is the smoothed track-χ^2 (the track-vertex compatibility)
  - Doc: SWGuide

Conclusion

- RECO & AOD: Offline primary vertex reconstruction
  - OfflinePrimaryVertices
  - OfflinePrimaryVerticesWithBS
  - If no reconstructed vertex is found in an event, beam-spot-vertex put in
  - Flags of true PV: isValid == true, isFake==false
  - Flags of BeamSpot PV: isValid == true, isFake==true
- Further vertex reconstruction can be done by user
- SW guide on Vertex reconstruction: SWGuideVertexReco
- Workbook on Vertex reconstruction: WorkBookVertexReco
- HyperNews: hn-cms-btag

ConfigurableVertexFitter

- Simplified usage through ConfigurableVertexFitter:
  - VertexFitter, that can be fully configured at runtime
  - Concrete VertexFitter used chosen at runtime through PSet

```
class ConfigurableVertexFitter : public VertexFitter {
  ConfigurableVertexFitter ( const edm::ParameterSet & );
```

- Documentation: SWGuide
- Fitters currently available:
  - KalmanFilter, Adaptive filter
- Examples PSet:

```
PSet vertexreco = {
  string fitter = "avf"
  double sigmacut =3.0
  double Tini=256
  double ratio=0.25 }
```

The configurables depend on the choice of the fitter!

ConfigurableVertexReconstructor

- Simplified usage through ConfigurableVertexReconstructor:
  - VertexReconstructor, that can be fully configured at runtime.
  - Concrete VertexReconstructor used chosen at runtime through PSet:

```
class ConfigurableVertexReconstructor :
        public VertexReconstructor {
  ConfigurableVertexReconstructor(const edm::ParameterSet&);
```

- Documentation: SWGuide
- Finders currently available:
  - Adaptive reconstructor, MultiVertexFitter, TrimmedKalmanVertexFinder

```
PSet vertexreco = {
 string finder = "avr"
 double primcut = 1.8
 double seccut = 6.0
 double minweight = 0.5 }
```

The configurables depend on the choice of the finder!