The Primary Vertex Reconstruction in ATLAS

G.Piacquadio (Uni. Freiburg)

K.Prokofiev (CERN)

A.Salzburger (CERN)

A.Wildauer (CERN)

Artemis school on calibration and performance of ATLAS detector

Munich, September 15-19, 2008

Outline

- Existing infrastructure.
- Approaches to the primary vertex finding in ATLAS.
- Algorithms for vertex fitting.
- Steering the primary vertex finding.
- Use of the beam information.
- How to configure your finder.
- The validation ntuple and its contents.

The current lecture is meant to be a helper for the following practical tutorial, but also to become a short user guide on ATLAS primary vertexing. If something is missing, wrong or remains unclear, please contact the authors: your feedback is much appreciated!

Reconstruction of primary vertices

- The reconstruction of primary vertices can be subdivided in two tasks
  - Primary vertex finding: association of reconstructed tracks to particular vertex candidate.
  - Primary vertex fitting: reconstruction of position of the primary vertex, covariance matrix, quantities related to the quality of the fit (e.g. χ^2, n.d.f., …), optional refit of parameters of incident tracks.

- Often these two stages are not distinguishable.
  - Both "Fitting after finding" and "Finding through fitting" approaches are implemented in ATLAS.
  - Vertex fitters and vertex finders are implemented as separate tools.
  - The implementation is based on the common Event Data Model and a set of abstract interfaces.
  - This allows an easy interchange of tools and adjustment of strategies..

Software infrastructure

Event Data Model: *Tracking/TrkEvent/VxVertex*, *VxMultiVertex* etc… Classes storing the information on reconstructed vertices and their relation to the incident tracks

Primary Vertex Finding: *InnerDetector/InDetRecAlgs/InDetPriVxFinder/* Main primary vertex finding tool, producing the container in the StoreGate.

InnerDetector/InDetRecTools/InDetPriVxFinderTool/ Various algorithms for primary vertex finding.

Vertex Fitters: *Tracking/TrkVertexFitter/TrkVertexFitters/* Top package containing algorithms for vertex fitting.

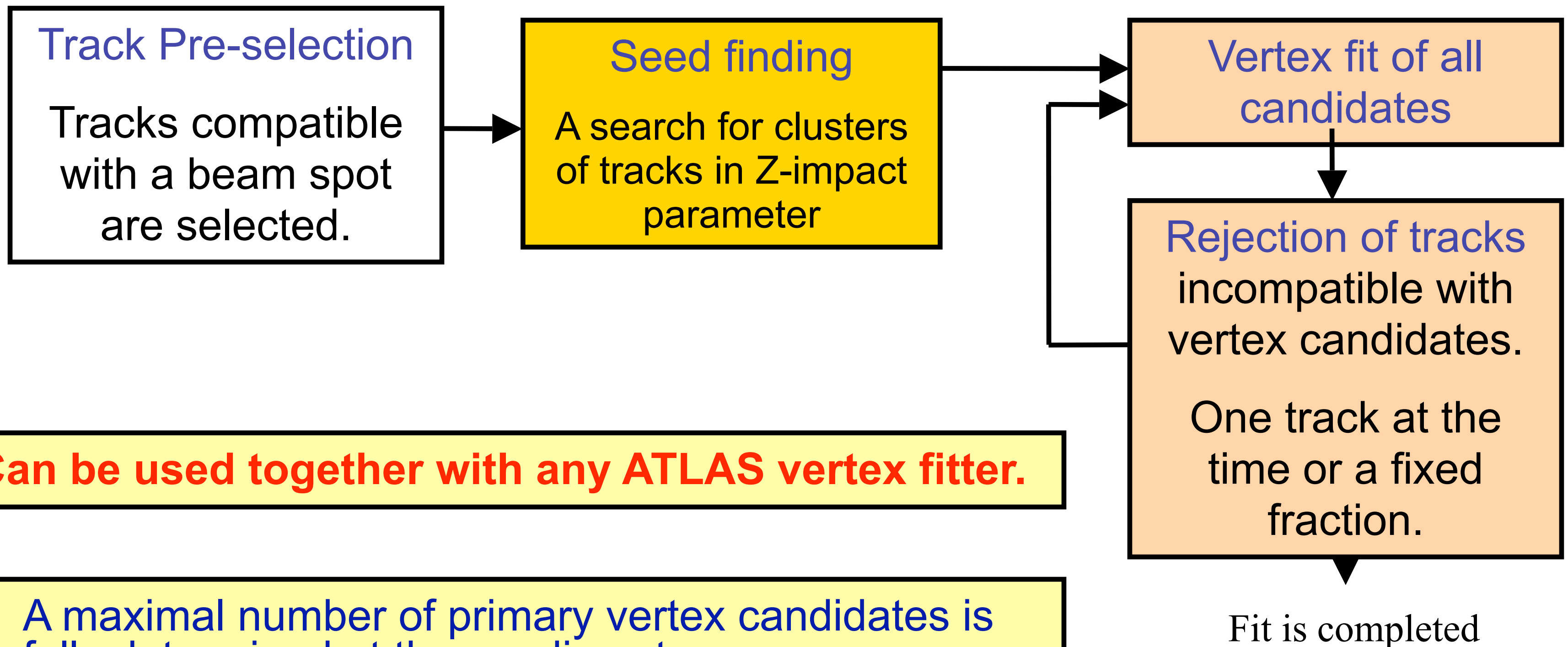
General Steering: *InnerDetector/InDetExample/InDetRecExample/jobOptions.py* and *ReadInDetJobOptions.py*

Validation code: *Tracking/TrkValidation/TrkVertexFitterValidation/*

RTT steering: *InnerDetector/InDetValidation/InDetVertexRTT/*

InDetPriVxFinder

(Fitting after finding approach)



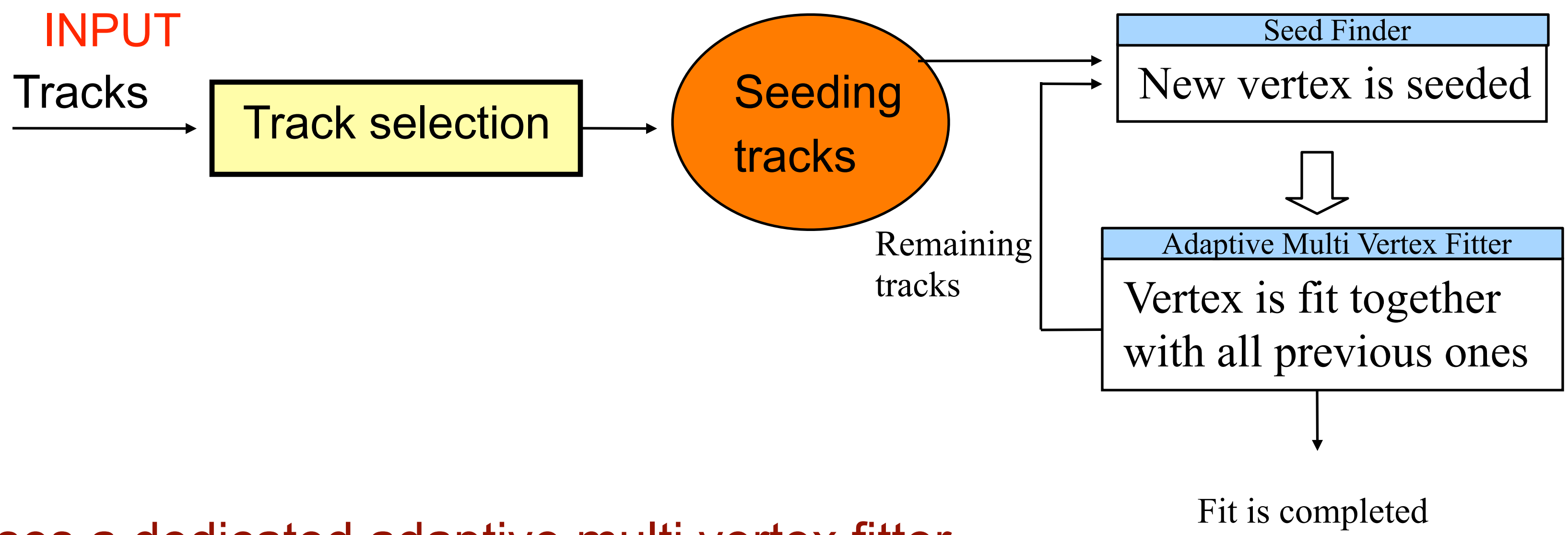
Can be used together with any ATLAS vertex fitter.

- A maximal number of primary vertex candidates is fully determined at the seeding stage.
- No possibility of re-using rejected tracks.
- The signal (tagged) primary vertex is selected according to the highest Σp_t of tracks.

AdaptiveMultiVertexFinder

(Finding through fitting approach)

- Adaptive multi vertex finder: default ATLAS reco. algorithm



- Uses a dedicated adaptive multi vertex fitter
  - Several vertices are fitted simultaneously, competing against each other in order to get a certain track assigned to them.
  - An annealing procedure is used: the assignement of tracks to vertices gets harder as the fit iteration number increases and the vertex position is known with more precision.
- The signal (tagged) vertex is selected according to the highest $\Sigma p_t^2/N_{trk}$

Algorithms for vertex fitting

- Billoir Tools package *(P.Billoir, S.Qian Nucl. Ins. and Meth. in Phys. Res. A311(1992) 139-150)*
  - The equations of motion of a charged particle in the magnetic field are approximated with their Taylor expansion in the vicinity of the vertex.
  - FastVertexFitter: the trajectories are approximated with straight lines in the vicinity of the vertex. No refit of the incident tracks is performed.
  - FullVertexFitter: The full parametrization of tracks is used, the refit of incident tracks is performed.
- Sequential vertex fitter *(R.Frühwirth Nucl. Ins. and Meth. 225(1984) 352)*
  - Implements a conventional Kalman filter for the vertex fitting.
  - A full analytical derivation of equation of motion is used.
- Adaptive vertex fitter *(R.Frühwirth et al. Nucl. Ins. and Meth. in Phys Res A 502 (2003) 699)*
  - An iterative re-weighted least square algorithm.
  - Down-weights tracks according to their compatibility to the vertex candidate.
  - The outliers are thus efficiently discarded.

What is stored inside a reconstructed primary vertex

VxCandidate

RecVertex

position, covariance matrix, χ^2, n.d.f.

vector<VxTrackAtVertex>

links to initial Tracks or TrackParticles

Since release 14.2.0, the refitted parameters of incident tracks are not stored in the VxCandidate anymore.

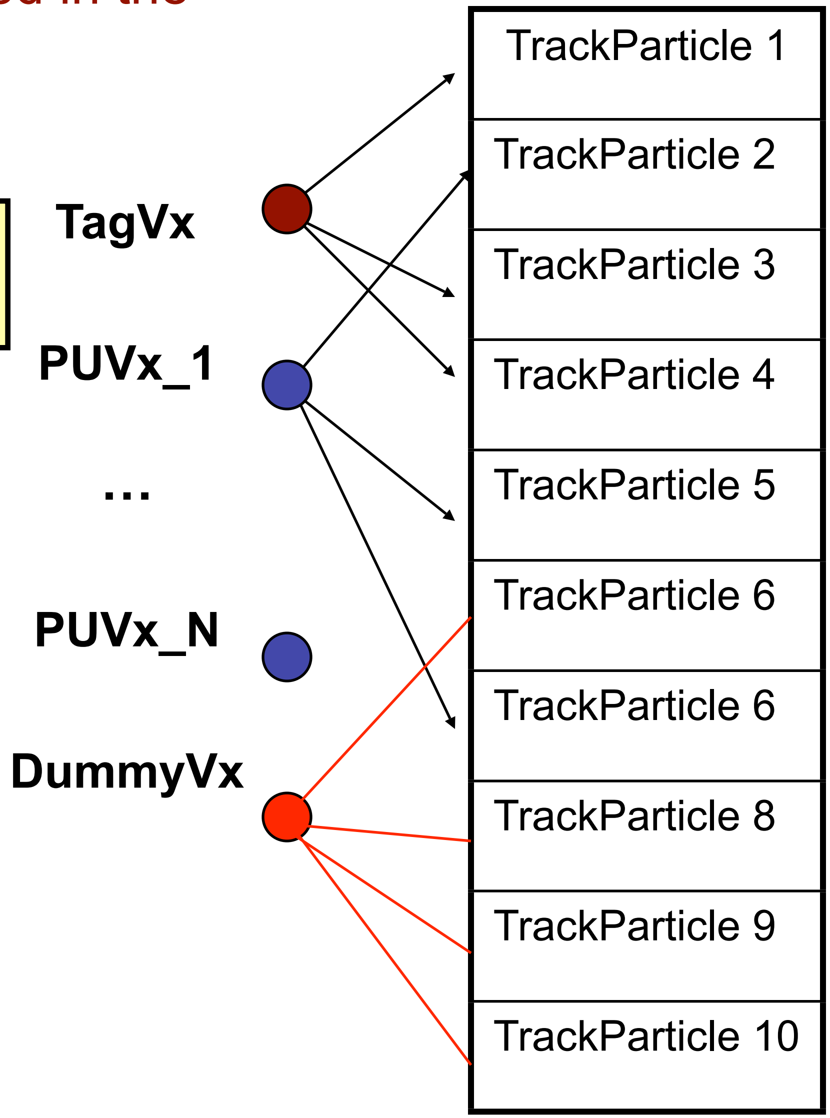
The re-fitting can be done by using a smoother from *TrkVertexFitterUtils* package.

The VxPrimaryContainer

The output of primary vertex finders is stored in the StoreGate using the VxContainer: *(New)VxPrimaryCandidate*

Order of VxPrimaryCandidate :
<Tagged PV, PileUp PV's, DummyVertex>

- The dummy vertex has the coordinates of a primary vertex, but no *Tack(Particle)'s* associated to it.
- If no primary vertex is found, only the dummy vertex is stored
- All the *TrackParticle's* not associated with tagged primary or pile up vertices point to the dummy vertex



Primary Vertexing Steering

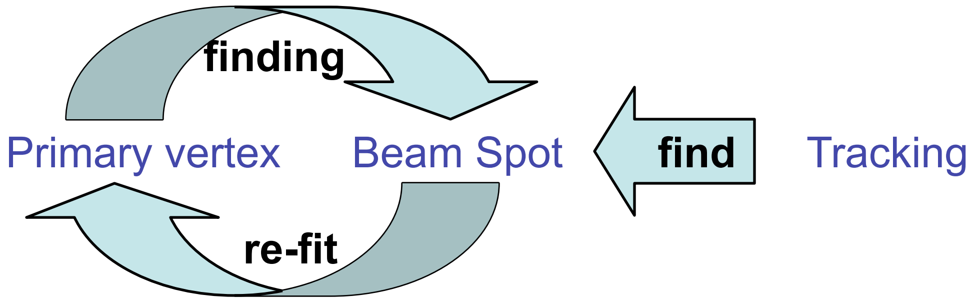
- *InnerDetector/InDetExample/InDetRecExample/*
  - *jobOptions.py* RDO-based reconstruction, produces primary vertices by default
  - *ReadInDetJobOptions.py* reading ESD or AOD data. To re-do the primary vertexing set *reDoPrimaryVertexing = True*

- Switching the primary vertexing strategy
  - Default algorithm is the InDetMultiAdaptivePriVxFinder.
  - *InDetFlags.primaryVertexSetup = "DefaultFast(Full)Finding"* switches on the InDetPriVxFinder + Billoir fast (full) vertex fitter.
  - …= *"DefaultKalman(Adaptive)Finding"* switches on the InDetPriVxFinder + Sequential (Adaptive) vertex fitter.
  - More possibilities available: VKalVrt etc…

- The use of the beamspot is controlled by a flag: *InDetFlags.useBeamConstraint=True*

Use of the beamspot

- Primary vertexing is connected to BS determination in two ways
  - As input to determine the BS (next to the track based approach).
  - As client: usage of BS as constraint improves primary vertex finding.



- Since 14.2.20 the vertexing can deal with different scenarios:
  - No beam spot available.
  - Beam spot available but larger uncertainty.
  - Beam spot available.
  - In the case no (uncertain) estimate is available, loose track pre-selection, 3D vertex seeding and 3D vertex finding are used.

Use of the beamspot

Currently implemented for *InDetAdaptiveMultiVertexFinder* only.

Reliable beam spot estimate is present?

NO → **Loose pre-selection, 3D seeding, 3D Vertex Finding**

YES → **3D is required?**

(Sperad of BS estimate is large?)

NO → **Pre-selection according to the Beam Spot, Z seeding, Z finding**

YES → **Pre-selection according to the Beam Spot, 3D seeding, 3D Vertex Finding**

Default case currently. → Pre-selection according to the Beam Spot, Z seeding, Z finding

The default case: The beam spot is used as constraint in vertex fit.

The primary vertex validation ntuple and what is inside

- The validation ntuple is produced by the *TrkValidation/ TrkVertexFitterValidation/PUVertexTest* algorithm.
  - Currently used for the RTT tests
  - To produce the validation ntuple during reconstruction, switch this flag in your jobOptions: *InDetFlags.doVtxNtuple = True.*

- The validation ntuple contains the VertexTree with following data entries:
  - Reconstructed signal vertex position (x,y,z).
  - Reconstructed signal vertex error (σ_X, σ_Y, σ_Z).
  - Reconstructed signal vertex χ^2 and probability.
  - Number of reconstructed vertices.
  - Number of tracks used in the reconstruction of the signal vertex.

The primary vertex validation ntuple and what is inside

- If the MC information is available in the data processed, the efficiency of the primary vertex reconstruction can also be estimated.
  - In the PUVertexTest algorithm the following flag should be set: McAvailable=True.

- The following Monte Carlo-based information is stored in the ntuple.
  - Position of the simulated signal primary vertex (x,y,z).
  - Number of simulated primary vertices.
  - Number of charged particles originating from the simulated primary vertex.
  - Number of reconstructed fake tracks (those not passing the truth matching)
  - Number of reconstructed in-lying (those having the corresponding GenParticle produced in the primary interaction) and out-lying tracks.
  - Number of reconstructed outlying tracks originating from the pile up.
  - Sum of weights used in the signal primary vertex fit for in- and outliers and fakes of all types.

Analysis of the Validation Output

- The analysis of the validation output can be performed using the following root script:

- If the MC information is available, the distributions of residuals of coordinates and corresponding pulls are produced.
- The efficiency of finding the signal primary vertex is calculated using two approaches.
  - Geometry-based: the simulated primary vertex is within 5mm from the reconstructed one.
  - Purity-based: The purity of the primary vertex is positive.

- Primary vertex purity
  - A quantity showing whether the dominant part of track participated in the vertex fit can be associated with MC signal inlying particles:

$$Pr = \frac{\sum w^{inliers} - \sum w^{fakes} - \sum w^{outliers} - \sum w^{pil\_up}}{\sum w^{all\_tracks}}$$

where w is the weight of a track in the vertex fit.

Practical session information

- Follow the instructions given on the tutorial Wiki page:
  https://twiki.cern.ch/twiki/bin/view/Atlas/VertexingTutorialMunich