Progress report on the VERTIGO/RAVE Vertex Reconstruction Toolkit

Winfried Mitaroff and
Wolfgang Waltenberger
Institute of High Energy Physics
Austrian Academy of Sciences
Nikolsdorfer Gasse 18
A-1050 Vienna, Austria, Europe

A proposal has been made for the design and implementation of a detector-independent Vertex Reconstruction Toolkit and Interface to Generic Objects (VERTIGO). It aims at re-using existing state-of-the-art algorithms for geometric vertex finding and fitting by both linear and robust estimation methods; kinematic constraints will also be included for the benefit of complex multi-vertex topologies. The design is based on modern object-oriented techniques. A core (RAVE) is surrounded by a shell of interfaces and a set of analysis & debugging tools. The implementation follows an open source approach and is easily adaptable to future standards.

Reminder of motivation and goals

Offline data reduction chain

- The early stages – local pattern recognition, track search and track fitting – are highly detector-dependent;
- whereas the next stage – vertex reconstruction (finding and fitting) – is almost fully detector-independent.
- Vertx fitting with kinematic constraints may be subject to the requirements of a subsequent physics analysis.

Why looking for a toolkit ?

- Geometric vertex finding and fitting must not compromise the high spatial resolution of modern vertex detectors.
- This goal can be achieved by new, sophisticated methods beyond the traditional least squares or Kalman filter estimators, using robust, non-linear, mostly adaptive algorithms – like the deterministic annealing filter (DAF).
- It is not desirable for each new detector to re-code vertex reconstruction software from scratch – provided there exists an adequate, reliable and easy-to-use TOOLKIT.

A good point to start from

- In an initiative called "The RAVE Manifesto", one author (WW) proposes taking out vertex reconstruction from the CMS general reconstruction software ORCA, thus providing the basic stock for the core of such a toolkit.
- However, the core must be complemented by flexible interfaces and a modular set of analysis & debugging tools.

General design considerations

The RAVE core

- Collection of the best algorithms available for vertex reconstruction – **finding, fitting, kinematics**.
- Starting with the packages developed by CMS, but open for entries by other parties (e.g. ZVTOP).
- Code to be based on C++ and HEP-wide (not CERN-specific) OO standards. Well-designed abstraction for interfacing with the outside world will be the key of success.

Shell of interfaces

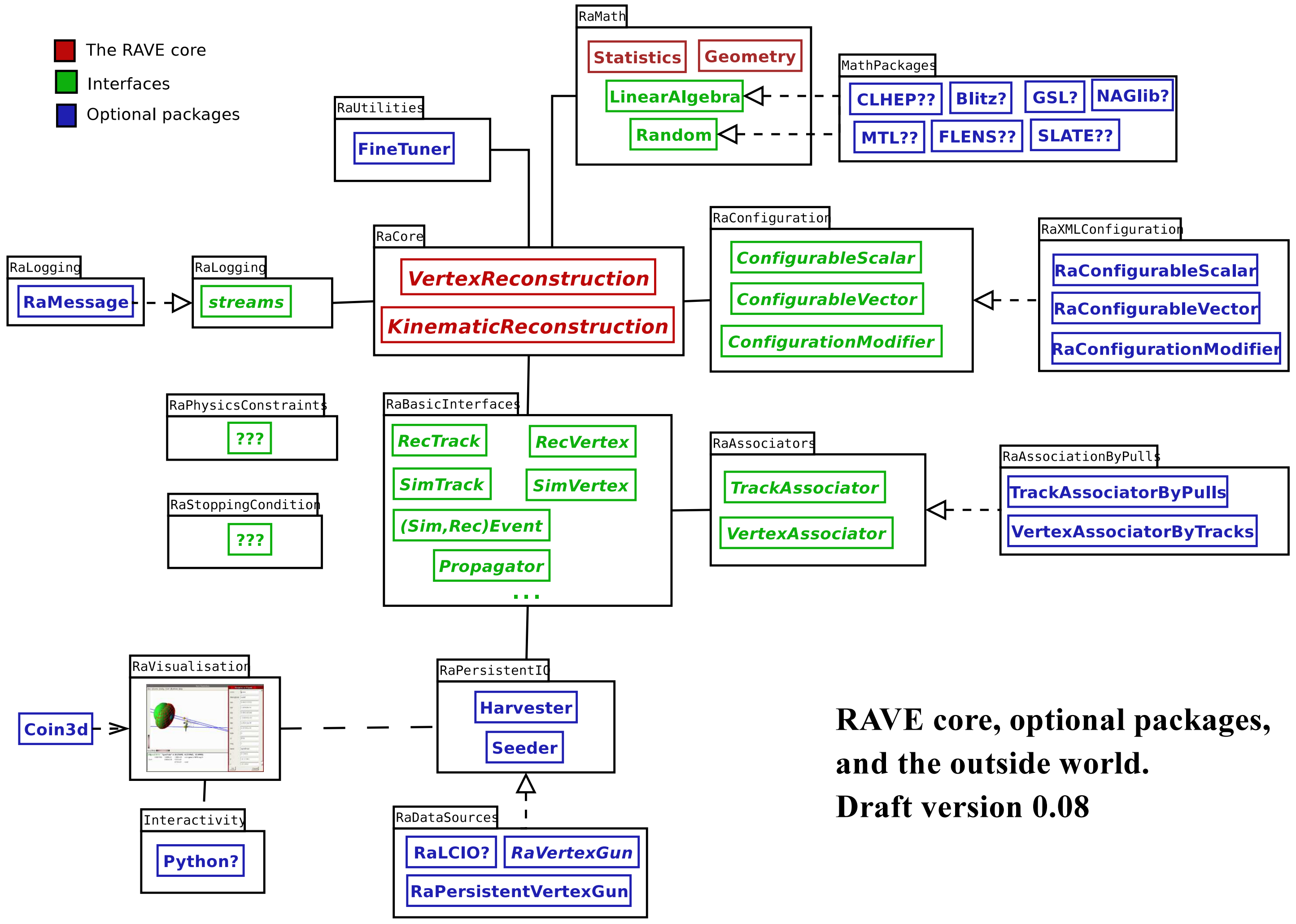
- Access from/to the outside world will exclusively proceed via a shell of interfaces surrounding the core.
- These interfaces make use of adaptors in order to keep a high level of abstraction.

Analysis & debugging tools

- Optional packages, containing those parts of code which might be helpful without being strictly necessary.
- Prototypes of a few packages have already been written:
  - Framework for a stand-alone realisation of VERTIGO,
  - DataSources ("vertex gun", interfaces to LCIO etc.),
  - Persistency storage solution ("data harvester" concept),
  - Visualisation tool (based on COIN3D and PYTHON),

  but much more work is still to be done.
- Extensive use of open standards will minimize the burden of development for this part of the toolkit.

VERTIGO overall design – draft 0.08



RAVE core, optional packages, and the outside world. Draft version 0.08

News on the RAVE core (1)

RAVE = "Reconstruction Algorithms for Vertices"

The list of candidate core algorithms is, at present, dominated by algorithms for the CMS offline reconstruction software (ORCA). Contributors are: *R. Frühwirth, W. Waltenberger (HEPHY Vienna), J. d'Hondt, P. Vanlaer (IIHE Brussels), E. Chabanat, N. Estre (IN2P3 Lyon), K. Prokofiev, T. Speer (Univ. Zurich).*

Vertex fitting packages

- `LinearizationPointFinder` :
  Since all fitting is performed by linearized model equations (mapping "parameter space" onto "virtual measurement space"), a reliable guess is needed for the expansion point. Implemented are two categories of algorithms:
  - A 3d mode finder (HSM, LMS, FSMW or ISMS) on the "crossing points" (mean of the 2 points of closest approach) of some or all track pairs;
  - A robust fitter (unweigthted Trimmer) on the "apex points" (as found by MTV, see below) of all tracks.
- `LinearVertexFitter` (LVF) and `KalmanVertexFitter` (KVF):
  Both are least-squares estimators, implementing the Kalman filter/smoother method. They differ in the linearization onto the virtual measurement space (standard vs. perigee track parameters).

  All least-squares estimators are not "robust" w.r.t. distorsions caused by "outliers" (i.e. measuremens with true errors much bigger than expected from the covariance matrix, or not belonging to the right set to be fitted). In case of multiple outliers, the combinatorial overhead of their detection becomes prohibitive.

News on the RAVE core (2)

Vertex fitting packages (cont'd)

- `TrimmingVertexFitter` (TVF):
  A simple robust estimator, iteratively identifying and discarding outlier tracks ("hard assignment").
- `AdaptiveVertexFitter` (AVF):
  A sophisticated robust estimator, iteratively downweighting the contribution of outlier tracks to the objective function ("soft assignment"). The extra weights w_i on the reduced residuals r_i are calculated by a Fermi function with cutoff parameter r_{cut}. In addition, an annealing schedule with decreasing "temperature" T may be introduced – `DeterministicAnnealingFitter` (DAF):

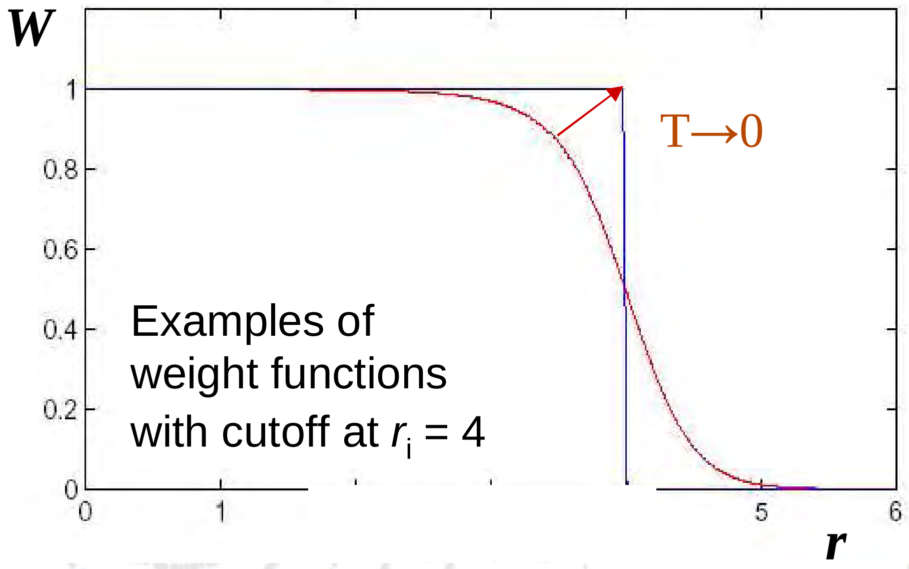
$$w_i(r_i, T) = \frac{e^{-r_i^2/2T}}{e^{-r_i^2/2T} + e^{-r_{cut}^2/2T}} = \frac{1}{1 + e^{(r_i^2 - r_{cut}^2)/2T}}$$

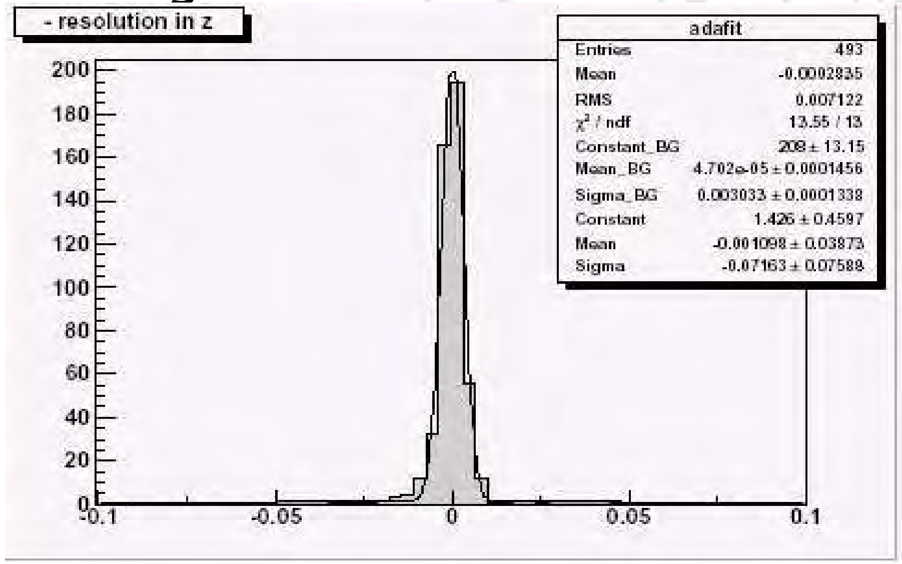
Iterations (index k, omitted above) start with $w_{i,1} = 1$ (least-squares). For $k > 1$, the $w_{i,k} = w_i(r_{i,k-1}, T_k)$, with $T_k \leq T_{k-1}$ defined by the annealing schedule. For $T \longrightarrow 0$, the Fermi function approximates the Heaviside function, and the assignment turns into a "hard" one ($w_i = 1$ or 0).

AdaptiveVertexFitter

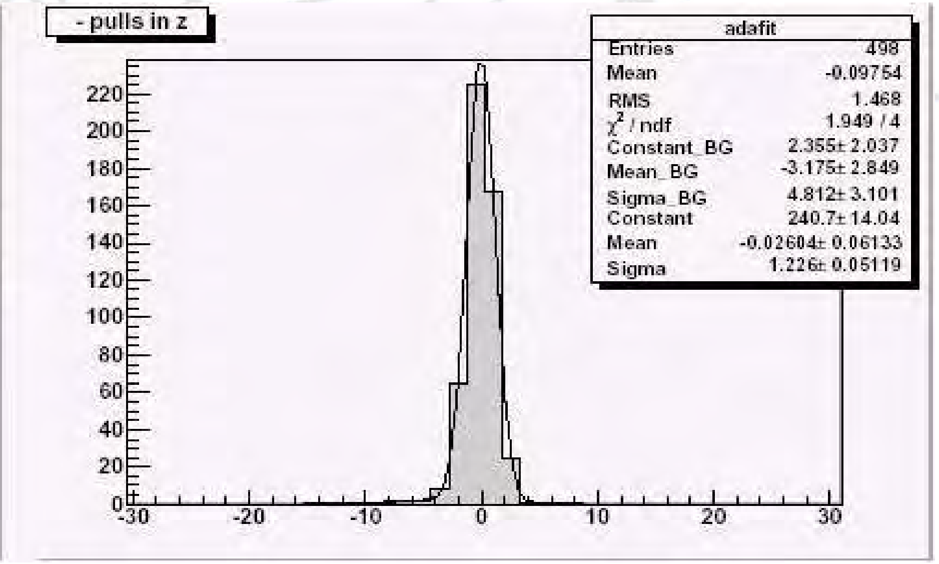
$$\hat{\beta}_{Adaptive} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(w_i \cdot r_i^2(\beta) \right)$$

- fast iterative, re-weighted LS fit with annealing.
- breakdown point: 0.5
- weight $w_i(r_i)$ of track i at iteration k depends on distance r_i to vertex at iteration k-1, and temperature
  - $w_i(r_i) \equiv$ assignment probability
  - r_i = reduced distance
- general-purpose algorithm
- user can choose cutoff on r_i and annealing schedule





```
adafit
Entries              493
Mean          -0.0002835
RMS             0.007122
χ² / ndf         13.55 / 13
Constant_BG      208 ± 13.15
Mean_BG    4.702e-05 ± 0.0001456
Sigma_BG    0.003033 ± 0.0001338
Constant      1.426 ± 0.4597
Mean      -0.001098 ± 0.03873
Sigma      -0.07163 ± 0.07588
```



```
adafit
Entries              498
Mean            -0.09754
RMS                1.468
χ² / ndf         1.949 / 4
Constant_BG    2.355± 2.037
Mean_BG       -3.175± 2.849
Sigma_BG       4.812± 3.101
Constant       240.7± 14.04
Mean        -0.02604± 0.06133
Sigma          1.226± 0.05119
```

News on the RAVE core (3)

Vertex fitting packages (cont'd)

- `MultiVertexFitter` (MVF):
  This robust estimator is a generalized AVF, simultaneously fitting n vertices by "soft assignment" of each track to more than one vertex. The extra weights w_{ij} on the reduced residuals r_{ij} w.r.t. vertex j are

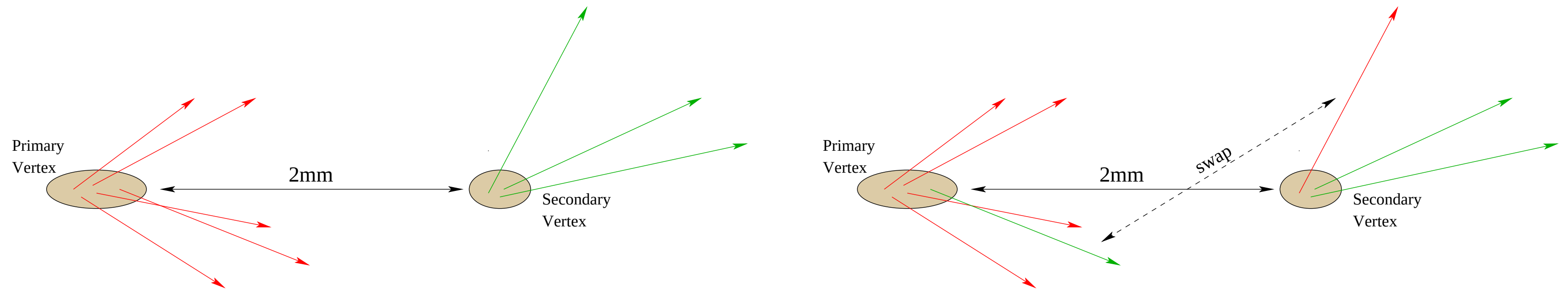
$$w_{ij}(r_{ij}, T) = \frac{e^{-r_{ij}^2/2T}}{\sum_{\ell=1}^{n} e^{-r_{i\ell}^2/2T} + e^{-r_{cut}^2/2T}}$$

- `GaussianSumVertexFitter` (GSF):
  If the errors of the track parameters are not Gaussian distributed (e.g. because of thin-layer Landau scattering or electron bremsstrahlung energy loss), they may be modelled as a weighted sum of several covariance matrices: one describing the core, and one or more the tails of the distribution.
  In this case, also the vertex parameter vector must be modelled as a weighted sum of several Kalman filters running in parallel, with each component of the track contributing to each component of the vertex. Thus, after adding track k having N_k components to the fit, the number M_k of vertex parameter components will be $M_k = N_k \cdot M_{k-1}$, i.e. exponentially increasing if not limited by some trimming.

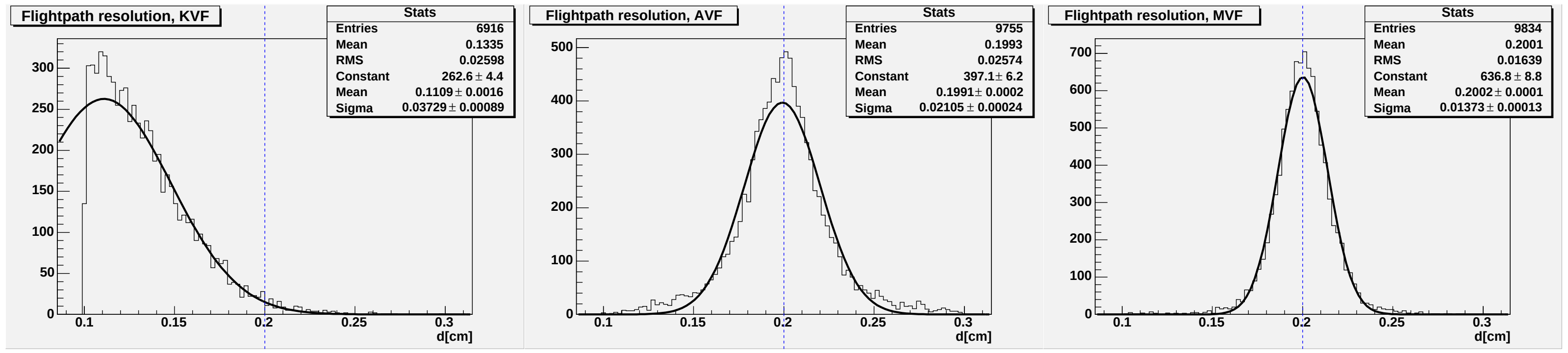
Performance of the MVF

Generated events: PV at (0, 0, 0) cm with 5 tracks, forming a jet of total momentum (0, 25, 25) GeV and openening angle 0.5 rad; one SV at (0, 0, 0.2) cm with 3 tracks, total jet momentun (-15, 0, 20) GeV and openening angle 0.5 rad. Track errors Gaussian and correctly described by the tracks' covariance matrices.

Thereafter, one PV track and one SV track were chosen to be "swapped", i.e. assigned to the wrong vertex.



The two track bundles, each containing one wrongly assigned track, were submitted to the KVF, the AVF and the MVF. Resolutions of the reconstructed decay lengths are shown below.



Stats (KVF)	
Entries	6916
Mean	0.1335
RMS	0.02598
Constant	262.6 ± 4.4
Mean	0.1109 ± 0.0016
Sigma	0.03729 ± 0.00089

Stats (AVF)	
Entries	9755
Mean	0.1993
RMS	0.02574
Constant	397.1 ± 6.2
Mean	0.1991 ± 0.0002
Sigma	0.02105 ± 0.00024

Stats (MVF)	
Entries	9834
Mean	0.2001
RMS	0.01639
Constant	636.8 ± 8.8
Mean	0.2002 ± 0.0001
Sigma	0.01373 ± 0.00013

News on the RAVE core (4)

Kinematic fitting packages

- `KinematicConstrainedVertexFitter` :

  A “global vertex fit”, based on least-squares minimization with Lagrangian multipliers for a flexible set of kinematic constraints (energy-momentum conservation, collinearity, back-to-back, invariant masses, etc).

- `KinematicParticleVertexFitter` :

  A “sequential fit”, interfaced to Kalman vertex fitting with perigee parameters extended by the particle’s mass m, with kinematic constraints applied sequentially after the vertex fit on a reconstructed decayed particle.

Vertex finding packages

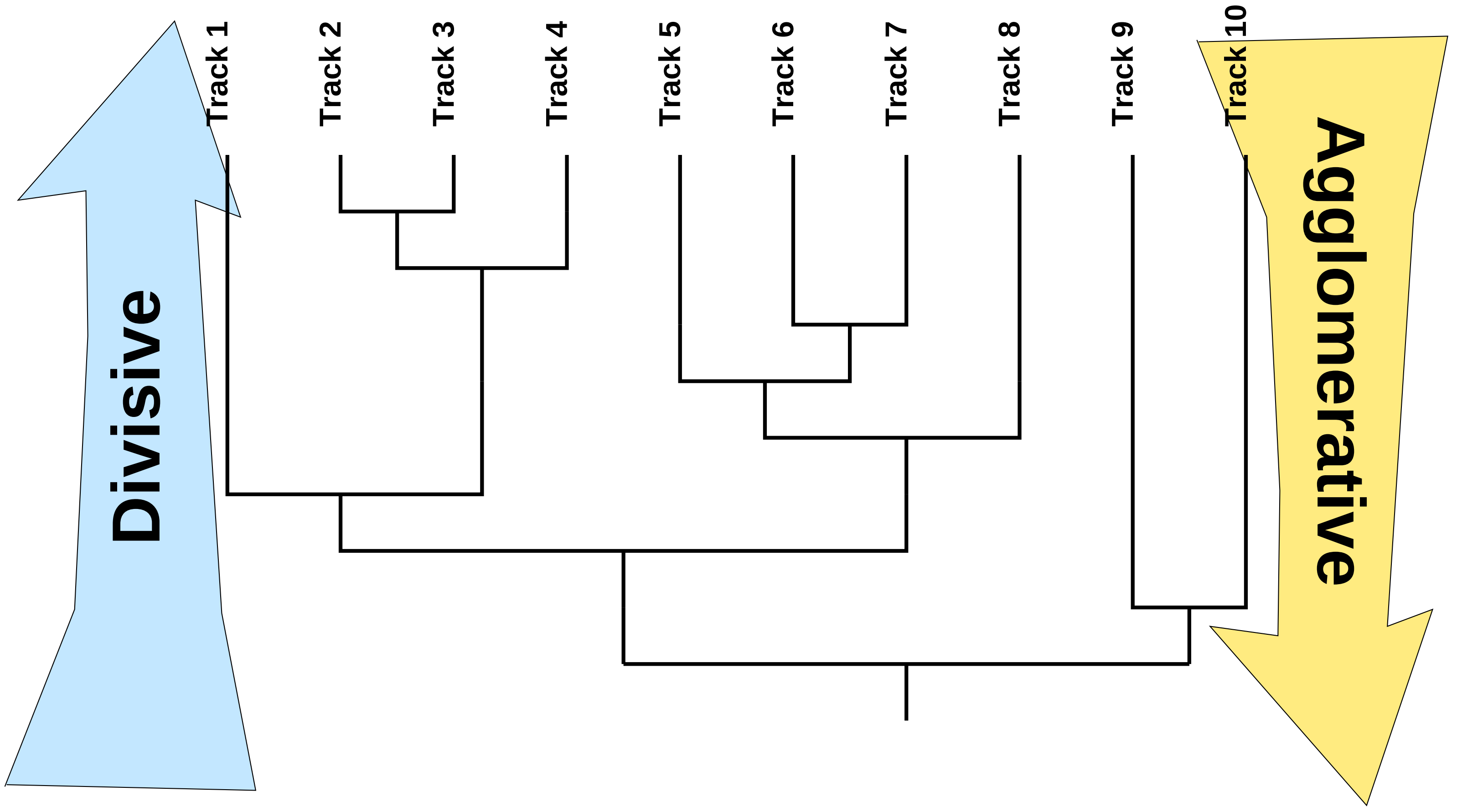
- `ApexPointFinder` :

  An “apex point” fully represents a track in the context of the vertex finding problem at hand. Starting from the set of points of closest approach w.r.t. all other tracks, the apex point finder may chose one of various algorithms (LMS, HSM, MTV, MSV, MAM) to yield a 3d-space point with error matrix on the track.

  These apex points are then used for vertex finding, e.g. by a hierarchic, agglomerative clustering method (see below). The apex point formalism may also be used as a linearization point finder (see above).

Hierarchic clustering example

Hierarchic clustering algorithms are either “divisive” or “agglomerative”, and may be visualized in a dendrogram:



News on the RAVE core (5)

Vertex finding packages (cont'd)

- `PrincipalVertexReconstructor` (PVR):
  A simple hierarchic, divisive algorithm based on a vertex fitter.
- `AgglomerativeVertexReconstructor` (AVR):
  A sophisticated hierarchic, agglomerative algorithm. Compatible clusters are iteratively merged by using a "cluster representative" (e.g. fitted vertex, apex point) and a metric for measuring their "distance" (since "single linkage" violates the triangle inequality, "complete linkage" is the metric chosen).
- `VectorQuantisationVertexReconstructor` (VVR):
  A non-hierarchic algorithm, based on unsupervised learning. The prototypes used are apex points.
- `SuperFinder` :
  Combines the solutions found by several vertex finders in order to find the best (or a new, better) solution.
- `ZVTOP` :
  So far, the only non-CMS package; developed by *D. Jackson (RAL & Oxford)* for Z^0 decays at SLC.

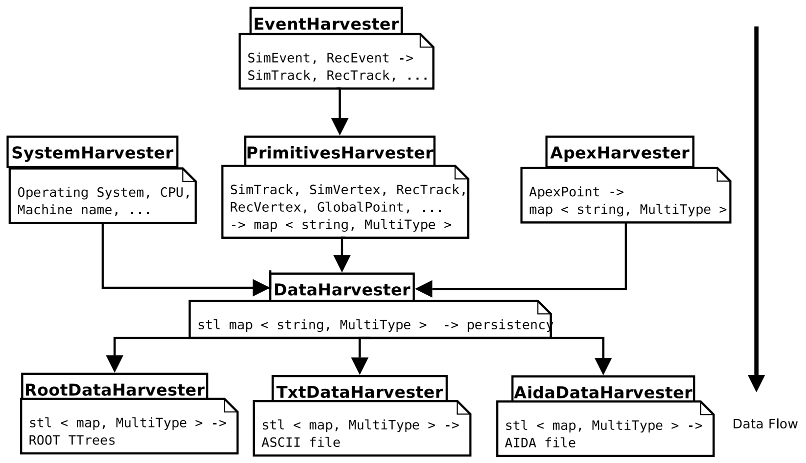
Documentation

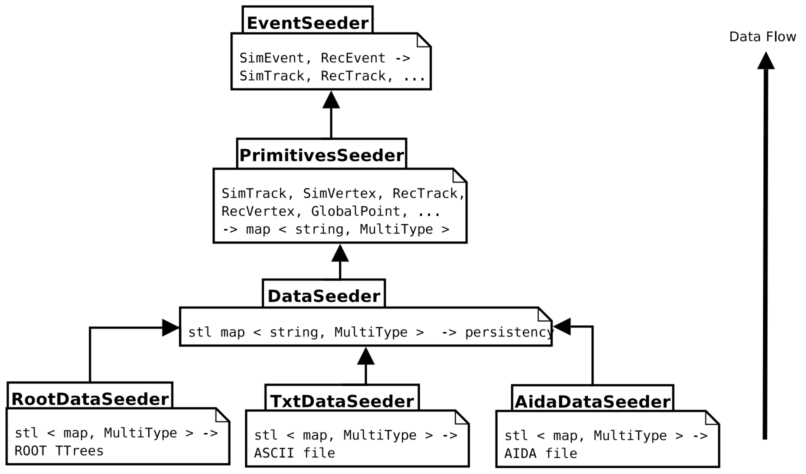
Documentation (based on Doxygen) of the algorithms, including information about their scope of application, will be provided. The proper choice of algorithms is also supported by the SKIN concept.

Analysis & debugging tools (1)

Persistency storage solution

- A persistency storage solution was originally realized on top of ROOT; it is currently being extended by more standard-compliant alternatives (AIDA and XML). `DataSources` include a "vertex gun", LCIO, etc.
- All I/O is handled through a "data harvesting" concept (which may possibly be integrated as front/end in AIDA): object $\longrightarrow$ STL map $\longrightarrow$ ASCII/ROOT/AIDA file (`Harvester`) and vice versa (`Seeder`).
- The STL mapping is heterogeneous: it handles int/double/string objects as `MultiType`.


EventHarvester
SimEvent, RecEvent -> SimTrack, RecTrack, ...
SystemHarvester
Operating System, CPU, Machine name, ...
PrimitivesHarvester
SimTrack, SimVertex, RecTrack, RecVertex, GlobalPoint, ... -> map < string, MultiType >
ApexHarvester
ApexPoint -> map < string, MultiType >
DataHarvester
stl map < string, MultiType > -> persistency
RootDataHarvester
stl < map, MultiType > -> ROOT TTrees
TxtDataHarvester
stl < map, MultiType > -> ASCII file
AidaDataHarvester
stl < map, MultiType > -> AIDA file
Data Flow


EventSeeder
SimEvent, RecEvent -> SimTrack, RecTrack, ...
PrimitivesSeeder
SimTrack, SimVertex, RecTrack, RecVertex, GlobalPoint, ... -> map < string, MultiType >
DataSeeder
stl map < string, MultiType > -> persistency
RootDataSeeder
stl < map, MultiType > -> ROOT TTrees
TxtDataSeeder
stl < map, MultiType > -> ASCII file
AidaDataSeeder
stl < map, MultiType > -> AIDA file
Data Flow

Analysis & debugging tools (2)

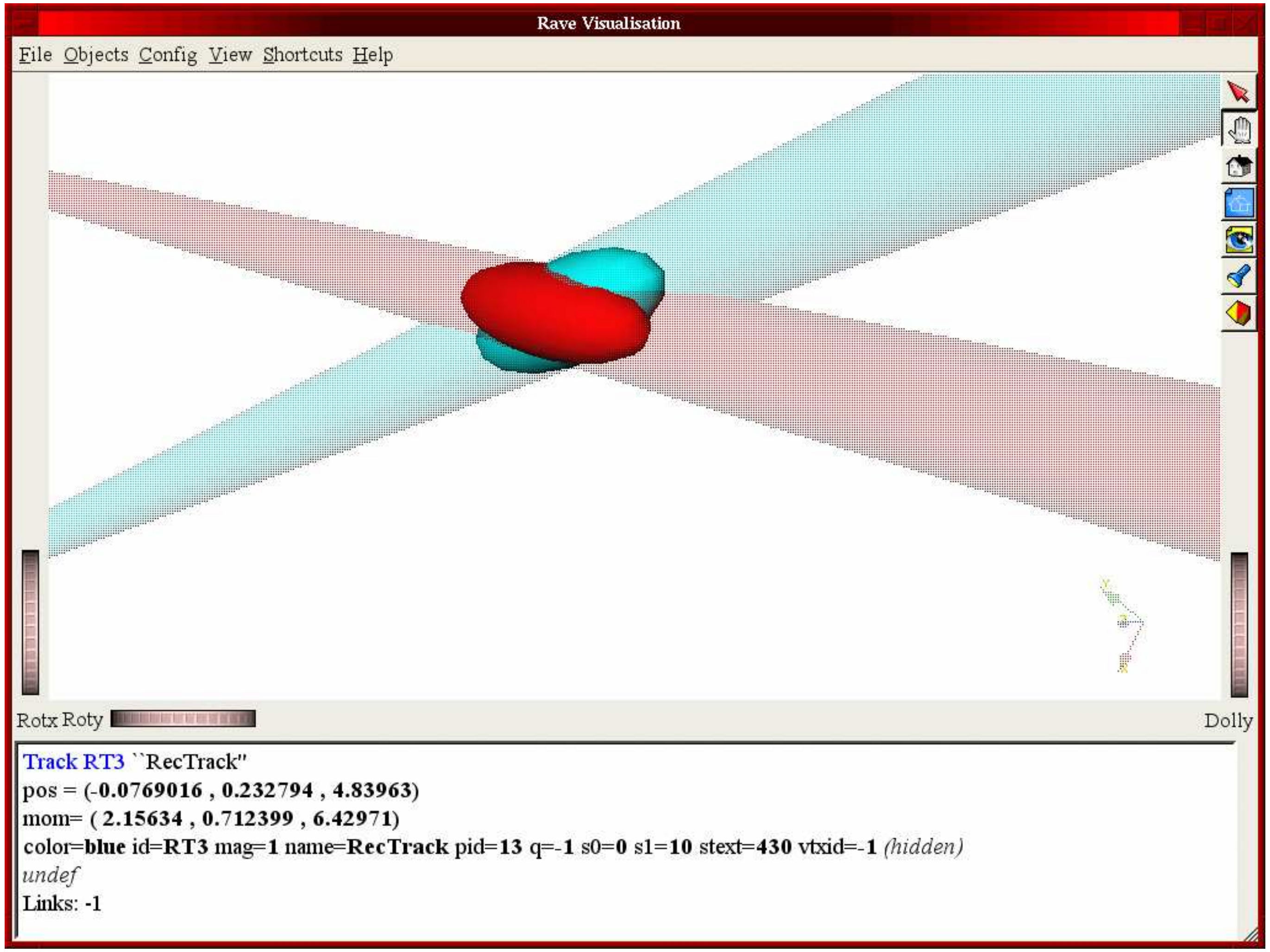
Visualisation tool

- Visualisation is deliberately kept simple for the sake of detector-independence. It follows the model-view-controller (MVC) paradigm and is based on COIN3D.
- Object data are accessed as `MultiType` STL maps:
  - at present only indirectly from an ASCII/ROOT/AIDA file through the `Seeder`;
  - in future maybe also directly through the `Harvester` and a TCP stream.
- Interactivity is at present limited to manipulators on graphic objects. The tool may later be augmented with full-scale interactivity, to be provided by PYTHON (or some other scripting language).

Example snapshots

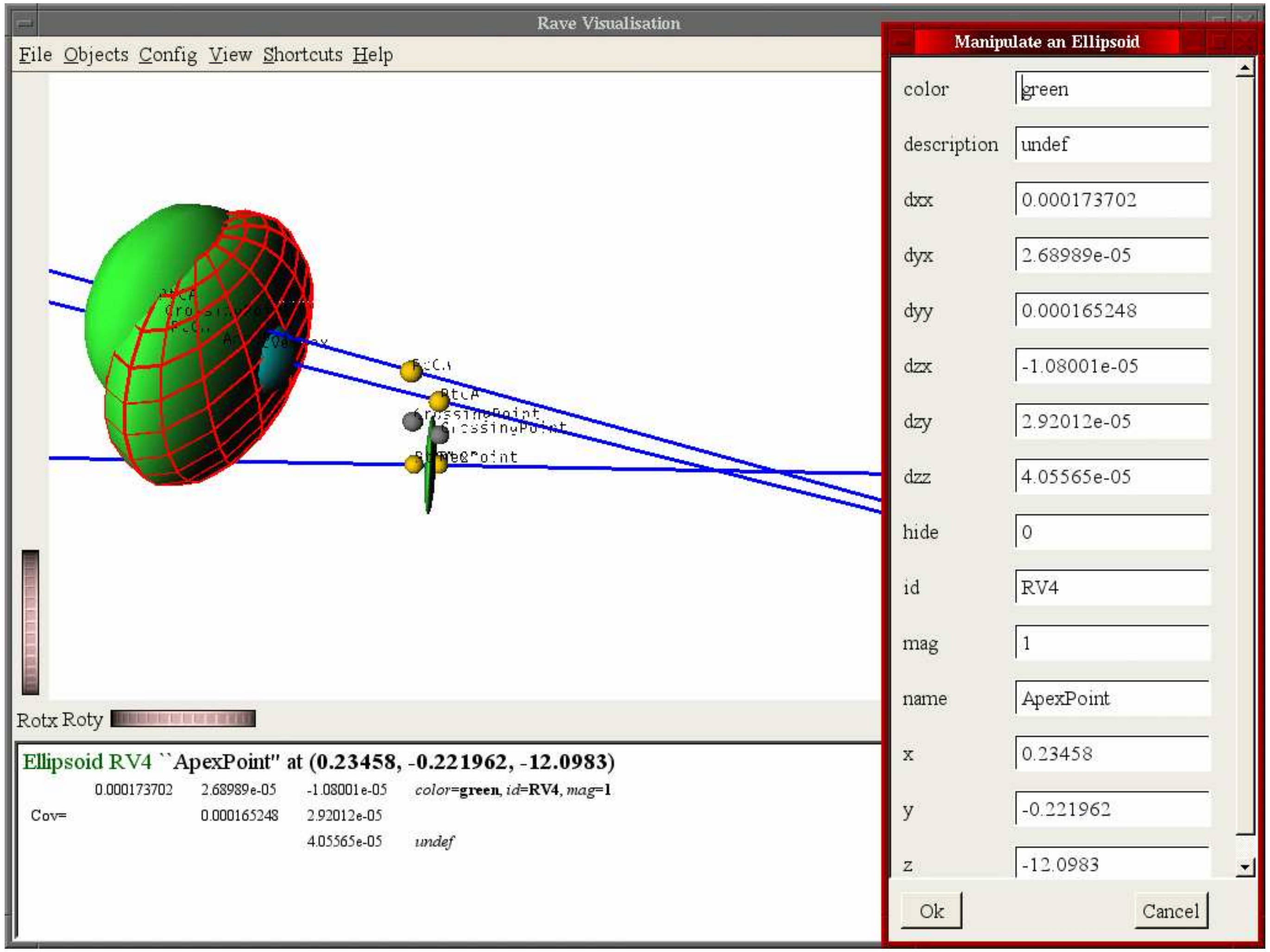
- **Example 1:**
  - Two reconstructed tracks (with their transversal errors shown as a tube), together with their apex points (with error ellipsoids).
  - The parameters of one track are displayed at the bottom.
- **Example 2:**
  - Three reconstructed tracks, together with two (out of three) triples of points of closest approach (yellow) and crossing points (gray); with their three apex points (with error ellipsoids, green); and with the fitted vertex position (with error ellipsoid, blue).
  - The parameters and covariant errors of one apex point are displayed at the bottom. The corresponding manipulator window is displayed on the right.

Visualisation tool – example 1



Track RT3 ``RecTrack''
pos = (**-0.0769016 , 0.232794 , 4.83963**)
mom= (**2.15634 , 0.712399 , 6.42971**)
color=**blue** id=**RT3** mag=**1** name=**RecTrack** pid=**13** q=**-1** s0=**0** s1=**10** stext=**430** vtxid=**-1** *(hidden)*
undef
Links: -1

Visualisation tool – example 2



color	green
description	undef
dxx	0.000173702
dyx	2.68989e-05
dyy	0.000165248
dzx	-1.08001e-05
dzy	2.92012e-05
dzz	4.05565e-05
hide	0
id	RV4
mag	1
name	ApexPoint
x	0.23458
y	-0.221962
z	-12.0983

```
Ellipsoid RV4 ``ApexPoint'' at (0.23458, -0.221962, -12.0983)
         0.000173702  2.68989e-05  -1.08001e-05  color=green, id=RV4, mag=1
 Cov=                 0.000165248   2.92012e-05
                                    4.05565e-05  undef
```

Analysis & debugging tools (3)

Math library package

- The choice of `MathPackages` (including linear algebra) is crucial for the efficiency and reliability of the toolkit.
- CLHEP appears to be the only choice freely available today, but there are serious doubts about its reliability.
- NAGlib is a reliable alternative, but may be too expensive for users outside of campus licence agreements.
- Generic (template) libraries would be our preferred choice. Candidates exist, e.g. MTL, GSL, FLENS, SLATE, BLITZ (all GPL-licenced); but none providing the full functionality required.

The SKIN concept

- Different experiments will use different sets of the optional packages. There are two alternatives:
  - the package is part of and is shipped with VERTIGO; or
  - the package is maintained by the particular experiment, and VERTIGO provides only the appropriate interface.
- An experiment-specific set of packages is called a SKIN. Examples are a stand-alone skin (called the "Framework"), CMS skin, ATLAS skin, TESLA skin, LCD skin, etc.
- Pre-defined skins may easily be selected by the user with `configure`.
- Maintenance and distribution of the toolkit is supported by a CVS repository at HEPHY Vienna.

Conclusions and outlook

Impact of the project

- This is (according to our knowledge) among the first large-scale attempts of refining a substantial part of reconstruction software into a detector-independent toolkit.
  - The idea of a detector-independent vertex library, based on the robust M-estimator and written in FORTRAN, was proposed by one author *(WM)* already in 1996.
  - Nowadays, this project is supported by recent advances in information technology, together with an overall increased IT maturity of the HEP user community.
- Interests in using the toolkit, once it will be released, have been expressed by CMS, ATLAS, LHCb, BELLE and Linear Collider (TESLA, LCD) collaborators. More to follow ?
- For track reconstruction, a general-purpose toolkit (RecPack) is currently being developed, following similar ideas, by *A. Cervera-Villanueva* et al. Possible synergies with our project (e.g. by data sharing, use of common interfaces, etc) are welcome and are actively pursued.

Manpower and funding

- So far, the project has been pushed by the enthusiasm of one author *(WW)*, with some help from a part-time graduate student, at HEPHY Vienna. External funding for a postdoc is being applied for.
- **Closer collaboration among the contributing labs is welcome and will be essential for success.**

Backup slides

Least square methods

$$\hat{\beta}_{LS} = \underset{\beta}{\text{argmin}} \sum_{i=1}^{n} r_i^2(\beta)$$

LinearVertexFitter

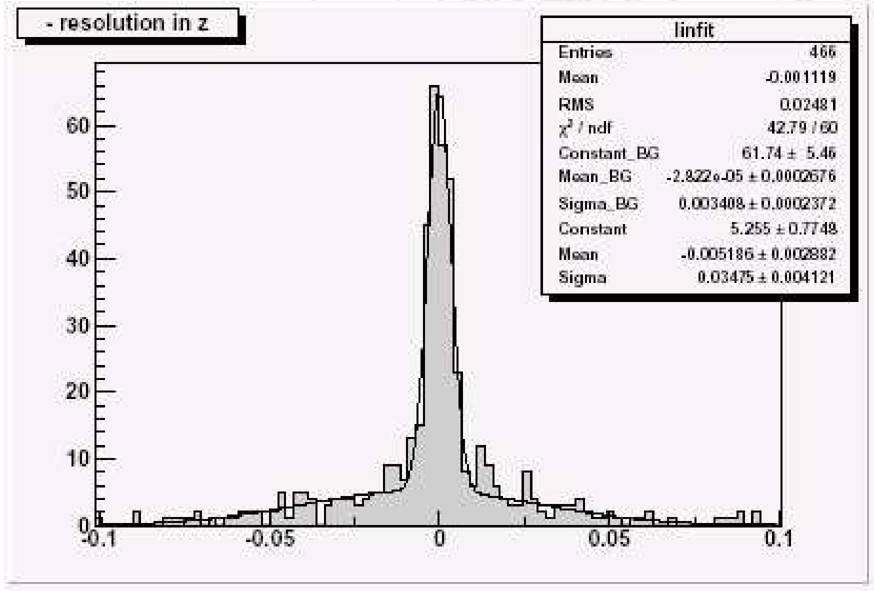
V.Karimäki, CMS Note 1997/051

KalmanVertexFitter

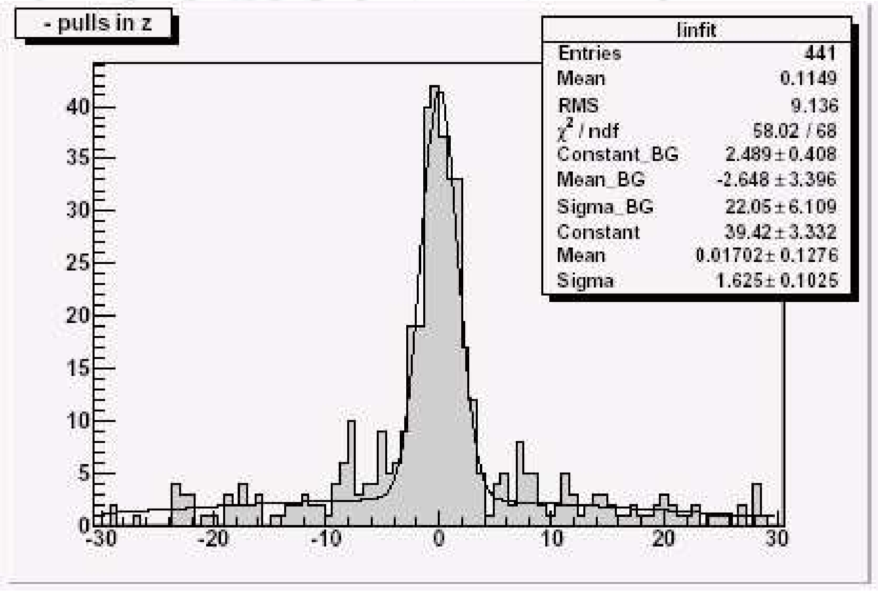
R.Früwirth et al., Computer Physics Comm. 96 (1991) 189-208

$c\bar{c}$, 100 GeV, $\eta \leq 1.4$

Least squares fit, z-resolution and pull



linfit	
Entries	466
Mean	-0.001119
RMS	0.02481
χ^2 / ndf	42.79 / 60
Constant_BG	61.74 ± 5.46
Mean_BG	-2.822e-05 ± 0.0002676
Sigma_BG	0.003408 ± 0.0002372
Constant	5.255 ± 0.7749
Mean	-0.005186 ± 0.002882
Sigma	0.03475 ± 0.004121



linfit	
Entries	441
Mean	0.1149
RMS	9.136
χ^2 / ndf	58.02 / 68
Constant_BG	2.489 ± 0.408
Mean_BG	-2.648 ± 3.396
Sigma_BG	22.05 ± 6.109
Constant	39.42 ± 3.332
Mean	0.01702 ± 0.1276
Sigma	1.625 ± 0.1025

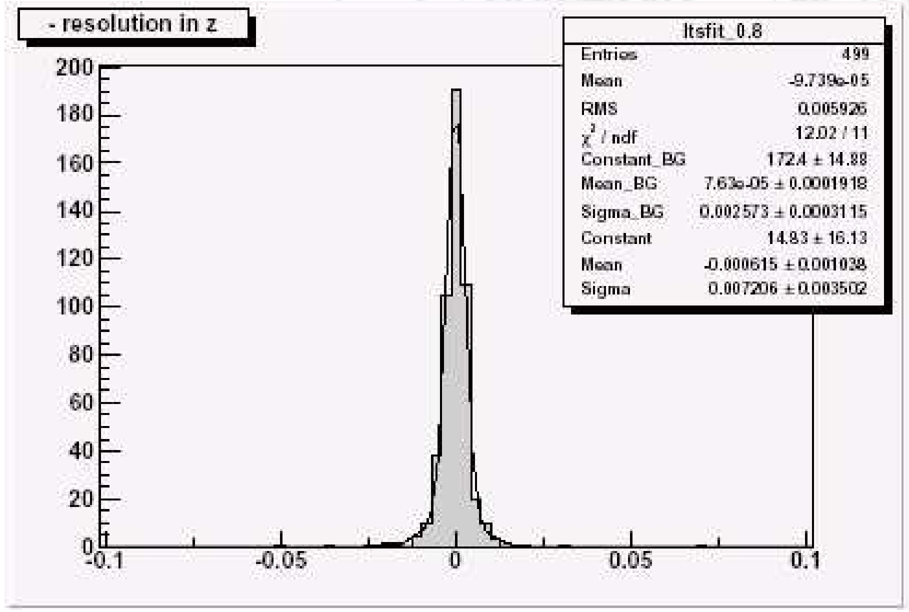
Trimming Vertex Fitter

$$\hat{\beta}_{LTS} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{\boxed{h<}n} r_i^2(\beta)$$

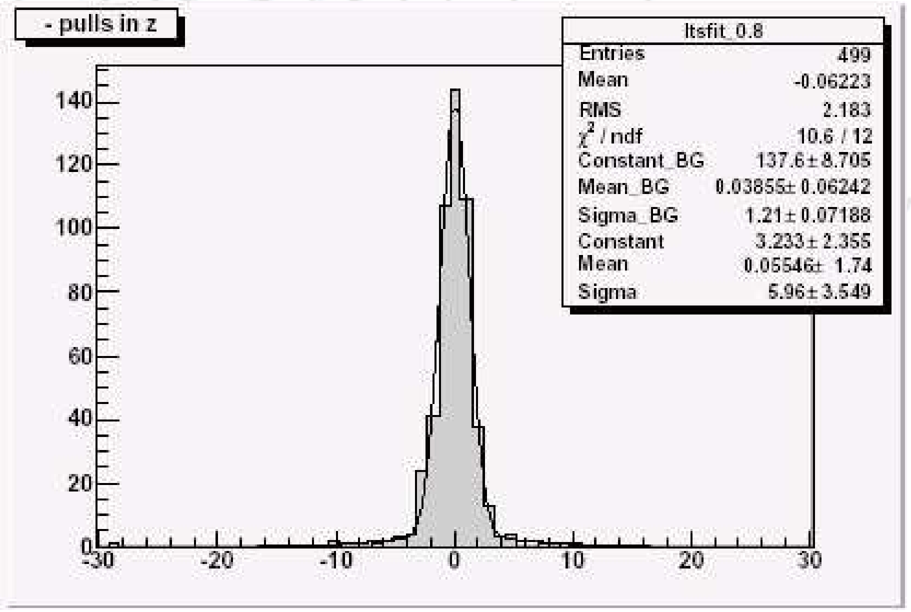
LTSVertexFitter: fast Least Trimmed (sum of) Squares

- use h most compatible tracks out of N (1 - h/N: trimming fraction) and fit them with one of the LS fitters
- algorithm: Fast-LTS (iterative) *P.J. Rousseuw, 1999*
- breakdown point ≈ 1-h/N
- user can choose trimming fraction
  - e.g. 3-prong τ, 4 tracks in cone
    - choose h/N = 0.75

$c\bar{c}$, 100 GeV, $\eta \leq 1.4$
LTSVertexFitter (80%)

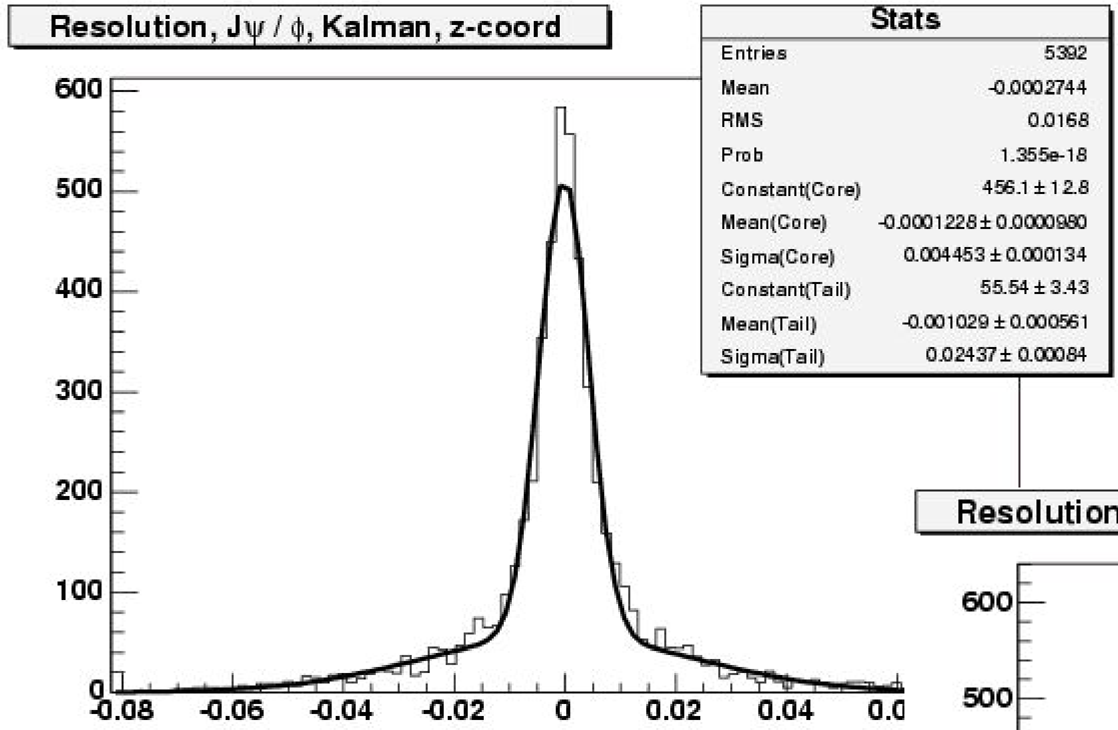


ltsfit_0.8	
Entries	499
Mean	-9.739e-05
RMS	0.005926
χ^2 / ndf	12.02 / 11
Constant_BG	172.4 ± 14.88
Mean_BG	7.63e-05 ± 0.0001918
Sigma_BG	0.002573 ± 0.0003115
Constant	14.83 ± 16.13
Mean	-0.000615 ± 0.001038
Sigma	0.007206 ± 0.003502



ltsfit_0.8	
Entries	499
Mean	-0.06223
RMS	2.183
χ^2 / ndf	10.6 / 12
Constant_BG	137.6 ± 8.705
Mean_BG	0.03855 ± 0.06242
Sigma_BG	1.21 ± 0.07188
Constant	3.233 ± 2.355
Mean	0.05546 ± 1.74
Sigma	5.96 ± 3.549

AdaptiveVertexFitter(2)

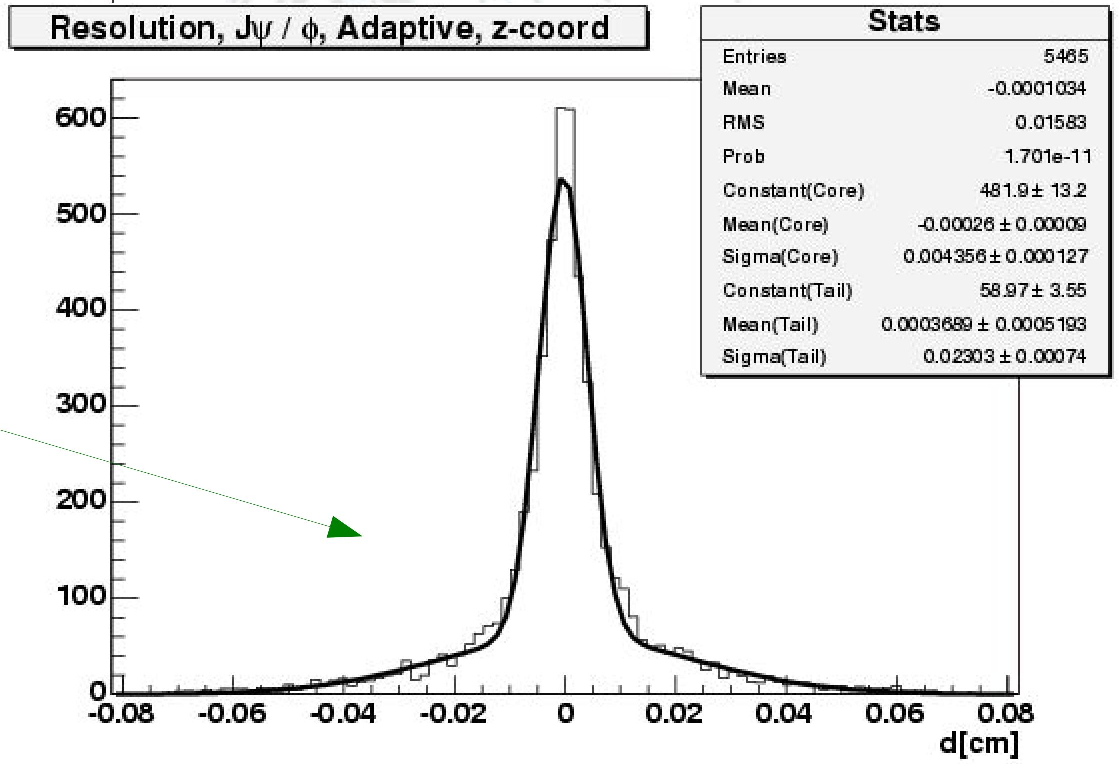


Stats	
Entries	5392
Mean	-0.0002744
RMS	0.0168
Prob	1.355e-18
Constant(Core)	456.1 ± 12.8
Mean(Core)	-0.0001228 ± 0.0000980
Sigma(Core)	0.004453 ± 0.000134
Constant(Tail)	55.54 ± 3.43
Mean(Tail)	-0.001029 ± 0.000561
Sigma(Tail)	0.02437 ± 0.00084

Comparing AVF with KVF in J/ψ φ->KKμμ.

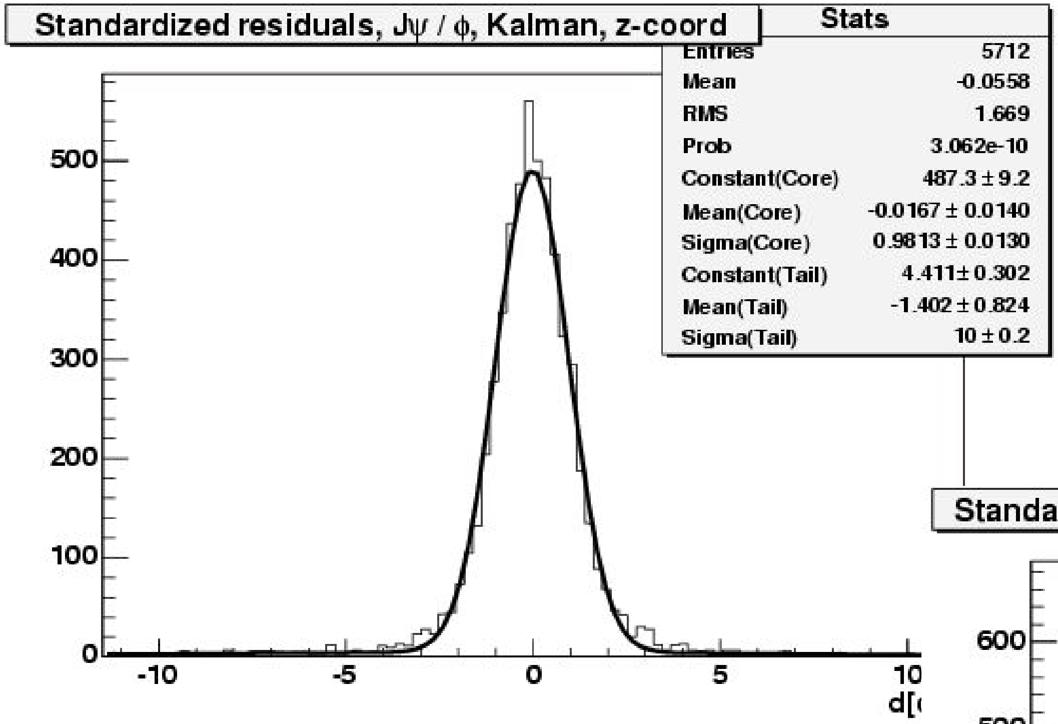
Resolutions, z coordinate

2% more events within |z|<800 microns.



Stats	
Entries	5465
Mean	-0.0001034
RMS	0.01583
Prob	1.701e-11
Constant(Core)	481.9 ± 13.2
Mean(Core)	-0.00026 ± 0.00009
Sigma(Core)	0.004356 ± 0.000127
Constant(Tail)	58.97 ± 3.55
Mean(Tail)	0.0003689 ± 0.0005193
Sigma(Tail)	0.02303 ± 0.00074

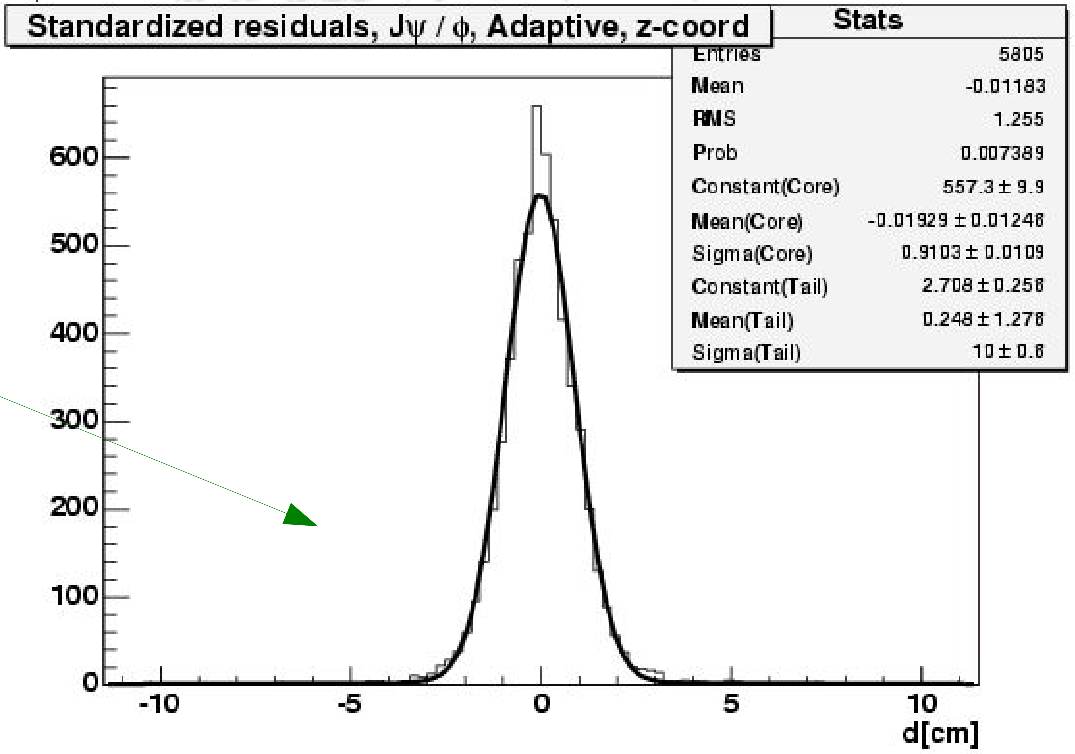
AdaptiveVertexFitter(3)



Comparing AVF with KVF in J/ψ φ->KKμμ.

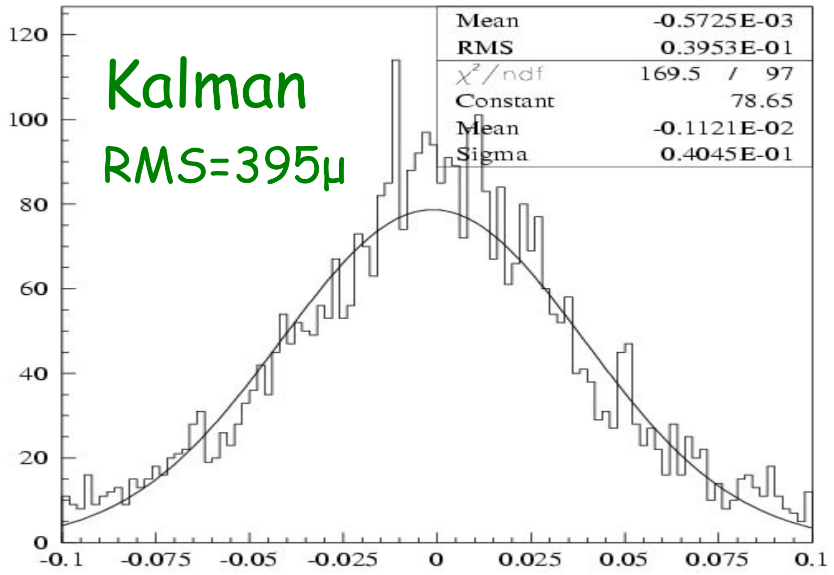
Resolutions, z coordinate

Much closer to Gaussian

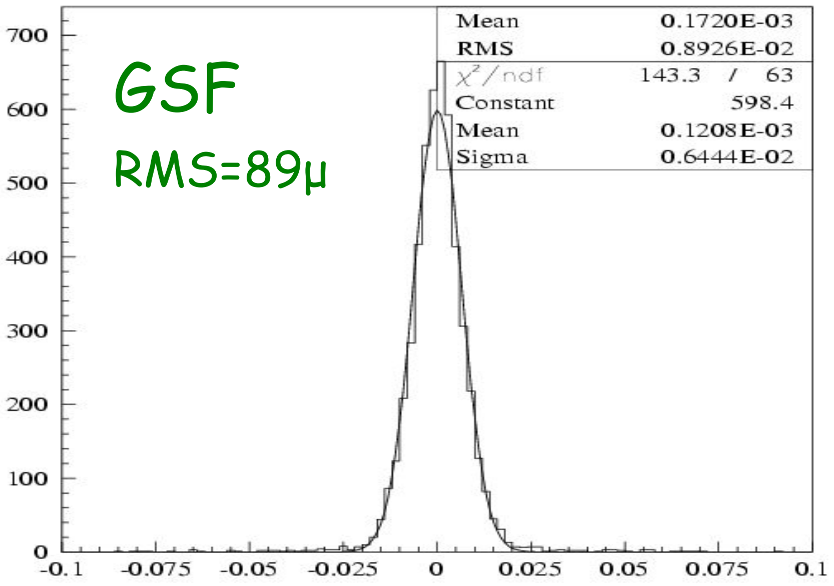


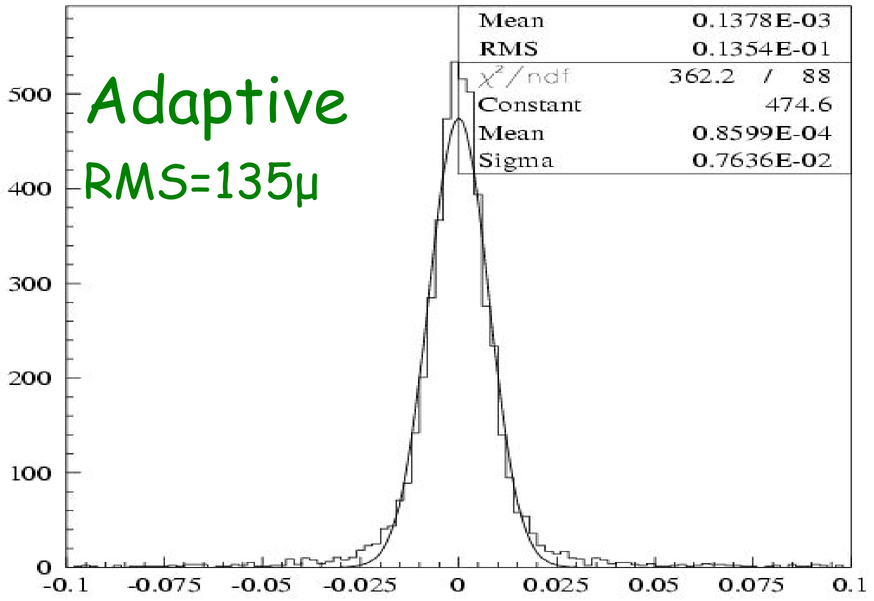
GaussianSumVertexFitter(2)

Artificial data, 4 tracks from a primary vertex, 1 track from a secondary vertex (3mm away). Tracks smeared with two Gaussians,"tail Gaussian" has a weight of 10 %, and is 10 times wider. GSF "sees" the right mixture; KVF and AVF see only core component.

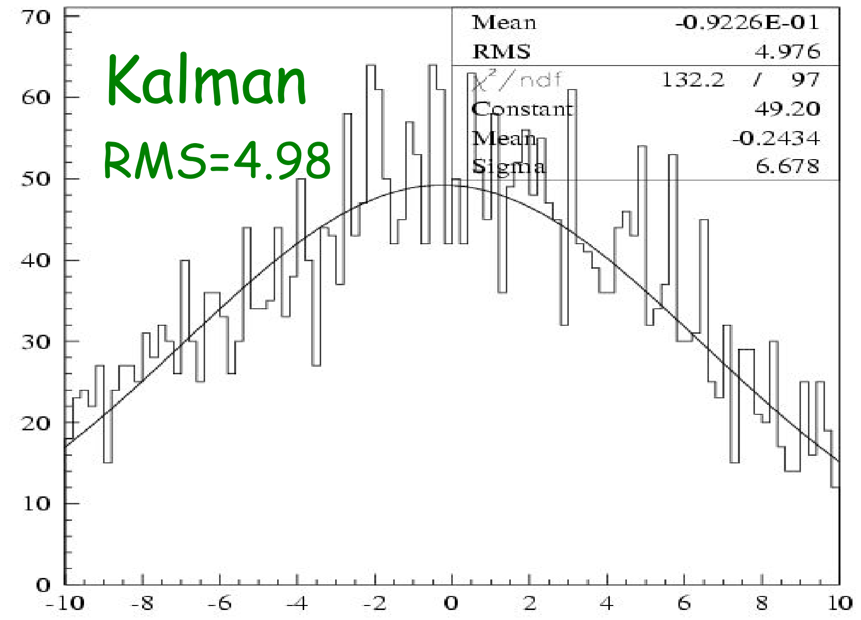


Resolution, z coordinate

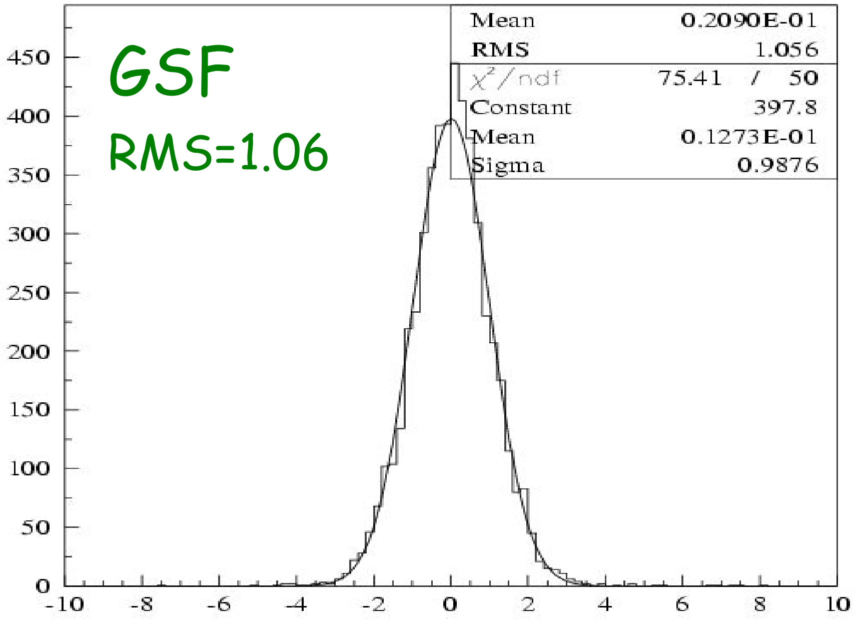


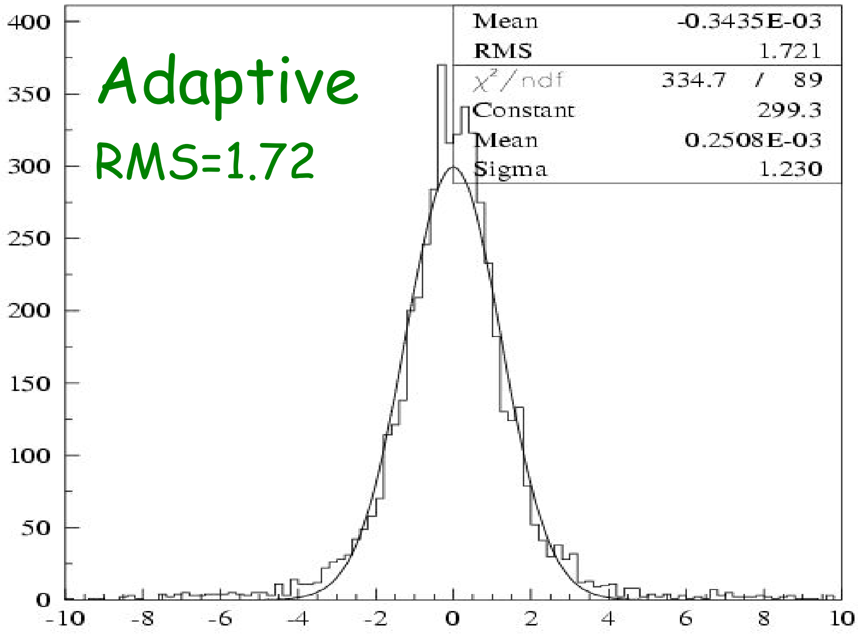


GaussianSumVertexFitter(3)



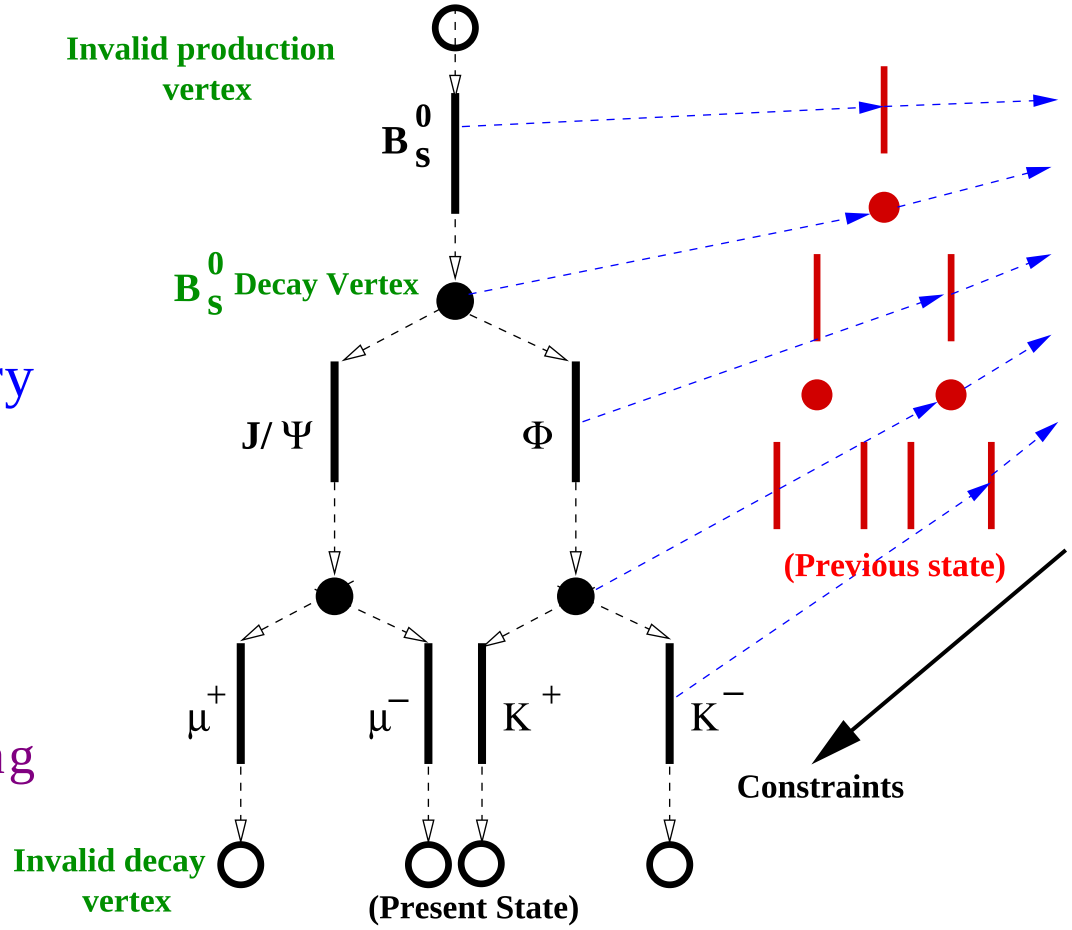
Pulls, z coordinate





The decay tree: KinematicTree

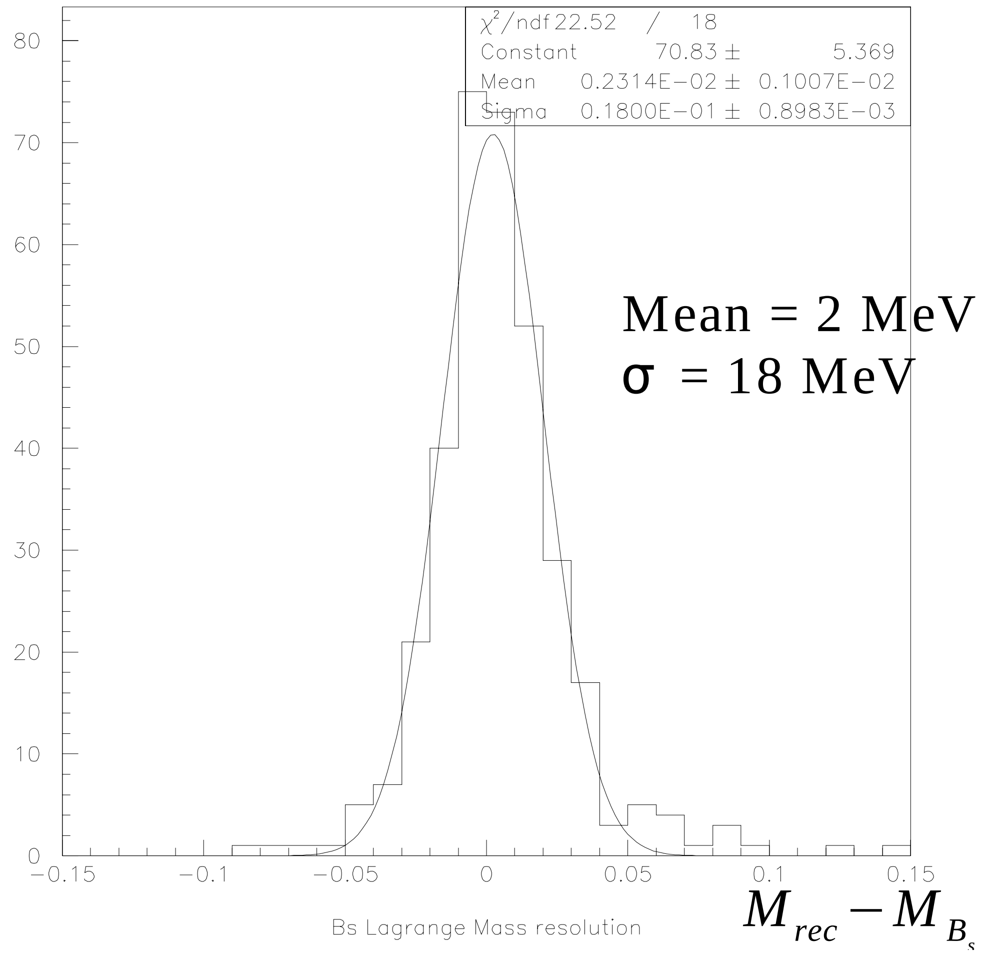
- Decay tree made of independent particles and vertices
- Navigation mechanism
- Tree is created from " bottom" to " top" , i.e. from final state to primary particle
- Produced by the kinematic fit
- Could also be produced by Exclusive Decay Package:
  - ➢ Collection of trees, each representing one possible combination of input tracks.



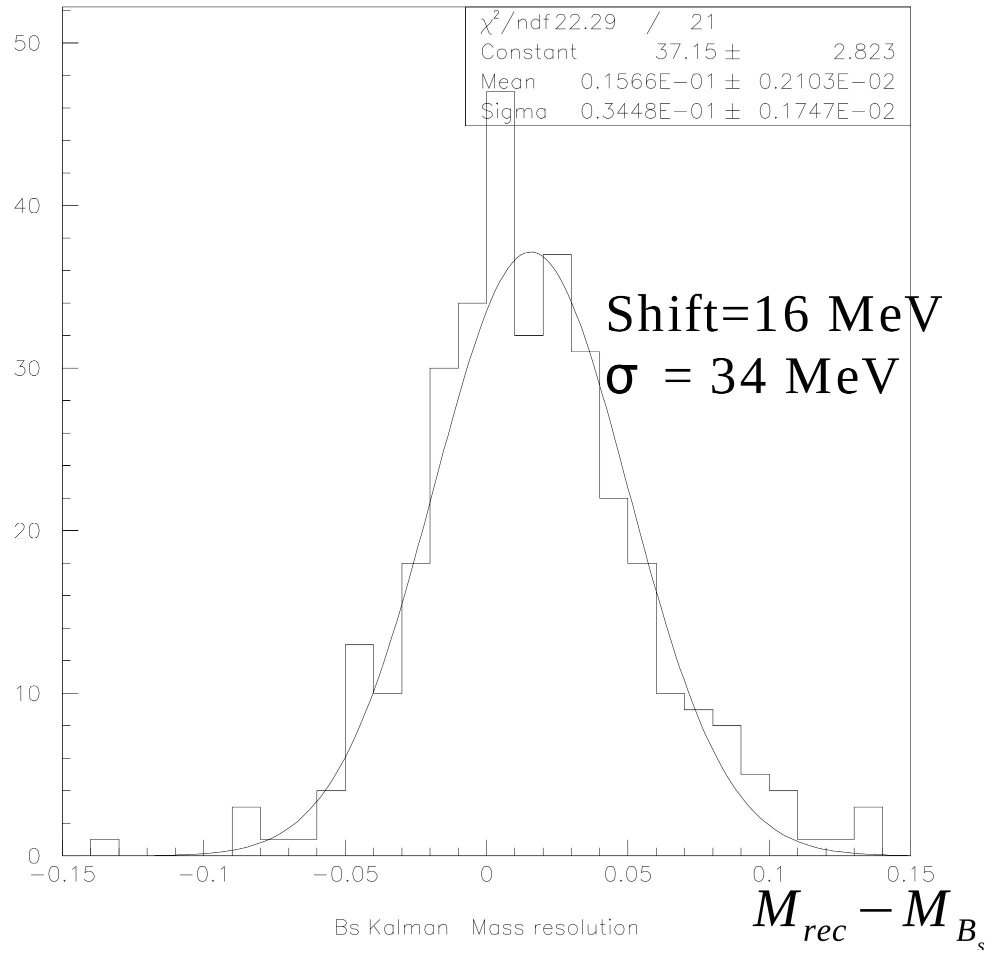
Example: The decay $B_s \to J/\psi\ \varphi$

Residual of the $\mu^+\mu^- K^+ K^-$ 4-track invariant mass with and without muon invariant mass and pointing constraints:

Vertex fit with constraints

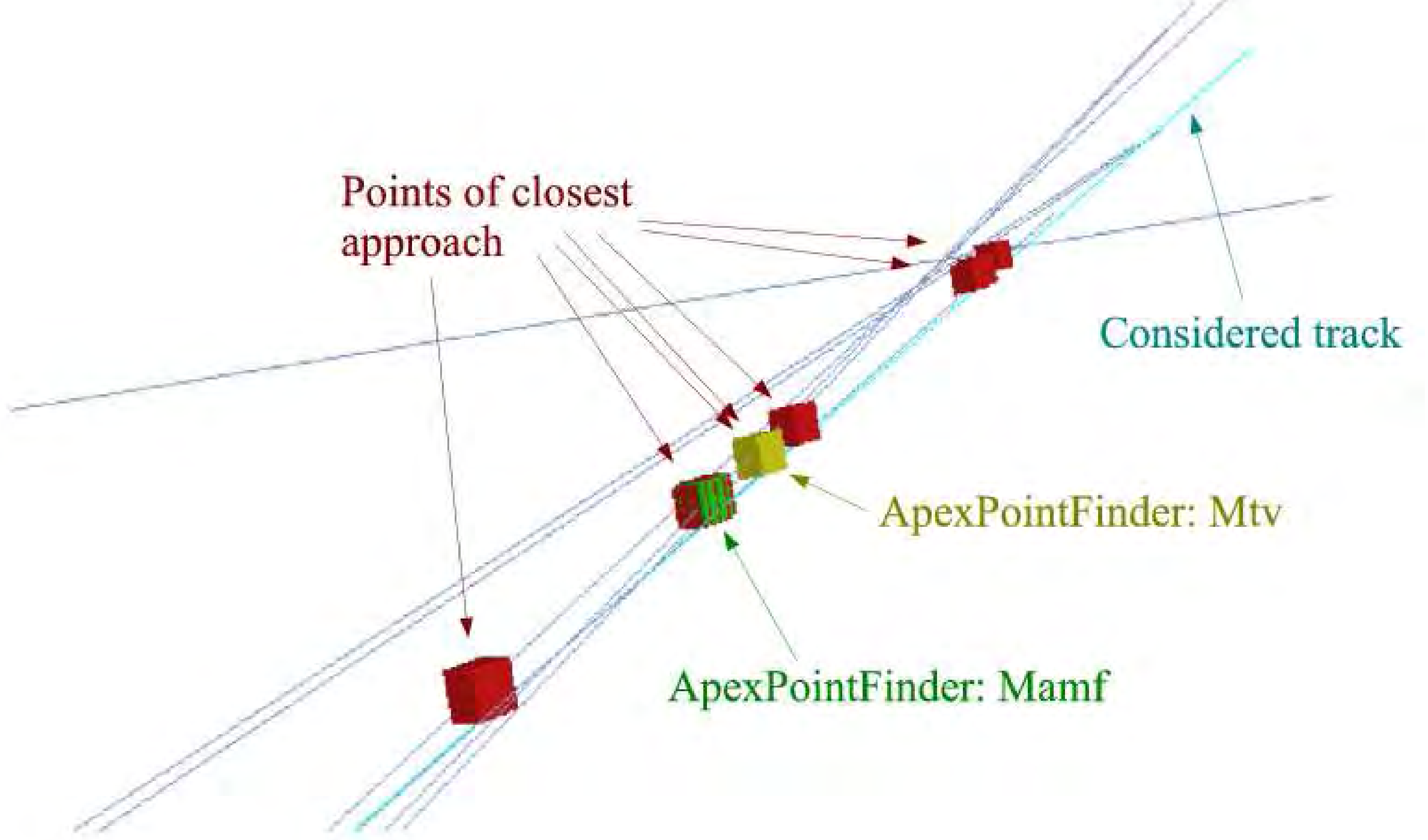


Vertex fit only



Apex Points

To fix the problem with the triangle inequality, one may try to find **a point that fully represents the track**. One such point could be the ApexPoint; that is a cluster point in the set of all Points of Closest Approach that lie on the considered track. The most promising ApexPoint finding algorithm is "Mtv": Minimal Two Values.



Divisive Clusterers - Principal Vertex Finder

Divisive algorithm, search for primary and secondary vertices

- based on track ↔ vertex compatibility at point of closest approach
- at each iteration:

fit all tracks to a common vertex

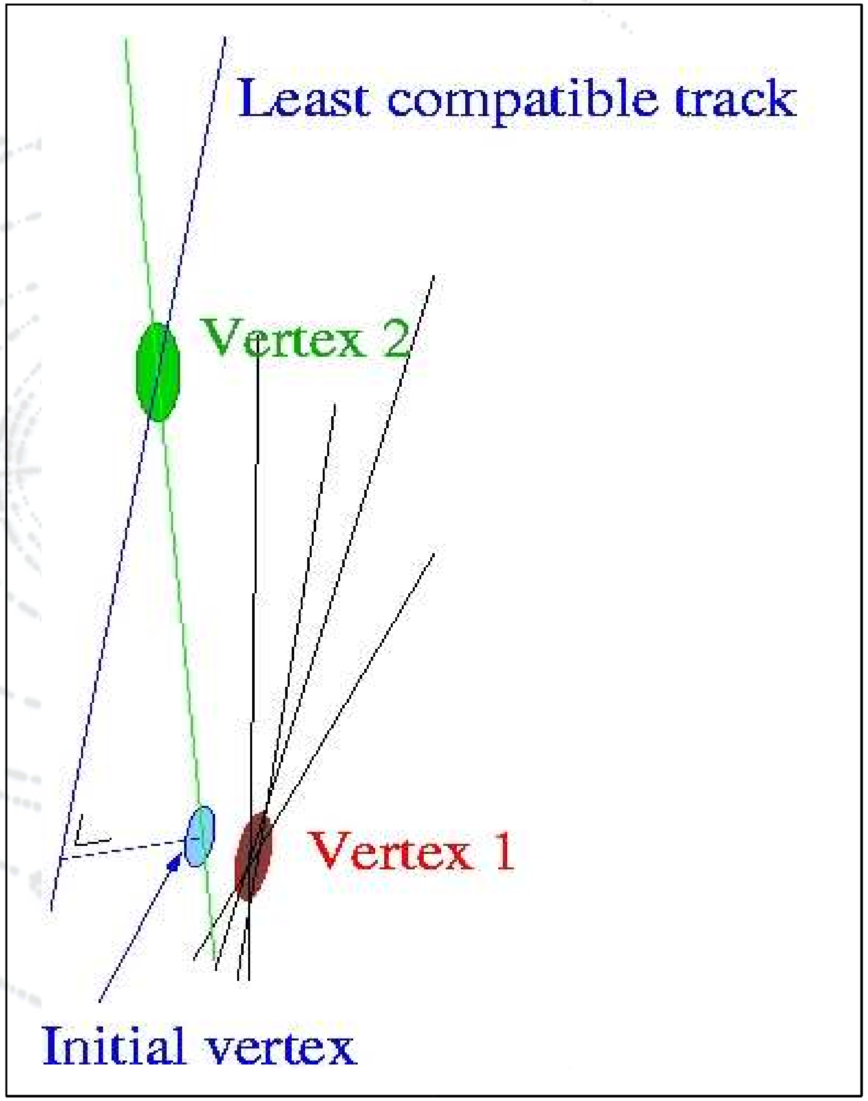
remove least compatible track

refit vertex.

1 vertex candidate

+ 1 set of discarded tracks

- final cleanup: vertices with low χ^2 probability discarded

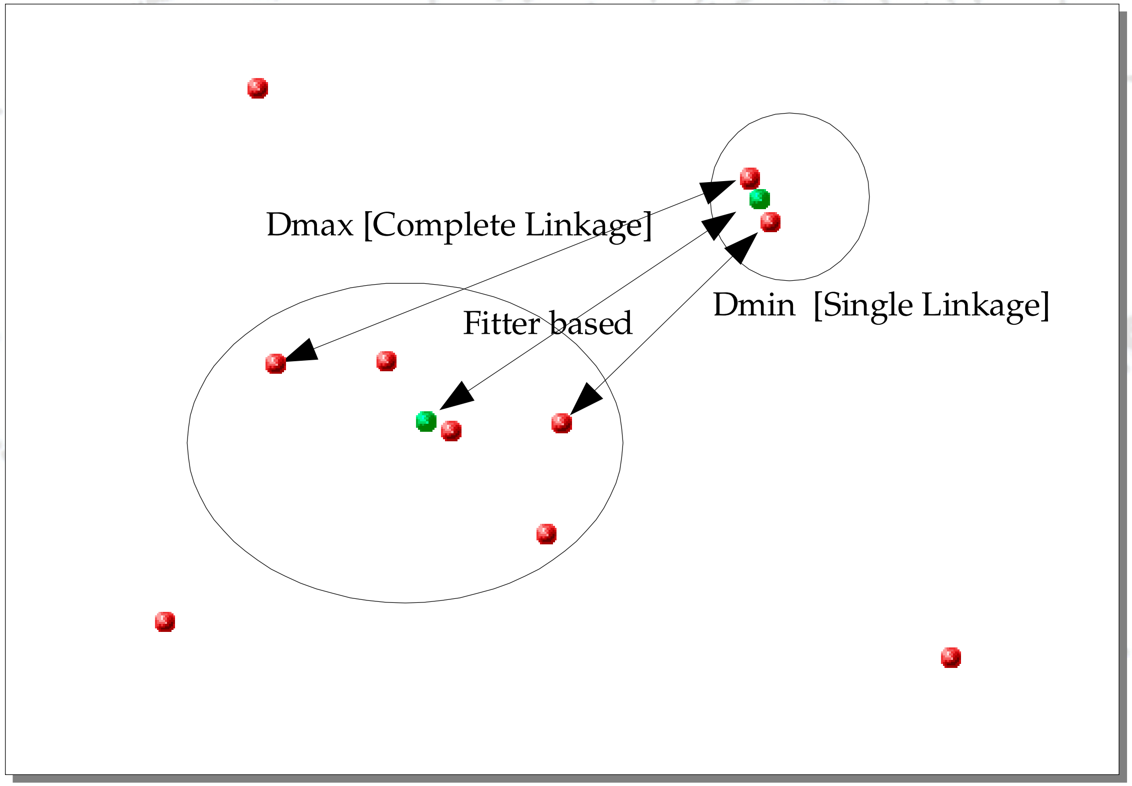


Agglomerative Finders (2)

The adding of the most compatible track requires the proper definition of a metric; a distance measure between two track clusters.

D(cluster, cluster) = Dmin, Dmax, Dmedian, Dmean, ...

The distance measure can also be defined by representatives of a cluster (e.g. fitter based).

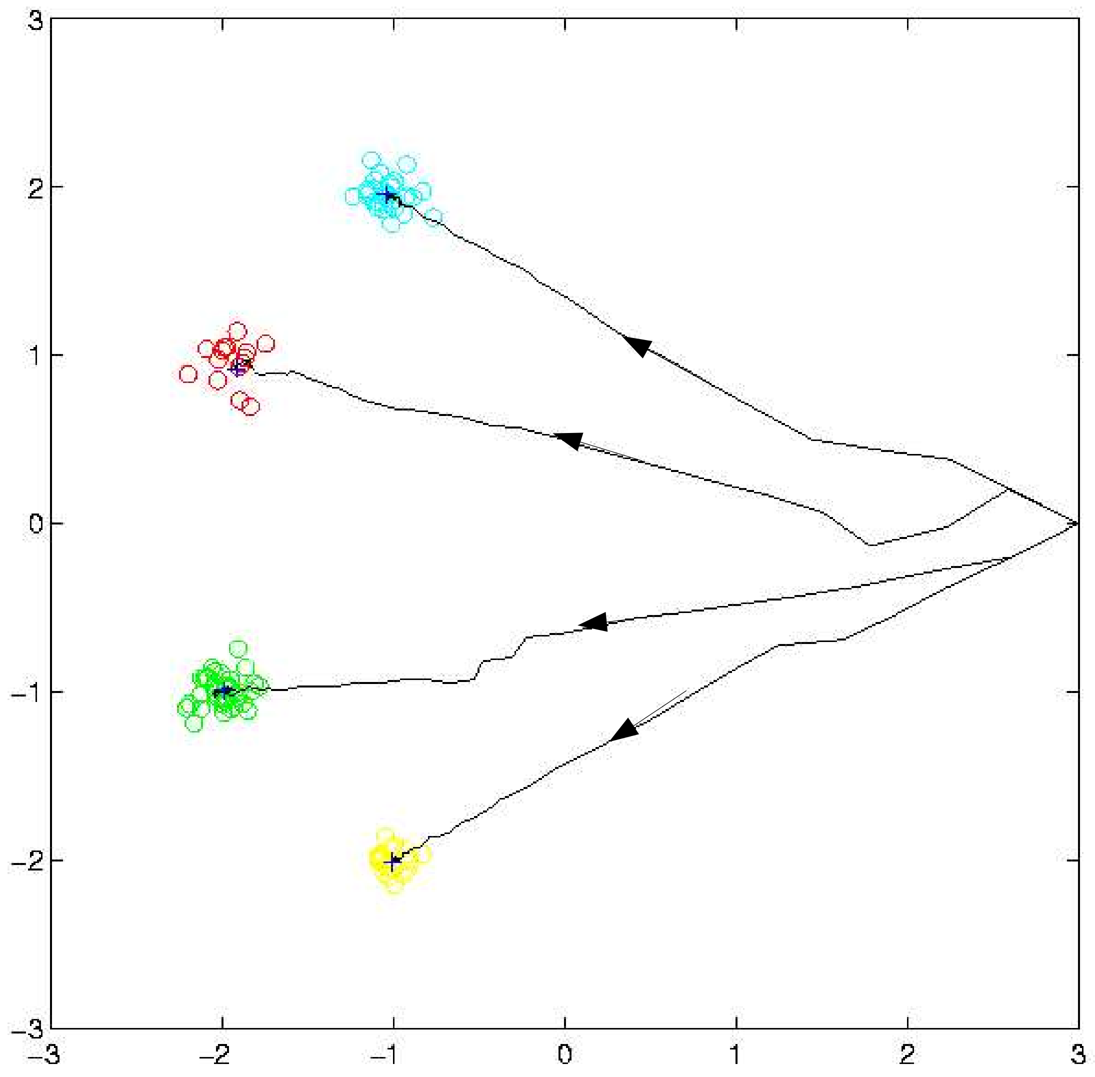


Vector Quantisation

Vector quantisation works by having a set of prototypes learn to represent the ApexPoints. The prototypes will then be interpreted as Vertex candidates.

Learning algorithm:
frequency sensitive
competitive learning.

This is work in progress, only
(very promising) preliminary results
have been obtained so far.



3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3