

A Vertex Reconstruction Toolkit and Interface to Generic Objects (VERTIGO) (part 1 of 2)

A Vertex Reconstruction Toolkit and Interface to Generic Objects (VERTIGO)



Winfried Mitaroff



Gerald Richter



Wolfgang Waltenberger

Austrian Academy of Sciences, Institute of High Energy Physics
Nikolsdorfer Gasse 18, A-1050 Vienna, Austria, Europe

Abstract

A proposal is made for the design and implementation of a detector-independent vertex reconstruction toolkit and interface to generic objects (VERTIGO). The first stage aims at re-using existing state-of-the-art algorithms for geometric vertex finding and fitting by both linear (Kalman filter) and robust estimation methods. Prototype candidates for the latter are a wide range of adaptive filter algorithms being developed for LHC/CMS, as well as proven ones (like ZVTOP of SL/C/SLD). In a second stage, also kinematic constraints will 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 abstract interfaces (using adaptors for access from/to the particular environment) and a set of analysis and debugging tools. The implementation follows an open source approach and is easily adaptable to future standards.

Work has started with the development of a specialized visualisation tool, following the model-view-controller (MVC) paradigm; it is based on COIN3D and may also include interactivity by PYTHON scripting. A persistency storage solution, intended to provide a general data structure, was originally based on top of ROOT and is currently being extended for AIDA and XML compliance; interfaces to existing or future event reconstruction packages are easily implementable. Flexible linking to a math library is an important requirement; at present we use CLHEP, which could be replaced by a generic product.

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.
- Vertex fitting with kinematic constraints may be subject to the requirements of a subsequent physics analysis.

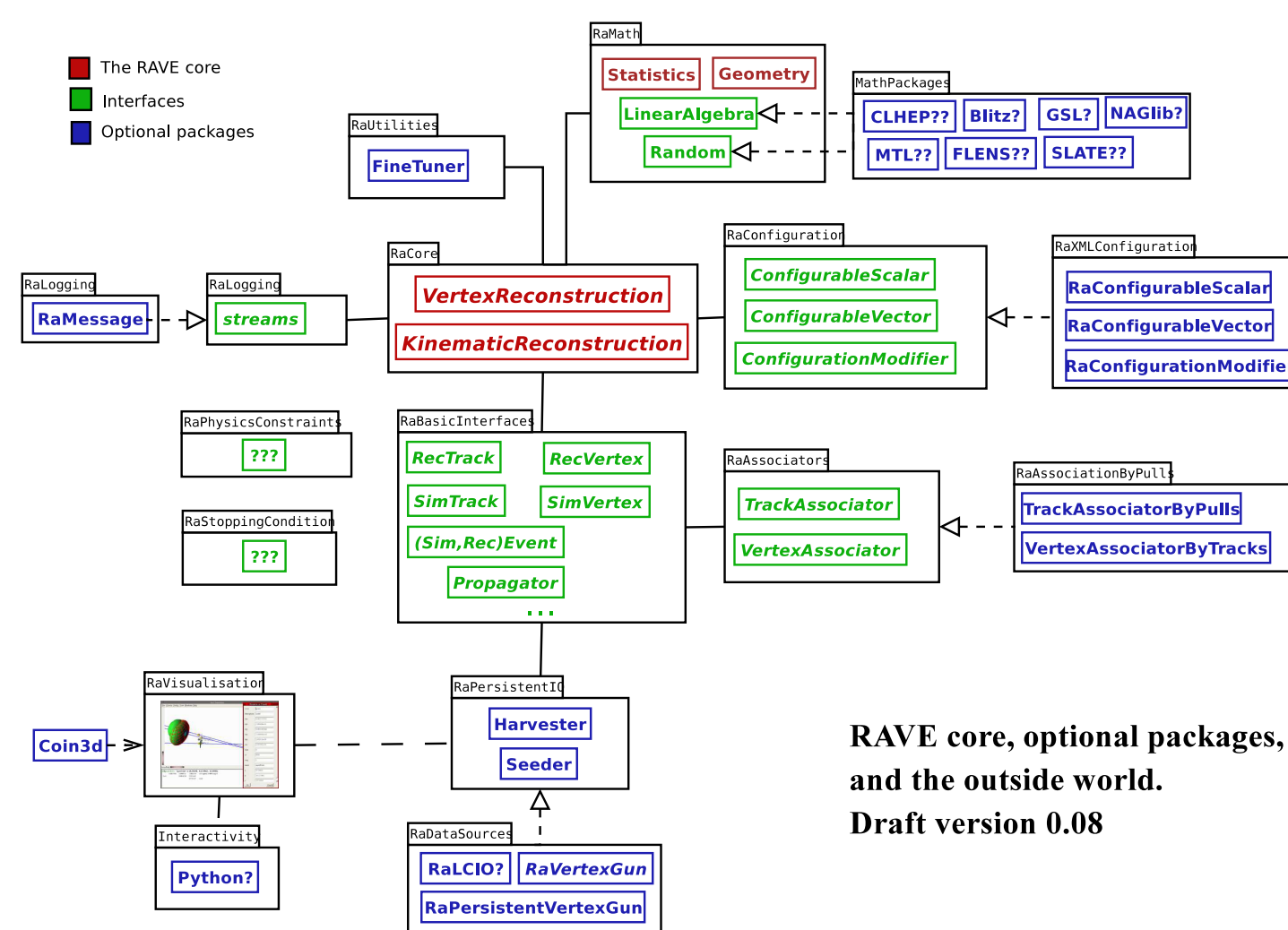
Why looking for a toolkit ?

- 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.

VERTIGO overall design – draft 0.08



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.

The RAVE core (1)

RAVE = "Reconstruction Algorithms for Vertices"

The list of candidate core algorithms is, at present, dominated by algorithms from the CMS offline reconstruction software (ORCA). Contributors are: R. Frühwirth, W. Waltenberger (HEPHY Vienna), J. d'Hondt, P. Vanlaere (IIHE Brussels), E. Chabanaud, N. Estre (IN2P3 Lyon), K. Prokofjev, 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 (unweighted 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/smoothing 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. distortions caused by "outliers" (i.e. measurements 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.

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 $T \rightarrow 0$, the Fermi function approximates the Heaviside function, and the assignment turns into a "hard" one ($w_i = 1$ or 0).

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

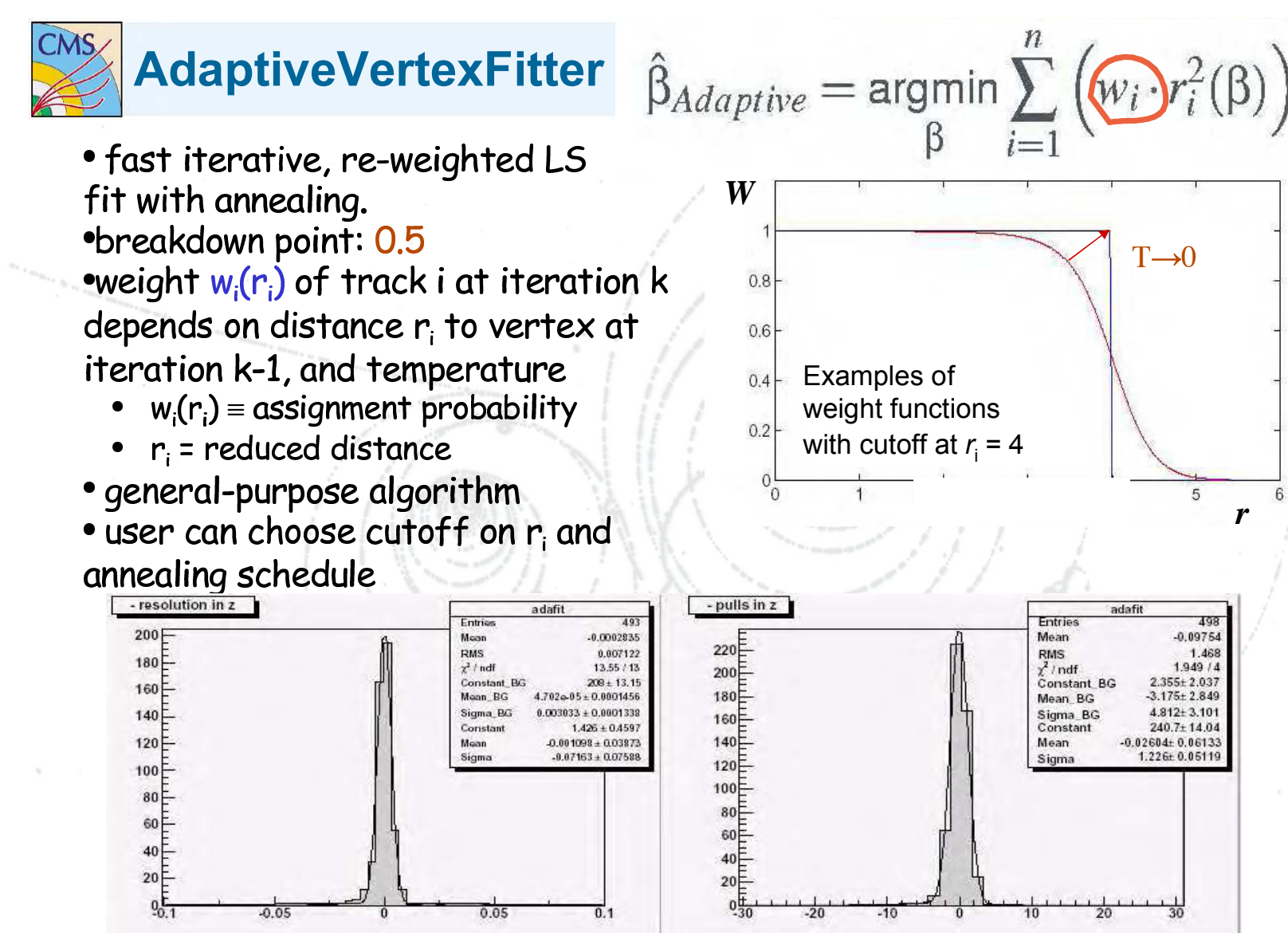
$$w_{ij}(r_{ij}, T) = \frac{e^{-r_{ij}^2/2T}}{\sum_{\ell=1}^n e^{-r_{i\ell}^2/2T} + e^{-r_{cut}^2/2T}}$$

- 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.

The RAVE core (4)

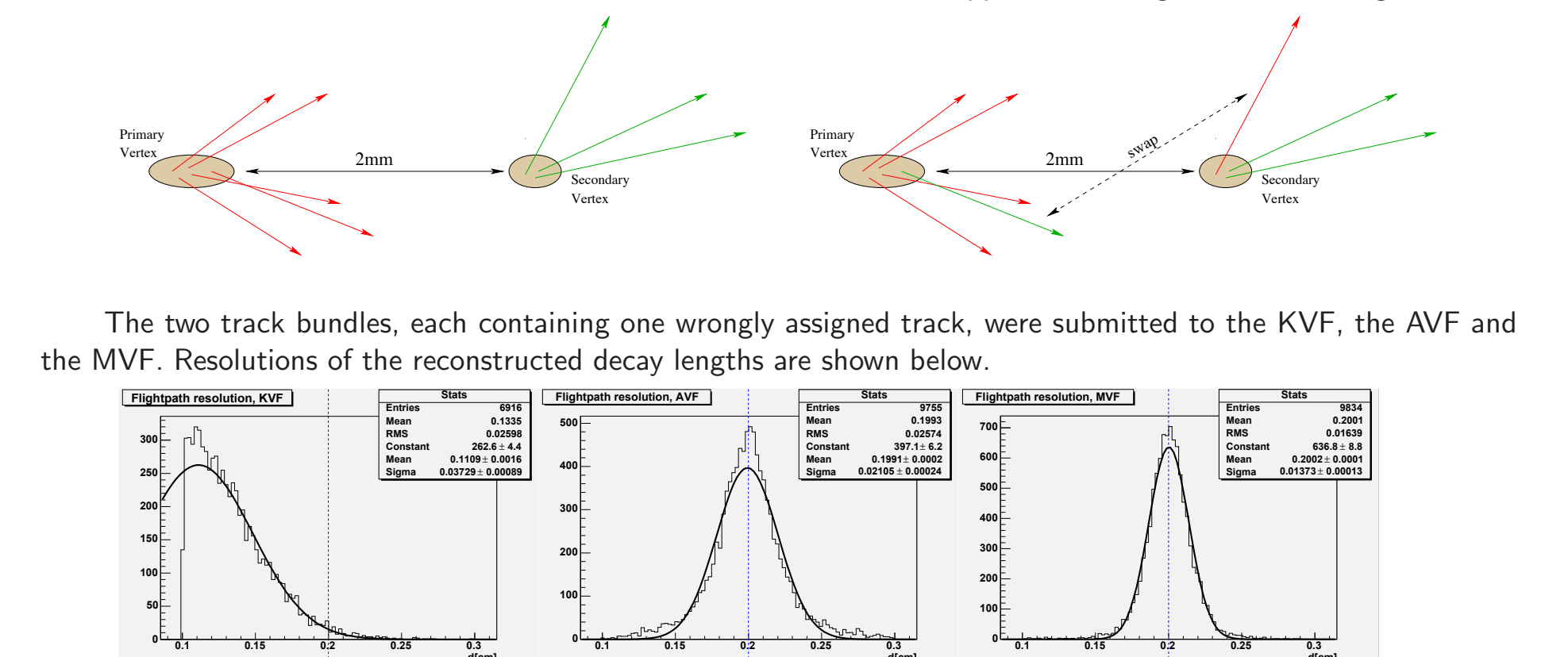
Vertex fitting – some remarks

- Beam spot**: All presented vertex fitting packages can include, as an additional virtual measurement, prior information of the vertex position – e.g. the beam interaction profile ("beam spot") in case of fitting the primary vertex (PV). The beam spot can be obtained from the distribution of fitted PV positions of a sample of events with high number of well-measured and unambiguously assigned PV tracks over a period of stable beam conditions.
- Test statistic**: When using algorithms with soft assignment (weights), the minima of the objective function are no longer χ^2 -distributed. Therefore, their power as a test statistic (e.g. by formally calculating a "probability") is questionable, and other methods (e.g. based on the last iteration's weights) should be considered. The weakness of a χ^2 -based test statistic is true also with hard assignment algorithms, as long as the vertex fit still includes outlier measurements.
- Neutral tracks**: Both the KVF and AVF are able to include, if available, measurements of neutral tracks – e.g. originating from a calorimeter, or calculated from the decay products of a neutral particle. For taking full advantage of neutral tracks, however, a kinematic vertex fit must be used (see below).



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 opening angle 0.5 rad; one SV at (0, 0, 0.2) cm with 3 tracks, total jet momentum (-15, 0, 20) GeV and opening 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 method of Least Squares is seen to be our best course when we have thrown overboard a certain portion of our data -- a sort of sacrifice which has often to be made by those who sail the stormy seas of Probability."

F. Y. Edgeworth, 1887

"Let us not throw away data all too hastily. Instead, let us weight and reweight the data, consider and reconsider alternative models. Only if we must, at the latest possible stage, shall we distinguish between 'in' and 'out', between signal and noise."

The CMS vertexing circle, 2004

A Vertex Reconstruction Toolkit and Interface to Generic Objects (VERTIGO) (part 2 of 2)

The RAVE core (5)

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 choose 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).

The RAVE core (6)

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

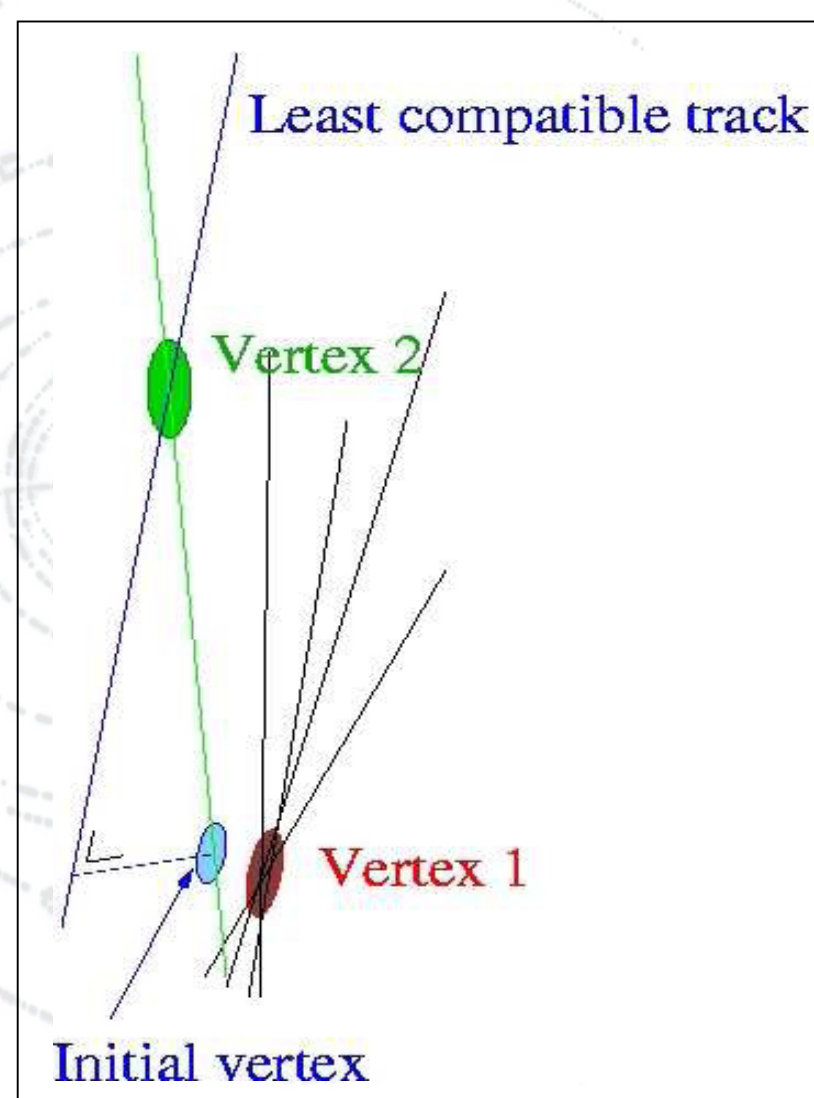
Documentation (based on Doxygen) of the algorithms, including information about their scope of application, will be provided. The proper choice of algorithms is also supported by the SKIN concept.

Divisive Clusterers - Principal Vertex Finder

Divisive algorithm, search for primary and secondary vertices

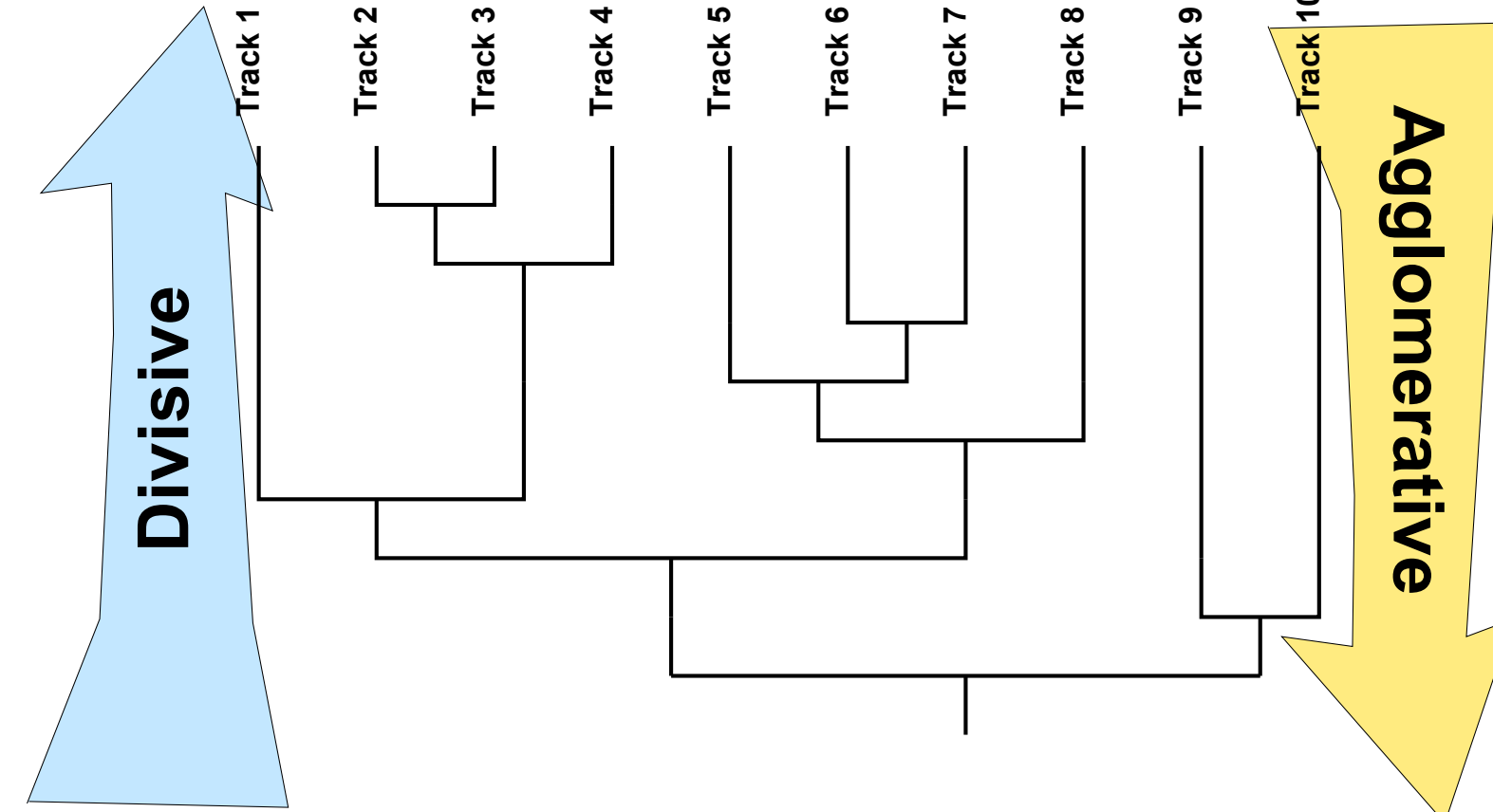
- based on track \leftrightarrow vertex compatibility at point of closest approach
- at each iteration:
 - fit all tracks to a common vertex
 - remove least compatible track
 - refit vertex.
 - 1 vertex candidate
 - + 1 set of discarded tracks

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



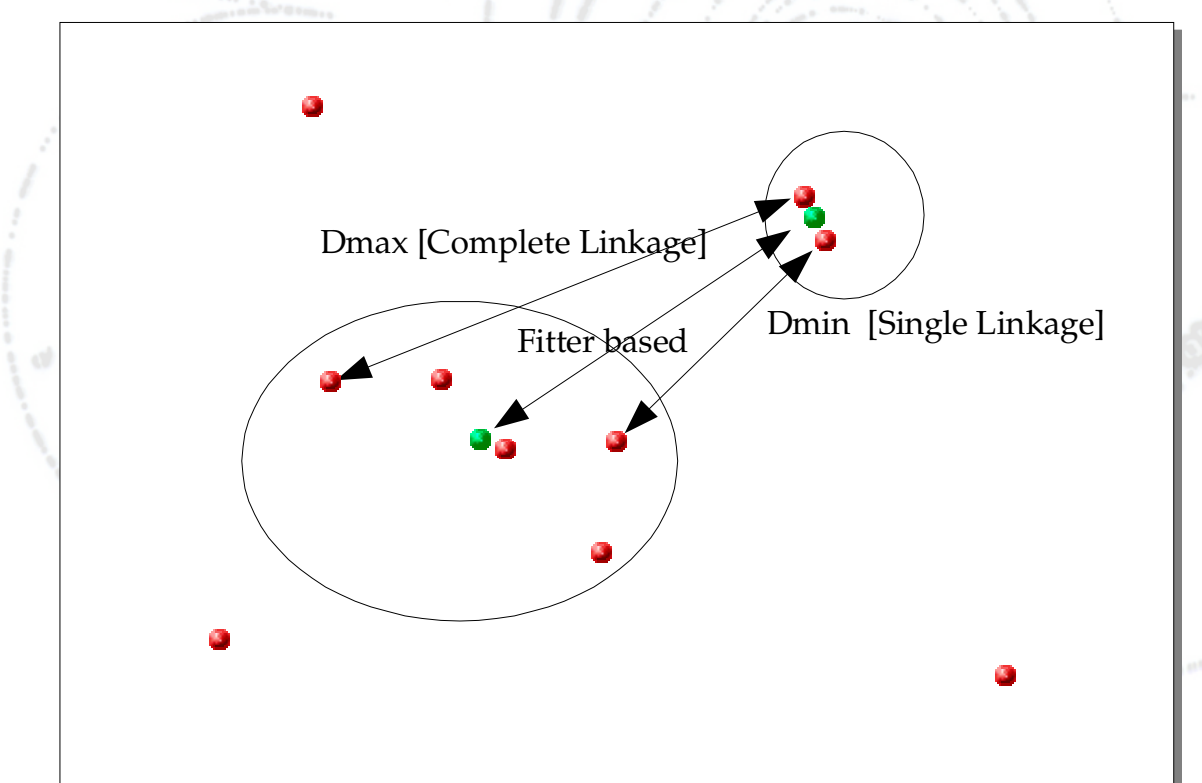
Hierarchic clustering example

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



Agglomerative Finders (2)

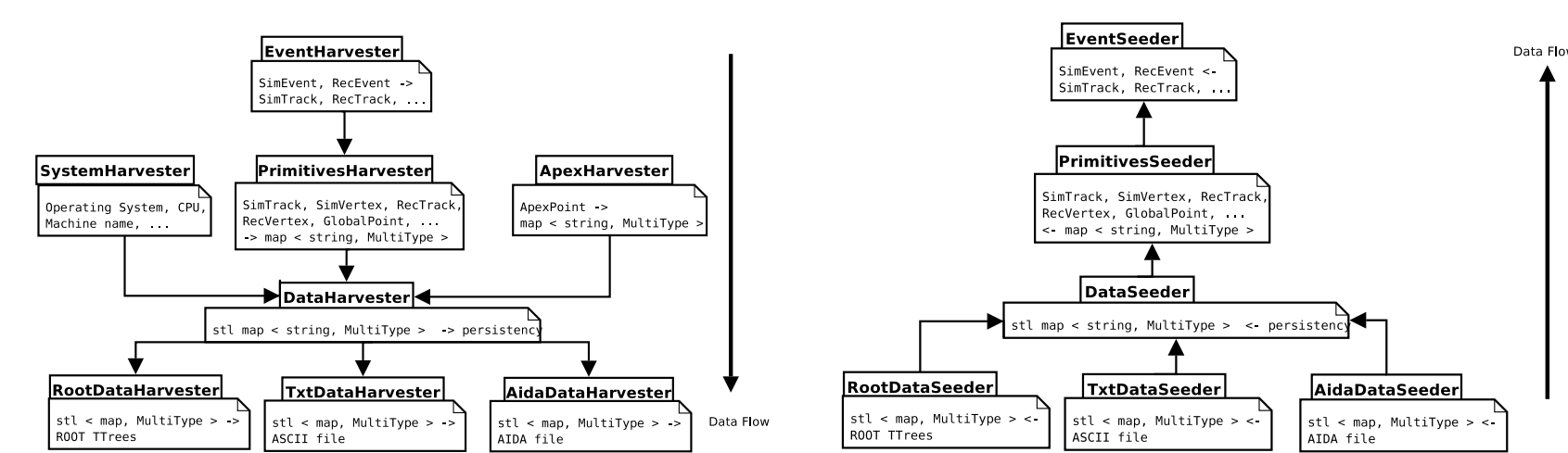
The adding of the most compatible track requires the proper definition of a metric; a distance measure between two track clusters.
 $D(\text{cluster}, \text{cluster}) = D_{\min}, D_{\max}, D_{\text{median}}, D_{\text{mean}}, \dots$
 The distance measure can also be defined by representatives of a cluster (e.g. fitter based).



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 \rightarrow STL map \rightarrow ASCII/ROOT/AIDA file (Harvester) and vice versa (Seeder).
- The STL mapping is heterogeneous: it handles int/double/string objects as MultiType.



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.

Analysis & debugging tools (3)

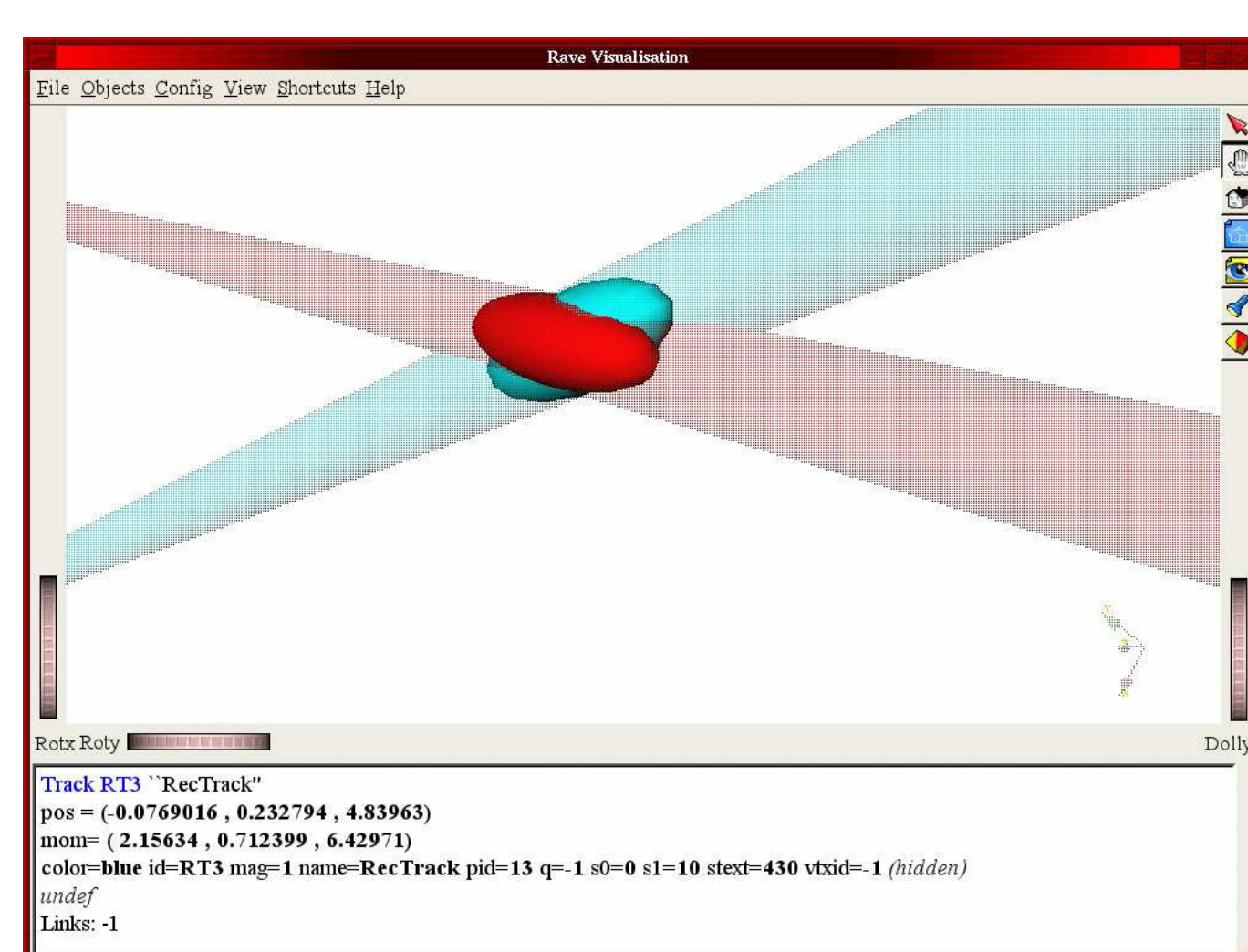
Math library package

- 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-licensed); but none providing the full functionality required.

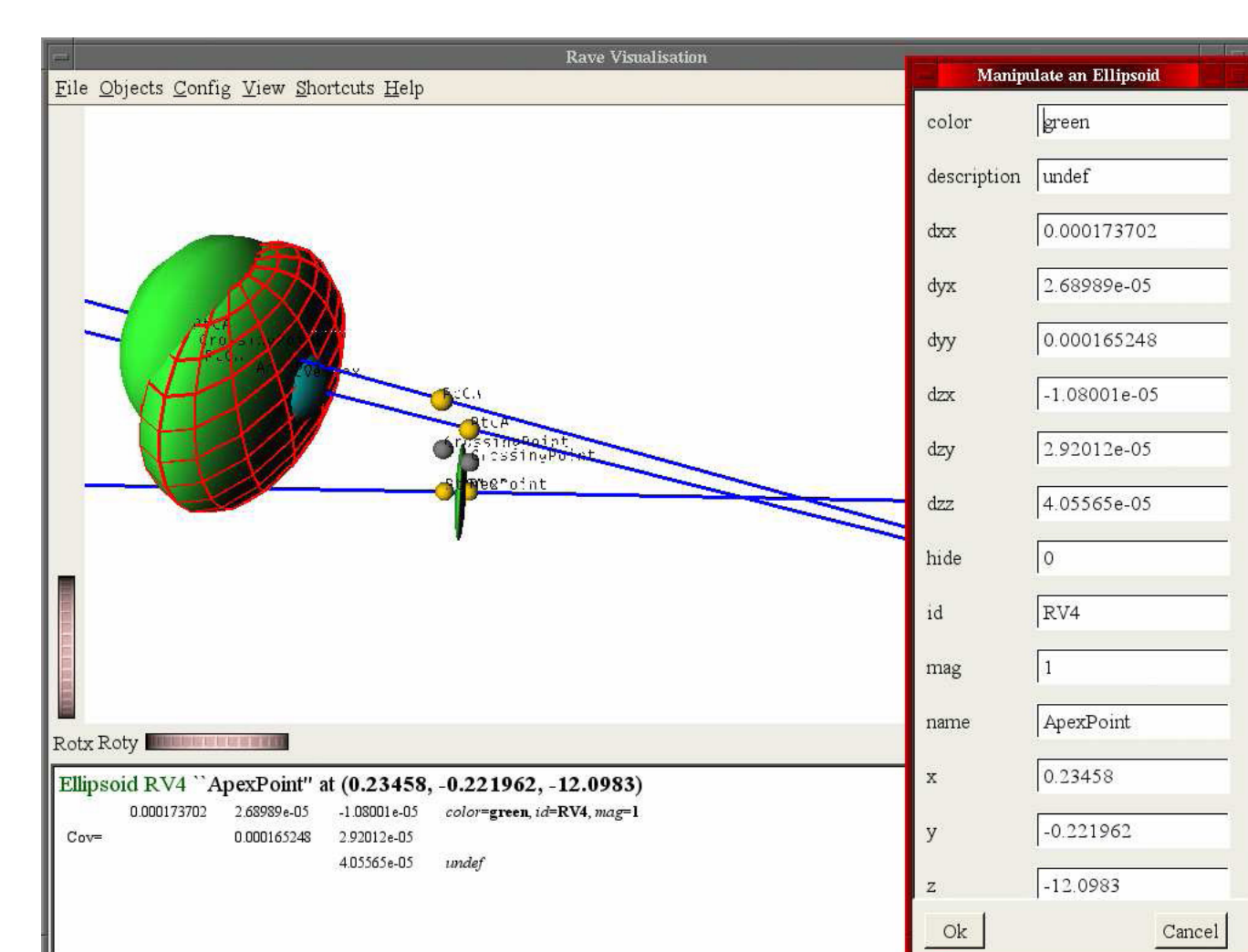
The SKIN concept

- 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.

Visualisation tool - example 1



Visualisation tool - example 2



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 (WV) 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 (WV), with some help from a graduate student (GR), 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.**