DETECTOR-INDEPENDENT VERTEX RECONSTRUCTION TOOLKIT (VERTIGO)

W. MITAROFF, G. RICHTER and W. WALTENBERGER
Institute of High Energy Physics, Austrian Academy of Sciences, Vienna, Austria

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 SLC/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 e.g. a generic product.

MOTIVATION AND GOALS

In the offline data reduction chain, the early stages – local pattern recognition 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 rather 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. It is not desirable for each new detector to re-code vertex reconstruction from scratch – provided there exists an adequate, reliable and easy-to-use TOOLKIT.

As a good point to start from, we propose 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.

DESIGN CONCEPTS OF VERTIGO

A draft version of the overall design – core, interfaces and optional packages – is shown in Fig. 1.

The RAVE core

The core, called RAVE (“Reconstruction Algorithms for Vertices”), is to become a collection of the best algorithms available for vertex reconstruction – finding, fitting and kinematics; starting with the packages developed by CMS [1], but open for entries by other parties. The code is to be based on C++ and HEP-wide (albeit not CERN-specific) OO standards.

Candidate core algorithms include packages of general tools (e.g. clustering), for vertex fitting (e.g. the deterministic annealing filter, DAF), and for vertex finding (e.g. the “apex point” method). At present, the list is dominated by algorithms implemented in the CMS offline reconstruction (ORCA [2]), but non-CMS candidates exist (e.g. ZVTOP [3]). New entries of first-class algorithms are highly welcome.

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 (see below).

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; good design will be the key of success.

Analysis & debugging tools

Analysis & debugging tools are optional packages, containing those parts of code which might be helpful without being strictly necessary. Prototypes of a few packages have already been written: the framework for a stand-alone realisation of VERTIGO, a persistency storage solution, data sources, and a visualisation tool; but much more work is

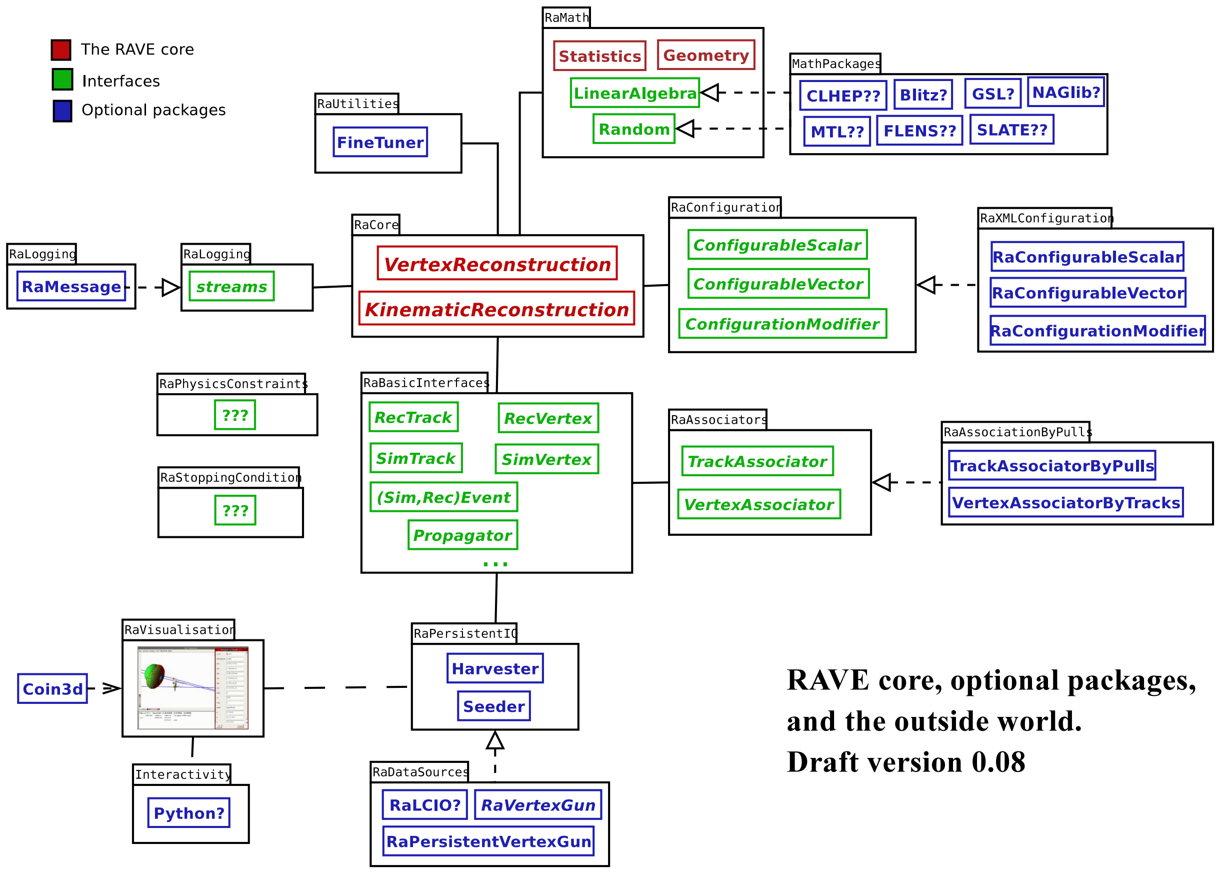


Figure 1: VERTIGO overall design (draft).

still to be done. Extensive use of open standards will minimize the burden of development for this part of the toolkit.

The persistency storage solution was originally based on top of ROOT [4]; it is currently being extended by more standard-compliant alternatives (AIDA [5] and XML). Data sources include a “vertex gun”, interfaces to LCIO [6], etc. All I/O is handled through a data harvesting concept (which may possibly be integrated as front/end in AIDA): object → STL map → ASCII / ROOT / AIDA / binary file / stream (“harvester”) and vice versa (“seeder”), see Fig. 2. The STL mapping is heterogeneous: it handles int/double/string objects as multi-type, thus supporting a syntactically simple use of this tool.

Visualisation is deliberately kept simple for the sake of detector-independence, fulfilling solely the need of giving insight into the geometric and associative correlations of the vertex reconstruction algorithms input and output. It follows the model-view-controller (MVC) paradigm and is based on Coin3D [7]. Object data are accessed as multi-type STL maps: at present only indirectly from a 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 are shown in Figs. 3 and 4.

Since the proper choice of a math library package (including linear algebra) is crucial for the efficiency and reliability of the toolkit, several candidates are being evaluated. CLHEP [4] appears to be the only choice freely available today, but there are serious doubts about its reliability. NAGlib [8] 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. Blitz++, FLENS, GSL, MTL [9] (all GPL-licenced), but up until lately none provides the full functionality required. As Blitz++ and GSL are becoming an integral part of the SEAL project [4] at CERN, they might emerge as promising alternatives.

The SKIN concept

The different experiments should be free to use different sets of the optional packages. A package may be part of and shipped with VERTIGO; or it may be 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, BELLE skin, and skins for the emerging ILC detectors [10]. Pre-defined skins may easily be selected by the user. Maintenance and distribution of the toolkit will be supported by a CVS repository at HEPHY Vienna.

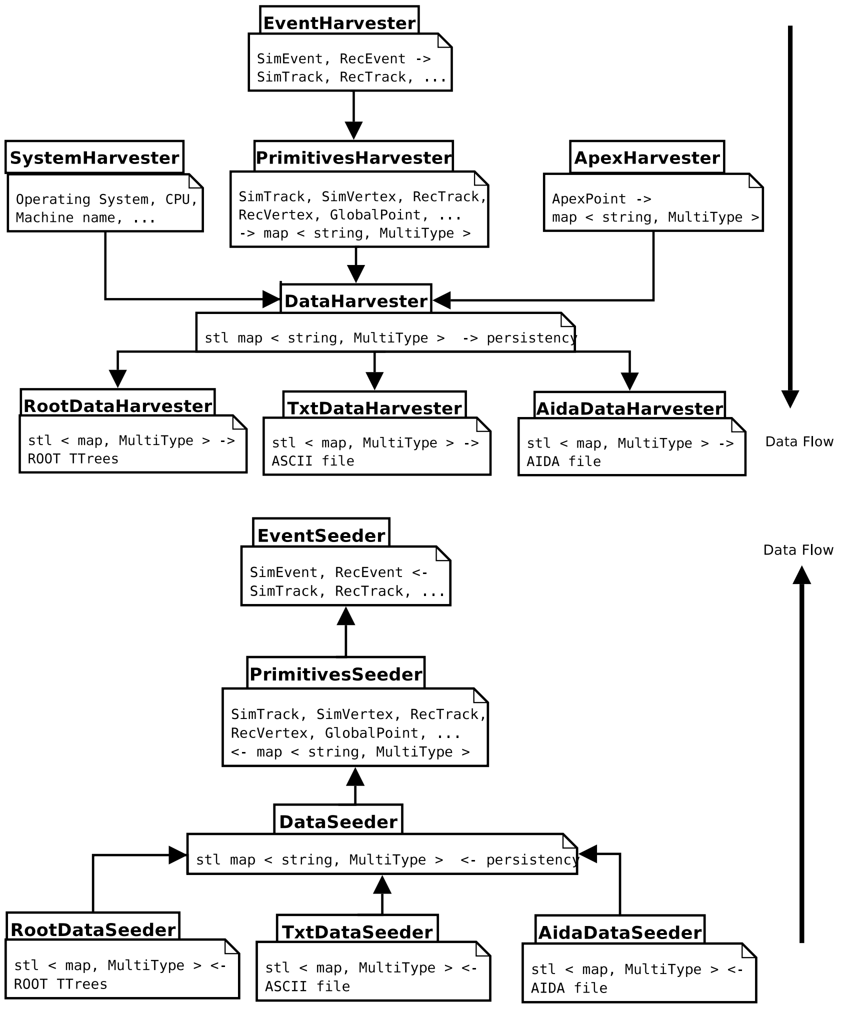

EventHarvester
SimEvent, RecEvent -> SimTrack, RecTrack, ...
SystemHarvester
Operating System, CPU, Machine name, ...
PrimitivesHarvester
SimTrack, SimVertex, RecTrack, RecVertex, GlobalPoint, ... -> map < string, MultiType >
ApexHarvester
ApexPoint -> map < string, MultiType >
DataHarvester
stl map < 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

Figure 2: The data harvester/seeder concept.

CONCLUSIONS AND OUTLOOK

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. Ideas for such a development have been around since long [11], but it is only now that, with the high level of abstraction and generalization of the algorithms required to meet the challenges of LHC and ILC computing, building blocks sufficiently versatile for such a toolkit have come into existence. A similar approach is currently being attempted for track reconstruction (RecPack [12]).

Interests in using the toolkit, once it will be released, have been expressed by CMS, ATLAS, LHCb, BELLE and International Linear Collider (ILC) collaborators, with more expected to follow. Close collaboration among the contributing laboratories is welcome and will be essential for success.

REFERENCES

[1] R. Frühwirth et al., Proc. CHEP 2003, La Jolla (Cal, USA), ed. J. Branson, SLAC-R-636 / eConf C0303241.

[2] V. Innocente et al., Proc. CHEP 2000, Padova (Italy), pp. 56-64, ed. M. Mazzucato, INFN Sezione di Padova.

[3] D.J. Jackson, Nucl.Instr.Meth. A 388 (1997) 247-253.

[4] ROOT, `http://root.cern.ch/`
CLHEP, `http://proj-clhep.web.cern.ch/`
SEAL, `http://seal.web.cern.ch/`

[5] AIDA, `http://aida.freehep.org/` and `http://wwwasd.web.cern.ch/wwwasd/lhc++/AIDA/`

[6] LCIO, `http://lcio.desy.de/`

[7] Coin3D, `http://www.coin3d.org/`

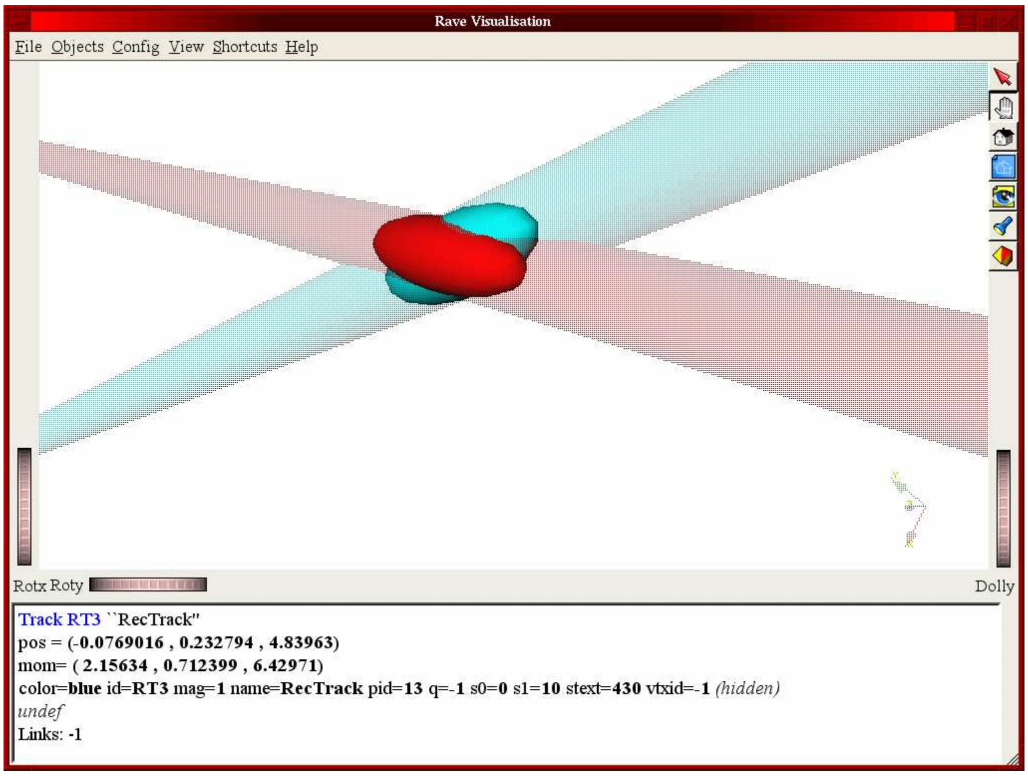
[8] NAG libraries, `http://www.nag.co.uk/`

[9] Blitz++, `http://www.oonumerics.org/blitz/`
FLENS, `http://flens.sourceforge.net/`
GSL, `http://www.gnu.org/software/gsl/`
MTL, `http://www.osl.iu.edu/research/mtl/`

[10] `http://www.desy.de/conferences/ecfa-lc-study.html`
`http://blueox.uoregon.edu/~lc/wwstudy/`
`http://www.interactions.org/linearcollider/`

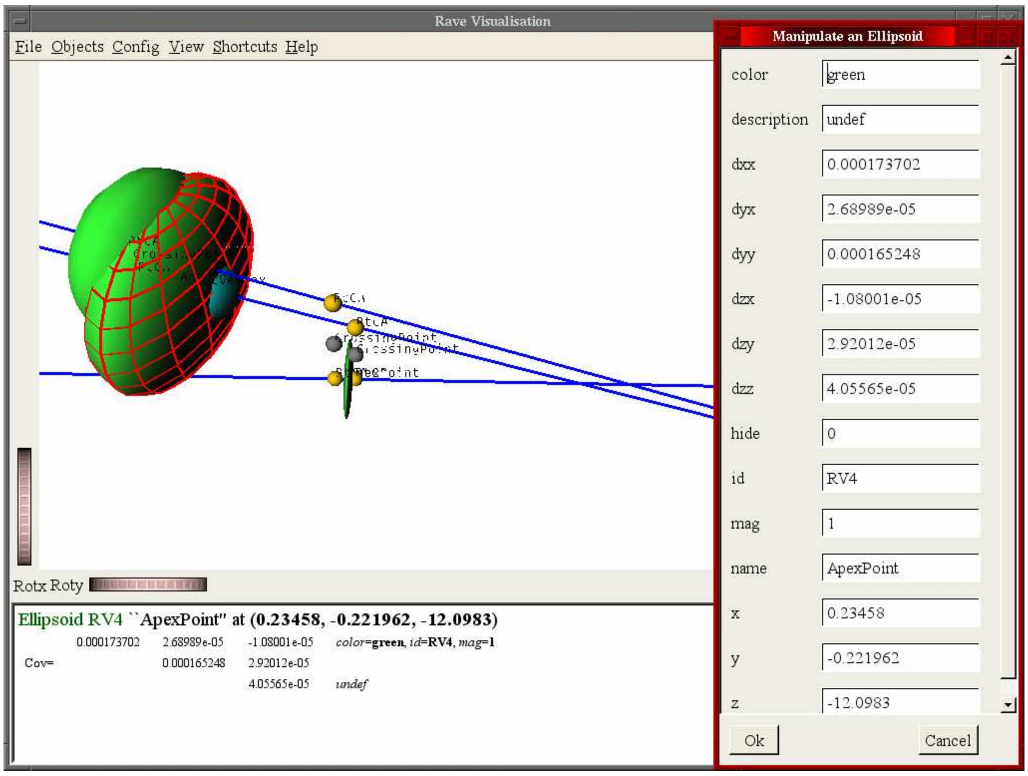
[11] R. Frühwirth et al., Comp.Phys.Comm. 96 (1996) 189-208.

[12] A. Cervera-Villanueva et al., Proc. ATAC 2003, Tsukuba (Japan), ed. T. Kaneko, Nucl.Instr.Meth. A (in print).



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

Figure 3: Snapshot of 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.



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
```

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

Figure 4: Snapshot of 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.