RAVE – a detector-independent toolkit to reconstruct vertices

Wolfgang Waltenberger, Institute for High Energy Physics, Austrian Academy of Sciences, Vienna, Austria.

***Abstract*—A detector-independent toolkit for vertex reconstruction (RAVE = "Reconstruction (of vertices) in Abstract, Versatile Environments") is presented that allows geometric and kinematic reconstruction of vertices. Both linear and adaptive estimation techniques are covered. Non-Gaussianinput data can be handled via the Gaussian-sum technique. Kinematic constraints are taken into account via the Lagrangian formalism. Finally, the toolkit also contains a simple flavor-tagger. Main design goals are ease of use, flexibility for embedding into existing software frameworks, extensibility, and openness. The implementation is based on modern object-oriented techniques, is coded in C++ with interfaces for Java and Python, and follows an open-source approach.**

***Index Terms*—Event Reconstruction, Kalman Filter, Gaussian Sum Filter, Adaptive Method, Kinematic Fitting, Flavor Tagging.**

I. Introduction

EXPERIMENTS at modern high-energy particle colliders rely on precise track and vertex reconstruction which must fully exploit the high spatial resolution achieved by state-of-the-art tracking detectors. Vertex and kinematic reconstruction have played crucial roles in physics analysis in the past twenty years, a prominent example of this being the discovery of the CP violation of B meson decays [1]. The reconstruction of secondary vertices is also important in the context of b-tagging [2], [3].

Data analysis therefore requires new, sophisticated methods beyond the traditional least squares (Kalman filter) estimators, using robust, non-linear, mostly adaptive algorithms.

Vertex fitting in this publication refers to the task of fitting a vertex from a given set of reconstructed tracks, using geometric information only. Vertex finding is in this publication defined as the pattern recognition task; i.e. the task of finding subsets of tracks which share a common point of origin. The term vertex reconstruction implies the vertex finding *and* the fitting step, the result is a set of reconstructed vertices, each containing the information of a position and its 3x3 covariance matrix, plus a set of associated tracks, which are considered to be part of the reconstructed vertex. "Track-to-vertex assignment" refers to the task of identifying the subset of an arbitary set of tracks originating from a given vertex. The assignment can be "hard", in which case a track either belongs or does not belong to a given vertex. It can also be "soft", in which case an assignment probability can be assigned to the vertex-track pair that is under consideration. Kinematic fitting in this publication refers to vertex fitting that takes into account knowledge of the underlying physics process in the guise of constraints. Kinematic reconstruction – in full analogy to the "geometric" case – refers to the reconstruction of not a single vertex but an entire decay tree of particles. Finally, to finish the naming conventions, flavor tagging refers to the task of infering the quark flavor of a given jet by examining the tracks and vertices in the jet.

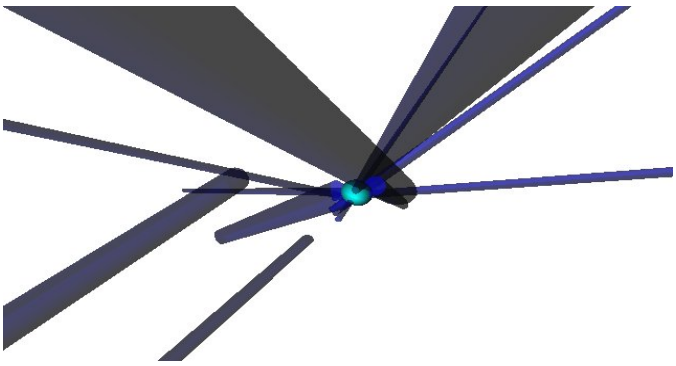

Fig. 1. A primary vertex fitted from an early 2 x 450 GeV collision event at the CMS detector. The large "tubes" represent the reconstructed tracks, the widths representing the positional 2x2 subpart of the covariance matrix. The cyan ellipsoid in the center is the reconstructed vertex; its shape and size corresponds to a 1-σ error ellipsoid, magnified by a factor of 10.

In the event reconstruction chain the first steps such as digitisation and track reconstruction tend to be highly detector-dependent, whereas the last steps, i.e. vertex reconstruction, kinematic fitting, and flavor tagging tend to be physics-driven and detector-independent. It seems thus neither necessary nor desirable for every new experiment to re-code these algorithms from scratch.

RAVE attempts to cover these use cases. It is a detector-independent toolkit that is flexible and usable enough to be integrated into the software frameworks of modern experiments. Originating from the CMS [4] community, it features adaptive vertex fitting algorithms, algorithms to deal with electron tracks, a flexible framework to allow for various kinds of kinematic constraints, and a simple flavor tagger. It has been attempted to minimize the number of assumptions that are hard-coded in the source code. One such assumption is the helix model; it is assumed that a track is geometrically described by a helix in first order approximation. Straight tracks do not pose a problem; however it must be noted that RAVE does not have CPU-efficient algorithms for this special case. More specialised algorithms could be faster, though not more precise.

Apart from CMS, RAVE code has already been run within the software frameworks of ILD [5], SiD [6], 4th concept [7], and Belle-2 [8], [9]. Recently, CMS has published vertexing

results on real data [10]. The vertex reconstruction code is identical between CMS and RAVE.

Fig. 1 depicts the result of a vertex reconstruction algorithm. In this example only a single vertex has been reconstructed.

II. History and status of vertex reconstruction in High Energy Physics

The most straightforward method for vertex fitting is a global least squares method [11], [12], though the estimator can also be formulated more efficiently as a Kalman filter [12], [13], which has become the de-facto standard since the LEP era. With the Kalman formalism, track smoothing – the refitting of the track parameters at the vertex – was introduced. Very soon, the experimental needs asked for dealing with non-Gaussian parameters and mis-measured and outlying tracks. For non-Gaussian parameters, the Gaussian-sum technique was considered [14], protection against outlying or mis-measured tracks was introduced with robust techniques such as the M-estimator [15] or trimming techniques [16], [17]; later the adaptive technique was established [16], [18], [19]. Precision measurements required algorithms which take into account not only geometric information but also the knowledge of physics properties such as particle masses. The Lagrangian formalism [20] proved to be an adequate framework for formulating kinematic fits [21]. Secondary vertex finding is often performed in the context of b-tagging. A general problem with this task is the fact that B mesons decay to D mesons with a high branching ratio, giving rise to two decay vertices and low track multiplicities in either vertex. This is often remedied with the *ghost track* formalism [22], [23].

RAVE includes all of the algorithms, except for the M-estimator, and the ghost track algorithm, which can be implemented by the user outside of RAVE.

A summary of the mathematics of both linear and non-linear methods has recently been published [24].

III. The software environment of the HEP community

The high energy physics community has a long tradition of writing experiment-independant software toolkits and programs. Examples can be found in the field of event generation (e.g. PYTHIA [25] or MADGRAPH [26]), event simulation (GEANT4 [27]), general-purpose frameworks (ROOT [28]), and libraries (e.g. CLHEP [29]), visualisation, and phenomenology (e.g. SOFTSUSY [30]). In fact, in analogy to the sourceforge [31] project, the high energy physics community has its own HEPFORGE project [32]. Despite the respectable successes of projects that have created de-facto standards, such as GEANT4, the situation in the event reconstruction world seems to be less standardized. While there exist standalone projects for jet clustering (SISCONE [33]) and track reconstruction (RECPACK [34], GENFIT [35]), the large experiments seem to tend to implement their own unique solutions. For vertex reconstruction and kinematic fitting, no attempts for detector-independant toolkits other than RAVE are known to the author. All of the LHC experiments developed individual solutions within their frameworks [3], [16], [36], [37].

IV. Software Analysis and Design

It is the aim of the toolkit to address typical vertex and kinematic reconstruction tasks of collider experiments. RAVE itself is decomposed into two parts whose design and domain analysis have to be discussed individually:

- RAVE's user interface classes – which are comprised of the data and algorithm classes that are visible to the user. Many, if not most of these classes can be interpreted as *adaptors* [38]
- RAVE's algorithmic code base originates from the CMS community and is kept 100 % source compatible with the CMSSW vertexing subsystem. It thus has to obey the coding conventions defined by the CMS software community. Only minor adaptations were made in the CMSSW code base for better integrability in RAVE. The design of this code is touched only marginally in this publication; it is better described elsewhere [16].

By design, the user interface classes must fulfill the following requirements:

(a) Embeddability – the toolkit must be easily embeddable in different environments, possibly even in languages other than C++. A few constraints on the design follow from this goal – e.g. the technique of operator overloading has been avoided in order to facilitate embedding the toolkit in Java.
(b) Encapsulation – the user interface classes are to "shield" the user from any changes that occur within the algorithmic code base which is shared with the CMS software framework and is thus not under full control of RAVE's authors. Thus, not a single CMSSW class is exposed to the user.
(c) Accessibility – all features of the toolkit must be accessible from the user interface, without violating (a) or (b). A string-based algorithm configuration mechanism (Sec. V-C) has been implemented to satisfy these needs.
(d) Simplicity – the user interface is to be kept as simple as possible, while maintaining (a), (b), and (c).

Specifically, the following design has been developed for RAVE: For the task of geometric vertex reconstruction there exist only two main data classes, in accordance with the design criteria stated above. For, e.g., the task of geometric vertex reconstruction, there are only two main data classes: the `rave::Track` class which represents a reconstructed track, i.e. a track state and its covariance matrix, and the `rave::Vertex` class which contains the information of the position of the reconstructed interaction vertex, its covariance matrix, and a list of tracks that were used to reconstruct the vertex. The track parameters can be given in pseudo-Euclidean $(\vec{x}, \vec{p})$ parametrization with a six-dimensional covariance matrix of rank five. Alternatively it can also be given in five-dimensional perigee parametrization [39], [40] with a regular, five-dimensional covariance matrix, given also in perigee parameters.

For the purpose of kinematic fitting, these classes have been extended via inheritance; there exists a `rave::KinematicParticle`, and a `rave::KinematicVertex`. Additionally, kinematic

reconstruction produces the information of a decay tree. This information is represented in the data structure of a `rave::KinematicTree` which contains a graph that has `rave::KinematicParticles` as its edges and `rave::KinematicVertices` as its nodes. The "data classes" and their interrelationships are depicted in Figs. 2, 3.

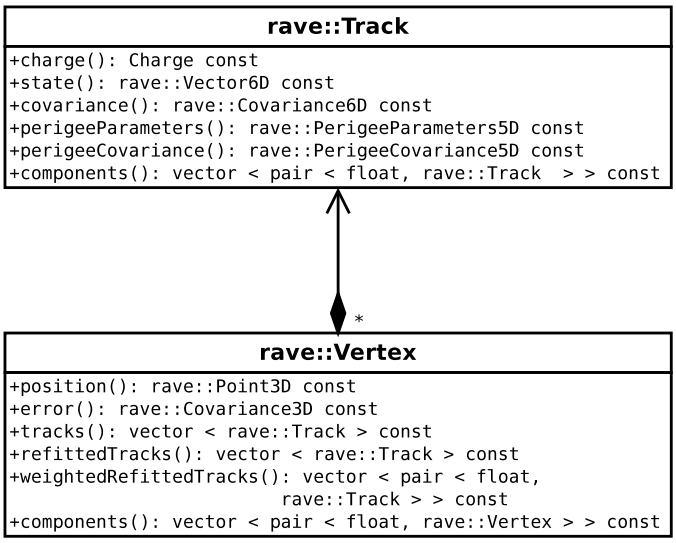

Fig. 2. The main classes that represent the input and the output of RAVE vertex reconstruction. Reconstructed tracks are the input for vertex reconstruction. The track parameters can be provided in a 6-dimensional pseudo-Euclidean $(\vec{x}, \vec{p})$ parametrization, together with a 6-dimensional covariance matrix of rank 5. Alternatively, the user can also provide perigee parameters with a 5-dimensional covariance matrix of full rank. A reconstructed vertex is described by an spatial point along with its covariance matrix. Additionally, the fitted χ^2, and a list of input tracks and their probabilities of vertex assignment are stored. The full meaning of the assignment probabilities will be discussed with the *adaptive* method, see Sec. V-A.

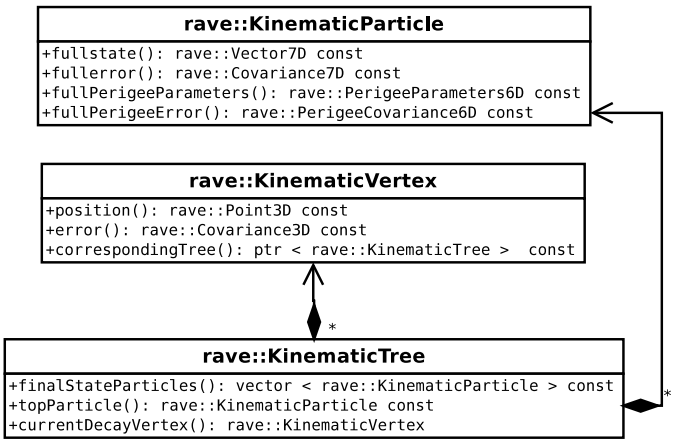

Fig. 3. Input and output classes of kinematical fitting. A KinematicParticle is a reconstructed track plus a mass hypothesis, resulting in a 7-dimensional (pseudo-Euclidean) or 6-dimensional (perigee) particle "state". Additionally, a class representing the decay chain is introduced: the KinematicTree. This tree has the particles as its edges, and the reconstructed KinematicVertices as its nodes, representing a full decay chain.

For every task that RAVE addresses, there exists a separate `rave::Factory`: the geometric vertex reconstruction algorithms can be accessed via `rave::VertexFactory`, while kinematical problems can be addressed via the `rave::KinematicTreeFactory`. Finally, flavor tagging is done via the `rave::FlavorTagFactory`. Although technically they are not singletons, they are intended to be instantiated only once by the user. Fig. 4 lists all current RAVE factories and their most important member functions.

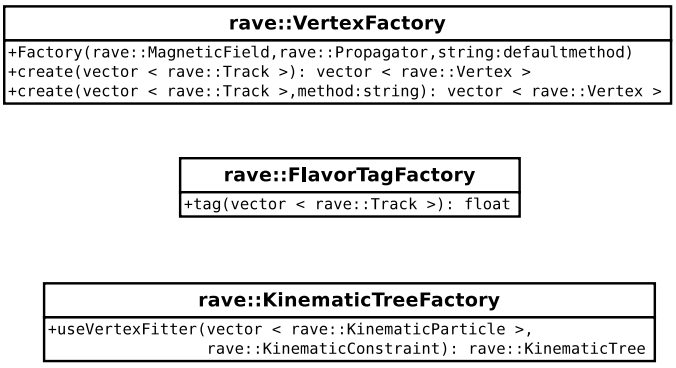

Fig. 4. All RAVE factories, and their main member functions. The factories implement the interface with the user. In the case of geometric vertex reconstruction, the user supplies the tracks to the factory, and defines the algorithm that should be used. The reconstructed vertices are then returned to the user by the factory.

V. Algorithms in RAVE

This section gives an overview of the algorithms and presents potential use cases for them. More technical details may be found in the RAVE online manual [41].

A. Vertex reconstruction

RAVE has a lengthy list of vertex reconstruction algorithms, each one serving a different need. The different algorithms will be presented, and an overview over all algorithms and their typical use cases will be given thereafter.

Kalman filter: The classical vertex fitting algorithm is the Kalman filter, which is a linear, least-squares method [13]. The pattern recognition problem of finding a consistent subset of tracks is in the following assumed to be solved.

Assume that a vertex candidate $\mathbf{v_{i-1}}$ and its error matrix $\mathbf{C_{i-1}}$ have been fitted with tracks $1 \dots (i-1)$. Assume also track i to have the estimated track parameters $\tilde{\mathbf{q}}_i$ and track momenta $\mathbf{p_i}$. Let its covariance be denoted with $\mathbf{V_i} = \mathbf{G_i^{-1}}$ and the approximated track model be

$$\mathbf{q_i} \approx \mathbf{c_i} + \mathbf{A_i v} + \mathbf{B_i p_i}$$

.

The update equations now read:

$$\begin{aligned}
\tilde{\mathbf{v}}_i &= \mathbf{C_i}\left[\mathbf{C_{i-1}^{-1}}\tilde{\mathbf{v}}_{\mathbf{i-1}} + \mathbf{A_i^T G_i^B}\left(\tilde{\mathbf{q}}_\mathbf{i} - \mathbf{c_i}\right)\right] \\
\tilde{\mathbf{p}}_i &= \mathbf{W_i B_i^T G_i}\left(\tilde{\mathbf{q}}_\mathbf{i} - \mathbf{c_i} - \mathbf{A_i}\tilde{\mathbf{v}}_\mathbf{i}\right) \\
\mathrm{var}\left(\tilde{\mathbf{v}}_i\right) = \mathbf{C_i} &= \left(\mathbf{C_{i-1}^{-1}} + \mathbf{A_i^T G_i A_i}\right)^{-1} \qquad (1) \\
\mathrm{var}\left(\tilde{\mathbf{p}}_i^n\right) &= \mathbf{W_i} + \mathbf{W_i B_i^T G_i A_i C_i A_i^T G_i B_i W_i} \\
\mathrm{cov}\left(\tilde{\mathbf{p}}_i^n, \tilde{\mathbf{v}}_i\right) &= -\mathbf{W_i B_i^T G_i A_i C_i}
\end{aligned}$$

As we can see, the recursive Kalman filter needs to invert only matrices of low rank: the method is very fast, compared with the global fit. Evidently, correlations between tracks can not be taken into account with this method.

Internally, the algorithm always uses perigee parameters, independent of the parametrization of the input tracks. A second parametrization has been attempted that linearizes in Euclidean space [42]. It has been found inferior to the perigee parametrization. This has been expected, considering that a helix is linear in perigee parameters, and non-linear in Euclidean parameters. The standard track model is a helix model, although in the fitting process the tracks are relinearized if the vertex candidate is too far from the original linearization point of the track, thus taking into account deviations of the track from the helix model. Fig. 5 shows the invariant masses of reconstructed J/ψ resonances fitted with the Kalman filter in Belle-II. As it is RAVE's least squares method, it is the optimal choice in the absence of mis-measured or outlying tracks. In the presence of mis-measured or mis-associated tracks the Kalman filter is biased, see Fig. 6. This method is available in RAVE as the `kalman` method.

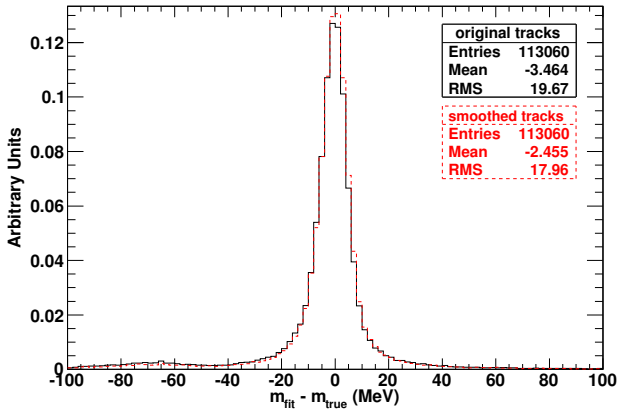

Fig. 5. Mass of a J/ψ resonance, reconstructed from two muons, fitted with the Kalman method within the Belle-II framework ("smoothed tracks" – red, dashed), comparing with the invariant mass of the "original track" pair (black, solid).

Adaptive technique: For CMS, adaptive algorithms have been formulated such as the adaptive fitting technique [18]. This set of algorithms introduces the concept of *soft assignment*; a track is associated to a specific vertex with an assignment probability, or "weight" w. The fitter is then implemented as an iterated, reweighted Kalman filter: in every iteration step new track weights are computed and subsequently for the determination of a vertex candidate.

This algorithm solves the vertex fitting and the track-to-vertex assignment problems; the pattern recognition problem is only partly solved: while incompatible tracks are automatically discarded in this algorithm, only one vertex is fitted by the algorithm. Note that it is the first algorithm presented in this paper which features a "soft" track-to-vertex assignment. The weight is a function of the track-to-vertex χ^2:

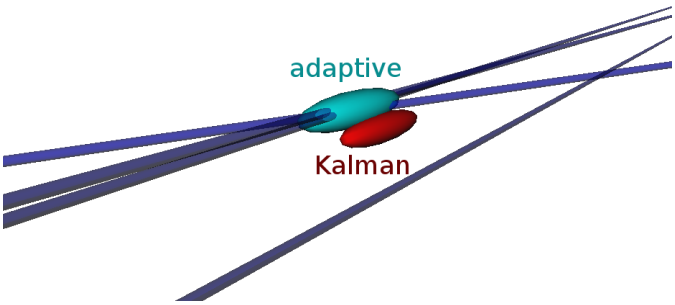

Fig. 6. Reconstruction of a CMS Monte-Carlo $J/\psi \rightarrow K^+K^-\mu^+\mu^-$ decay vertex, where one of the tracks is mis-measured. The estimate of the Kalman filter is drawn towards the mis-measured track, the estimate of the adaptive method is less influenced by it.

$$w = w\left(\chi_i^2\right) = \frac{\phi\left(\chi_i^2\right)}{\phi\left(\chi^2_{\text{cutoff}}\right) + \phi\left(\chi_i^2\right)} = \frac{e^{-\chi_i^2/2T}}{e^{-\chi_i^2/2T} + e^{-\chi^2_{\text{cutoff}}/2T}} \tag{2}$$

Here, the user-defined cutoff criterion χ^2_{cutoff} defines the point for which $w\left(\chi_i^2\right) = 0.5$. The exact definition of the track-to-vertex χ^2 is given in [43]. Additionally, the "temperature" parameter T defines the "softness" of the weight function. An annealing scheme is employed – T is lowered in each iteration step – in order to avoid falling prematurely into local minima of the objective function. Fig. 7 shows the weight function, as a function of the track-to-vertex χ^2 value and the annealing parameter T. A good choice for χ^2_{cutoff} is 9. In this case, about 1 % of the "inlying" tracks are assigned a weight < 0.5. Similarly, for $\chi^2_{\text{cutoff}} = 16$ results in 0.03 % of all inlying tracks being assigned a weight < 0.5. This number rises to 13 % for $\chi^2_{\text{cutoff}} = 4$. The bias due to the mismeasured track in Fig. 6 is significantly reduced when the adaptive technique is used.

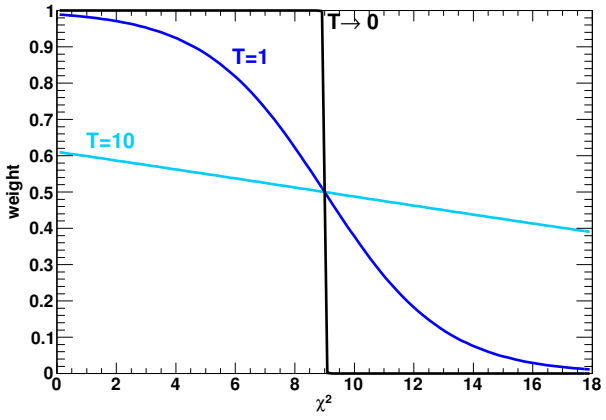

Fig. 7. Assignment probability w, as function of χ^2, for different temperatures T. χ^2_{cutoff} has been chosen to be 9, in this example.

Fig. 8 shows the distribution of primary vertices in the CMS experiment with early $\sqrt{s} = 450{+}450$ GeV collision data. The

primary vertices have been fitted with the adaptive technique. RAVE makes this technique available to its users under the name `avf`.

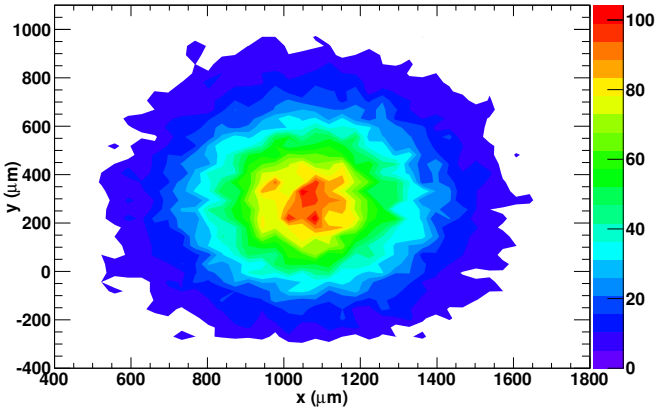

Fig. 8. A profile histogram, reconstructing the CMS beamspot with tracker data from an early $\sqrt{s} = 900$ GeV run. Values are given in percentage of maximum height.

Iterative adaptive technique: Using the adaptive method iteratively yields a powerful method that has become the default general-purpose geometric vertex finding method in CMS: the adaptive vertex reconstructor (`AVR`) [44]. Given a set of tracks, a vertex is fitted adaptively. After the fitting, all tracks with an assignment probability of less than 50 % are put into a new set of tracks, with which a subsequent vertex is fitted adaptively. The procedure is repeated until no new vertex is found. Naturally, this algorithm solves the full vertex reconstruction problem. Fig. 9 shows a $b\bar{b}$ pair that hadronizes into $\mathrm{B}^0(\mathrm{D}^0\tau\nu_\tau)\mathrm{B}^0(\mathrm{D}^0\mu\nu_\mu)$ with multiple reconstructed vertices. The event is from a CMS Monte Carlo production. The method name in RAVE for this algorithm is `avr`.

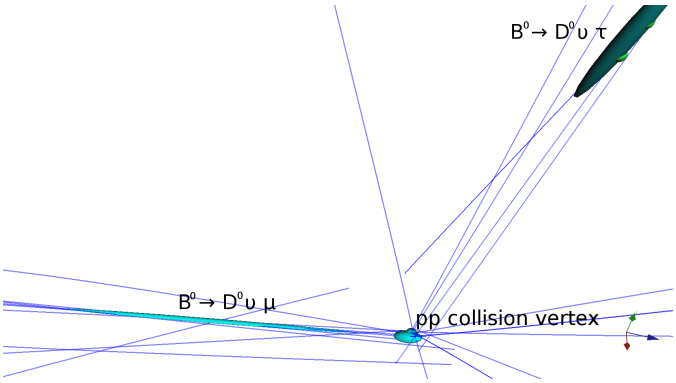

Fig. 9. $b\bar{b} \to \mathrm{B}^0(\mathrm{D}^0\tau\nu_\tau)\mathrm{B}^0(\mathrm{D}^0\mu\nu_\mu)$ with several reconstructed vertices, taken from a CMS Monte Carlo sample.

Trimmed Kalman filter: An alternative to the adaptive technique is the trimmed Kalman filter. From a given set of tracks, a vertex is fitted with a Kalman filter. The least compatible track is discarded from the vertex and the vertex is refitted. This step is iterated until no incompatible track is found. It is a full reconstruction algorithm in that all subtasks are solved. While this is a simple and robust algorithm, its performance is worse than that of the adaptive technique – the additional information encoded in the track weights indeed translates into an improved performance. The adaptive technique should thus be the preferred algorithm. Fig. 10 compares a few algorithms, Tab. I shows the performance differences in numbers. Still, RAVE knows of this algorithm as the `tkvf` method.

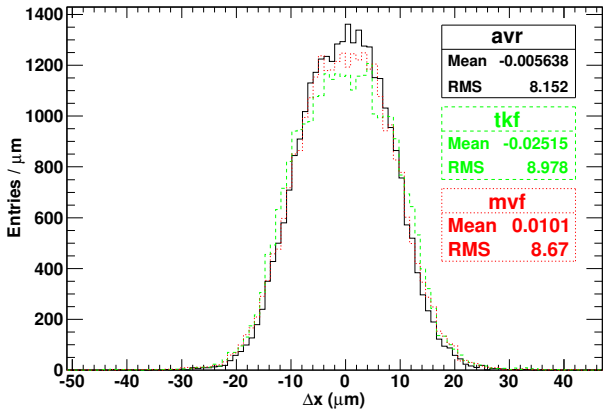

Fig. 10. Resolution histogram that compares the iterative adaptive technique (AVR), the trimmed Kalman filter (TKF), and the multivertex fitter (MVF), with 900 GeV CMS data.

TABLE I
Comparison of three general-purpose vertex reconstructors, applied to unfiltered 900 GeV CMS data. Δr is the transverse distance between the vertex and the beamspot which was reconstructed *a posteriori* from the reconstructed vertices. The algorithms have been run on an 64-bit Intel Linux machine.

method	# vertices, $\Delta r < 1$cm	# vertices, $\Delta r < 1$mm	# vertices, $\Delta r < 100\mu$m	t [ms]
AVR	2070 K	1370 K	119 K	110
TKF	1579 K	1129 K	92 K	1650
MVF	2113 K	1667 K	114 K	476

Multivertex fit: Another generalisation of the adaptive method is the multivertex fitter. This algorithm fits n vertices at once. The vertices "compete" for the tracks, in the sense that the sum of the track-to vertex associations of a single track cannot become greater than one. This is achieved by implementing the following weight function:

$$w_{ij} = \frac{e^{-\chi^2_{ij}/2T}}{e^{-\chi^2_{\mathrm{cutoff}}/2T} + \sum_{k=1}^{n} e^{-\chi^2_{ik}/2T}} \tag{3}$$

The multivertex fitter needs a "seeding algorithm": a solution to the vertex finding problem needs to be available in order for the multivertex fitter to be functional. The fitter itself then iterates on this solution. Thus, the vertex fitting and the track-to-vertex-assignment tasks are solved by the algorithm. The vertex finding problem is delegated to another algorithm, the solution that is found is then iterated upon. Technically, any other vertex finder or vertex reconstructor can be used as a seeding algorithm. To this date, no realistic use case was found in which the multivertex fitter performs better than the iterative adaptive technique, see Fig. 10. The algorithm is made available to RAVE users as the `mvf` method.

Gaussian-sum fitting: Electrons likely emit bremsstrahlung photons when traversing matter. The resulting tracks deviate significantly from the helix model – the transverse radius of the track after the emission is smaller than the one before. The Gaussian-sum track fitter [45] is a track fitting method that accounts for such effects. The resulting trajectory is composed of multiple trajectories each of which is assigned a weight. The sum of the weights must equal 1. RAVE has an algorithm which fully exploits the information of a multi-trajectory track in the vertex fitter: the Gaussian-sum vertex filter (GSF) [14]. Figs. 11 shows a comparison between Kalman filter and Gaussian sum filter, for "non-physical" data, i.e. tracks that are generated without taking kinematic constraints (e.g. energy conservation at the production vertex) into account. These non-physical tracks were produced with RAVE's own toy Monte Carlo framework called "vertigo" (see Sec. VII). They are typically used to study algorithmic properties of a method. The Gaussian sum fitter truly is a fitter – the vertex finding issue is not addressed. RAVE's name for this method is `gsf`.

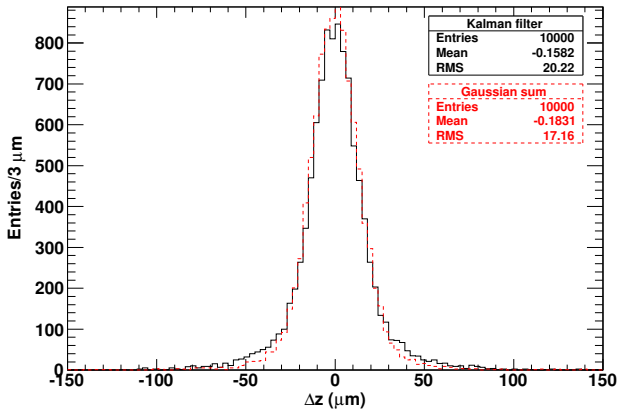

Fig. 11. Electrons fitted to a vertex, with the Kalman filter (black) and the Gaussian sum filter (red, dotted). Residuals in the z coordinate, for "non-physical" data.

Adaptive Gaussian-sum fitting: The Gaussian-sum technique can be combined with an adaptive fitter; e.g. several electrons can be fitted adaptively to a common vertex, taking into account all track components of the electrons. This algorithm is available in RAVE as the AGSF method. To this day, no realistic use case has been found for this algorithm. The Adaptive Gaussian-sum fitter is another vertex fitter that implements soft assignment. It is made available in RAVE as the `agsf` method.

Summary of vertex reconstruction algorithms: All of RAVE's algorithms are formulated in the Kalman formalism, making the Kalman vertex fitter the mother of all other vertex reconstructors. As such, the standard Kalman vertex fitter implements a least-squares technique, making it the ideal algorithm in the case of ideal (i.e. not mis-measured) tracks in the absence of outlying tracks. If the user wants to fit three ideal tracks to a single vertex, assuming that all tracks share the same vertex, then the Kalman method is the correct one. All other methods address different violations of the above assumptions. In the presence of track mis-measurements or outlying tracks, we consider the adaptive technique to be our best general-purpose algorithm. Additionally, it also solves the track-to-vertex association task. If more than one vertex needs to be reconstructed, the iterative adaptive technique is the preferred algorithm. In the case of non-Gaussian track errors, as e.g. is often the case with electrons, one should look at the Gaussian-sum techniques.

Algorithms known in literature not covered by RAVE: A few algorithms exist in literature that are not implemented in RAVE. These are the ones known to the author:

- Zvtop, or topological vertex finder [46] operates by constructing a "Gaussian tube" $f_i(\vec{r})$ around track i. A "vertex probability" is then given as
$$V(\vec{r}) \propto \left(\sum_{i=1}^{N} f_i(\vec{r})\right)^2 - \sum_{i=1}^{N} f_i^2(\vec{r}).$$
An inclusion of such an algorithm in RAVE is desirable.
- Ghost track algorithms. A "ghost" track is introduced that points from the primary vertex in the direction of e.g. jet axes, enabling the user to find "one-prong" decay modes, as may be interesting in the context of finding a B meson decaying into a D meson. [22], [23]. No inclusion in RAVE is foreseen, because much of the "ghost track functionality" can easily be implemented externally, in the various experiment-specific environments, using RAVE for performing the vertex reconstruction itself.
- Many fast methods are described in the literature, e.g. [39], [42]. Various assumptions simplify the fitting procedure, resulting in a faster algorithm. Because they imply more specific assumptions, they are not scheduled for inclusion in RAVE.

Track selection is considered to be the user's responsibility and is thus not covered by RAVE.

B. Features in vertex reconstruction

Beamspot constraint: A beamspot constraint can be added to the algorithm and will be treated as a Bayesian *a priori* information of the vertex position. The resolution of the primary vertex position can be significantly improved by such a constraint. Fig. 12 shows the distribution of the reconstructed primary vertices in a simulated ILD sample, with and without exploiting the information of the beamspot constraint. The widths (in x and y) of the beamspot constraint are expected to be known to the sub-micron level at the ILC.

Track refitting: The information of a reconstructed vertex can be used to constrain the track parameters to the vertex. This track refitting is sometimes also called track smoothing, and appears naturally in the Kalman formalism. Track smoothing can also be generalised to adaptive techniques. As in the case of the Kalman fitter, all smoothed tracks pass through the vertex position, independent of their track weights. The user has to decide if a smoothed track with e.g. a track weight < 0.5 carries any useful additional information. Fig. 13 shows how track refitting improves some track parameters in the context of CMS tracking. The beamspot constraint indirectly further adds to the performance improvement.

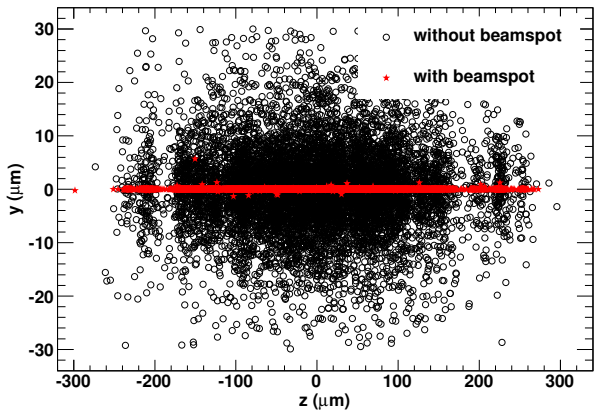

Fig. 12. Reconstructed primary vertices in ILD, without (black circles) and with (red stars) taking the beamspot constraint into account. The width of the beamspot is expected to be at the sub-micron level; it was taken to be 190 nm (y coordinate) in this study, the length of the beamspot was 100 μm.

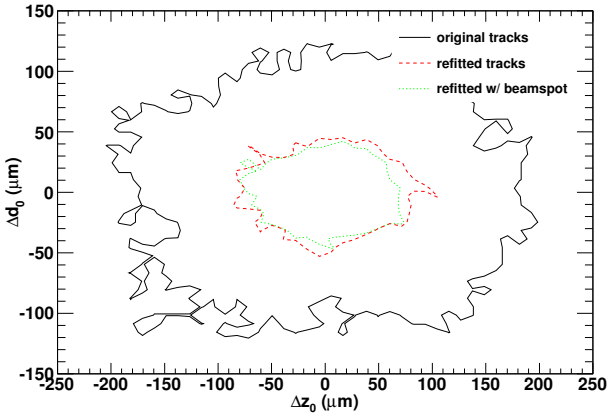

Fig. 13. Resolutions of the "transverse impact point" versus the "longitudinal impact point" of "original" (black) and refitted (green and red) tracks in CMS. Contours are at 5% of maximum value.

Linearization point finding: Before being able to fit a vertex with e.g. the Kalman filter, a robust *linearization point* needs to be found around which the tracks are linearized. This linearization point also serves as the first vertex candidate. In the process of fitting, tracks are re-linearized if the vertex candidate moves too much. Despite this "self-correcting" procedure, a good linearization point is important both for the overall speed (re-linearization is costly) as well as for the overall numerical stability of the algorithm. The default algorithm (the "fraction-of-sample mode with weights" – FSMW) has been shown to exhibit a good performance in a large diversity of use cases [43], such that the user usually does not have to take a choice. However, it should be at least brought to attention that the user *can* choose from a list of algorithms which are listed and described in [43].

C. Configuration of algorithms

Since the user has the possibility to choose from several vertex fitting and vertex reconstruction algorithms, and the algorithms have user-configurable parameters, he or she needs to be able to specify which algorithm to use, and with what configuration. Since by design the user is exposed to no implementation details (see Sec. IV), a simple ASCII string functions as the interface by which the user both chooses and configures the algorithm. The syntax of the string is defined as follows:

```
algorithm-par1:val1-par2:val2-...
```

Here `algorithm` is the name of the algorithm, whereas `parX` is a configurable parameter specific to the algorithm, which the user wants to take on the value `valX`. The algorithms define what parameters they take (and, also, the C++ types of the values).

The algorithms can also be nested, to allow for configuration of sub-algorithms:

```
algorithm-par1:val1-subalgo:(parS:valS-...)
```

A typical configuration string is

```
avf-Tini:256-ratio:0.25-sigmacut:3
```

resulting in an `AdaptiveVertexFitter` with $\chi^2_{\text{cutoff}} = 3^2 = 9$, an annealing ratio $r = 0.25$, and an initial temperature of $T_{\text{ini}} = 256$. Note that algorithms are required to define default values for all parameters.

The multi vertex fitter (see Sec. V-A) is an example for a nested configuration:

```
mvf-ini:(finder:avr)
```

In this case, the `AdaptiveVertexReconstructor` with default parameter settings is used as a `Seeder` for the `MultiVertexFitter`.

D. Flavor tagging

Heavy flavor jet identification plays an important role in many experiments [47], [48], [49], [50]. The presence or absence of a secondary vertex can be used to identify – to "tag" – heavy flavor jets.

RAVE also contains a simple flavor tagger that exploits vertex information only; it can be used to perform binary discrimination only (e.g. b versus non-b); discriminating between light flavors, c, and b is not possible. Also, it cannot be competitive with state-of-the-art multivariate techniques, but can have its application e.g. as a baseline algorithm, or for evaluating rough performance expectations. Also, it is very robust against misalignment, making it an interesting candidate for "early" data [51], [52]. The discriminator value d of the tagger is a simple function of the reduced distance between primary vertex and secondary vertex:

$$d = \ln(1 + s) \tag{4}$$

where s can be configured to be either the flightpath (divided by "[cm]"), or the flightpath significance, computed with or without the information from the z coordinate. Optionally, the Lorentz boost can be taken into account: $s \to s' = \frac{s}{\gamma}$.

E. Kinematic fitting

Kinematic fitting refers to the idea of using known properties of a specific decay chain to improve the measurements describing the process. Lagrangian multipliers are used to impose the kinematic constraints on the fit [21]. If $\mathbf{y} = (\mathbf{y_1}, \mathbf{y_2}, \cdots, \mathbf{y_n})$ denotes uncorrelated measurements and $\mathbf{V_y} = \mathbf{diag}(\mathbf{V_1}, \mathbf{V_2}, \ldots, \mathbf{V_n})$ is the covariance matrix of the measurements, then one minimizes

$$\chi^2 = (\mathbf{y}_{\mathrm{ref}} - \mathbf{y})\, \mathbf{V_y^{-1}}\, (\mathbf{y}_{\mathrm{ref}} - \mathbf{y})^T\,, \tag{5}$$

with the constraints

$$\mathbf{h}\,(\mathbf{y}_{\mathrm{ref}}) = \mathbf{0} \tag{6}$$

where $\mathbf{y}_{\mathrm{ref}}$ denotes the "improved" measurements. If the constraint equations are now linearized around at starting value $\mathbf{y}_{\mathrm{ini}}$, then

$$\mathbf{h}\,(\mathbf{y}_{\mathrm{ini}}) + \left.\frac{\partial \mathbf{h}}{\partial \mathbf{y}}\right|_{\mathbf{y}_{\mathrm{ini}}} (\mathbf{y}_{\mathrm{ref}} - \mathbf{y}_{\mathrm{ini}}) = \mathbf{d} + \mathbf{D}(\delta \mathbf{y}) = \mathbf{0} \tag{7}$$

Here we have used

$$\begin{aligned} \delta \mathbf{y} &\equiv \mathbf{y}_{\mathrm{ref}} - \mathbf{y}_{\mathrm{ini}} \\ \mathbf{d} &\equiv \mathbf{h}\,(\mathbf{y}_{\mathrm{ini}}) \\ \mathbf{D} &\equiv \left.\frac{\partial \mathbf{h}}{\partial \mathbf{y}}\right|_{\mathbf{y}_{\mathrm{ini}}} \end{aligned} \tag{8}$$

The function that needs to be minimized reads:

$$\chi^2 = (\mathbf{y}_{\mathrm{ref}} - \mathbf{y})\, \mathbf{V_y^{-1}}\, (\mathbf{y}_{\mathrm{ref}} - \mathbf{y})^T + 2\lambda^T\, (\mathbf{D}(\delta \mathbf{y}) + \mathbf{d}) \rightarrow \min. \tag{9}$$

RAVE supports several kinds of such constraints, e.g. constraints on the invariant mass at the vertex. A detailed description of the capabilities can be found in [53], a sample ILC application is documented in [54]. Fig. 14 shows an artificial example, where two muons are fitted to the decay vertex of a Z boson. The mass of the vector boson has been used to improve the vertex resolutions. Fig. 15 shows a more realistic ILC simulation, taken from [53]: the $e^+e^- \rightarrow W^+W^- \rightarrow 4$ jets channel has been used to reconstruct the W masses, using the known four-vector of the W^+W^- system as a constraint in the fit. Given that $\sigma\,(\bar{m}_W) = \frac{\sigma(m_W)}{\sqrt{N}}$, this Monte Carlo analysis measures

$$\bar{m}_W = 80.42 \pm 0.32\,\mathrm{GeV}/c^2 \tag{10}$$

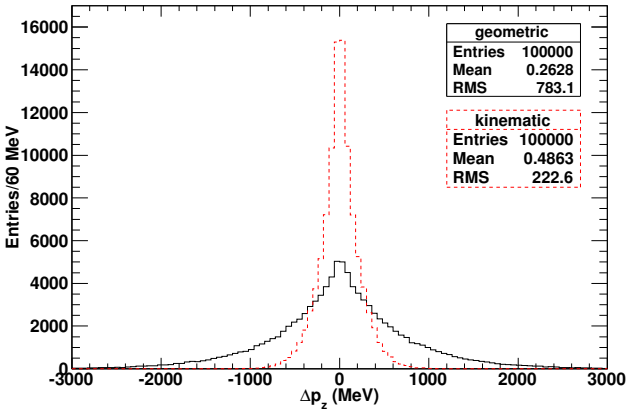

Fig. 14. p_Z resolution of Z decaying into $\mu\mu$, detector simulated within a dummy framework

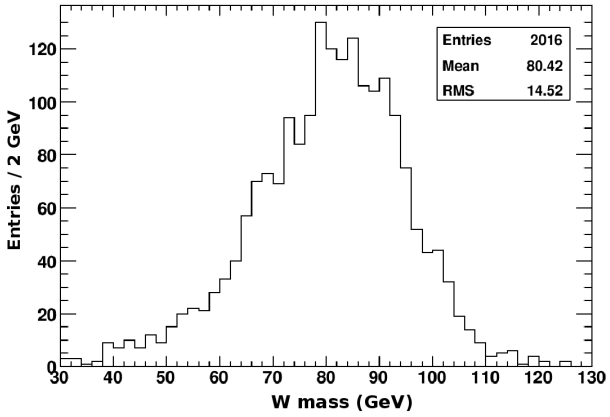

Fig. 15. Reconstructing the W mass from an ILC simulation: $e^+e^- \rightarrow W^+W^- \rightarrow 4$ jets, using an equal-mass constraint. Plot taken from [53].

VI. RAVE'S INTERNAL STRUCTURE

The classes and interfaces that are specific to RAVE compose a software layer that wraps around code that is 100% source compatible with the CMS software framework. Basic knowledge of the software infrastructure that is used by vertex reconstruction algorithms is explained in this section. For a more detailed description see [16].

Fig. 16 shows the basic abstract interfaces of the CMS vertex reconstruction software. A `LinearizationPointFinder` is an algorithm which finds a point in the vicinity of the expected vertex position, given a set of tracks. Such algorithms are used to find points around which the tracks can be linearized. Also, for adaptive methods, such a linearization point is used as a "seeding" point. The first set of track weights (Eq. 2) are computed using these points as the first vertex candidates. `VertexFitters` are algorithms which compute a vertex from a set of reconstructed tracks. `VertexRecontructors` reconstruct n vertices from a set of reconstructed tracks, solving also the pattern recognition problem of vertex finding. Technically they may or may not delegate the task of vertex fitting to a `VertexFitter`. Alternatively, a `VertexReconstructor` is free to reconstruct its own vertex fitting procedure, although the first strategy is probably more sensible. Finally, the figure depicts also `AbstractConfReconstructors`; these are `VertexReconstructors` which can be given a "configuration" object which details how the algorithm is to be used. CMS-specific technical issues are the reason behind the split between the `VertexReconstructor` and the `AbstractConfReconstructor` interface.

Algorithm developers can greatly benefit from the hierarchical structure of the algorithm classes. E.g. an algorithm for a novel vertex finder can be combined with existing `VertexFitters` to build a `VertexReconstructor`.

If an algorithm is to be made available to the user, it has to register itself at the `VertexRecoManager`. A simple template class called `ConfRecoBuilder` facilitates the registration, see Fig. 17. The reconstructors are *self-registering* – the `ConfRecoBuilders` are instantiated in the algorithm code itself. Thus, loading a library at runtime (e.g. via `dlopen`) that contains reconstructors augments the list of algorithms that are available in RAVE.

For an algorithm developer, the main input class is the `TransientTrack`. Apart from containing the information of the track state accessible in various parametrizations, it also contains information such as the goodness-of-track-fit χ^2 value, the corresponding number of degrees of freedom, and the track charge. The class also contains convenience member functions for easy track propagation, as depicted in Fig. 18. These methods are intended to facilitate the implementation of new algorithms.

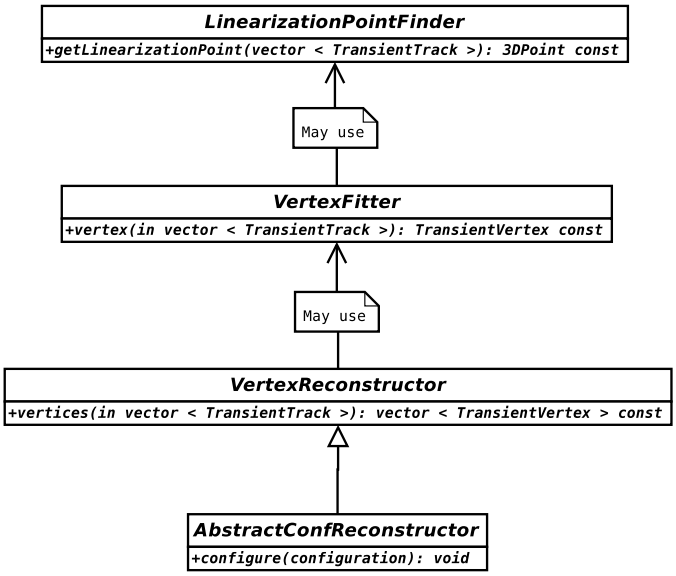

Fig. 16. The abstract interfaces used within CMS for vertex reconstruction.

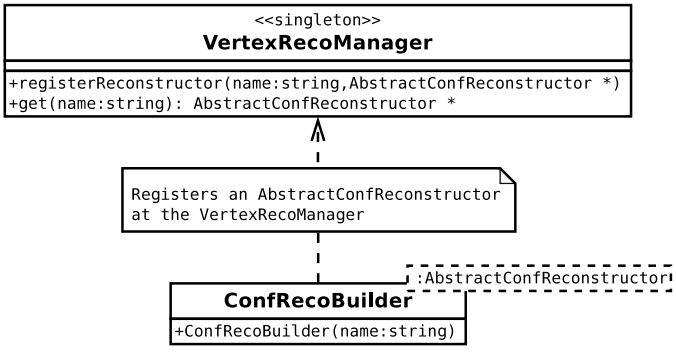

Fig. 17. The vertex reconstructors register at the `VertexRecoManager` singleton class.

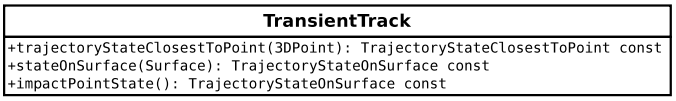

Fig. 18. Some convenience methods of the `TransientTrack`.

```
import vertigo
eventsource = vertigo.EventFactory
              ( "lcio:myfile.slcio" )
vertexfactory = vertigo.RaveVertexFactory()

for event in eventsource:
  vertices = vertexfactory.create
             ( event.tracks() )
  print vertices
```

Fig. 19. Simple python code snippet that reads tracks from an "lcio" file (ILC's official data format), reconstructing vertices.

VII. Validation

A separate small framework has been written that embeds RAVE for developing, debugging, and testing purposes. The framework is called "VERTIGO" and is written in C++ and python. Simple regression tests have been formulated in python. RAVE has to pass all these tests before a new release is issued.

Fig. 19 shows a simple VERTIGO event loop in python. VERTIGO serves as an excellent environment for algorithmic development.

Regression tests

VERTIGO has an extensive list of regression tests that are intended to cross-check on as many technical details as possible. Technically, all of RAVE's functionality has to be accessible from python. Many of RAVE's "subunits" are tested individually, such as the parameter parsing class, or the annealing schedulers. Numerically, residuals, standardized residuals, track weights, χ^2, and some more variables are checked, for all algorithms, against a fixed set of events. Since no randomness appears in these tests, some regression tests perform even an event-by-event comparison, up to a certain precision.

VIII. Availability

RAVE has been successfully tested on Intel and PPC CPUs, on several Linux platforms (Debian, Scientific Linux 4 and others), on Mac OS X, and on Windows, both compiled with cygwin and with MS Visual Studio.

The RAVE toolkit has been used from C++, Java, and Python. Interface to the latter two languages are provided via the SWIG [55] interface generator. RAVE interfaces ("glue code") exist for ILD's Marlin [56], org.lcsim [57], and for the "Roobasf" framework [58].

RAVE is now hosted at HEPFORGE [41]. It can be downloaded from there.

IX. Summary and Conclusions

A detector-indepedant toolkit has been presented which compiles and runs on many platforms and includes several state-of-the-art vertex and kinematic reconstruction algorithms. It was shown that the toolkit can be embedded in several different software environments, producing technically sound results. Interfacing RAVE with different programming languages has been shown to be simple. With RAVE new experiments can tab all of the CMS vertex reconstruction expertise gained by the CMS experiment with very little effort. Even if not used as an experiment's "main" vertex reconstruction solution, the experiment can still gain from having baseline algorithms at their hands. No direct comparison of RAVE with other vertex reconstruction solutions has yet been made; this is an important task for the future.

Acknowledgment

Thanks are due to the "CMS vertexing circle" for code shared with RAVE. Special thanks go to Meinhard Regler and to Rudi Frühwirth for formulating a large part of the algorithms used. Thanks to Rudi Frühwirth and Winni Mitaroff for comments on the manuscript and valuable discussions concerning this publication.

References

[1] K. Abe *et al.*, "Observation of mixing induced CP violation in the neutral B meson system," *Phys. Rev.*, vol. D66, p. 032007, 2002.
[2] C. Weiser, "A Combined Secondary Vertex Based B-Tagging Algorithm in CMS," *CMS Note*, vol. 014, 2006.
[3] A. Wildauer and P. F. Åkesson, "Vertex finding and b-tagging algorithms for the atlas inner detector." presented at Computing in High Energy Physics and Nuclear Physics, Interlaken, Switzerland, 2004.
[4] R. Adolphi *et al.*, "The CMS experiment at the CERN LHC," *J. Instrum.*, vol. 3, p. S08004, 2008.
[5] T. Abe *et al.*, "The International Large Detector: Letter of Intent," 2010, FERMILAB-LOI-2010-01.
[6] H. Aihara, (Ed.) *et al.*, "SiD Letter of Intent," 2009, SLAC-R-944.
[7] S. K. Park *et al.*, "The 4th Concept Detector for the International Linear Collider," 2007, presented at Lepton-Photon Conference, Daegu, Korea.
[8] S. Hashimoto *et al.*, "Letter of intent for KEK Super B Factory," 2004, KEK-REPORT-2004-4.
[9] I. Adachi *et al.*, "sBelle Design Study Report," 2008, KEK-REPORT-2008-7.
[10] "Tracking and Vertexing Results from First Collisions," *CMS Physics Analysis Summary*, vol. CMS-PAS-TRK-10-001, 2010.
[11] G. Patrick and B. Schorr, "Vertex fitting of several helices in space," *Nucl. Instrum. Meth. A*, vol. 241, p. 132, 1985.
[12] P. Billoir, R. Frühwirth, and M. Regler, "Track element merging strategy and vertex fitting in complex modular detectors," *Nucl. Instrum. Meth. A*, vol. 241, p. 115, 1985.
[13] R. Frühwirth, "Application of Kalman filtering to track and vertex fitting," *Nucl. Instrum. Meth. A*, vol. 262, p. 444, 1987.
[14] R. Frühwirth and T. Speer, "A Gaussian-Sum Filter for Vertex Reconstruction," *Comp. Phys. Comm.*, vol. 174, p. 935, 2006.
[15] R. Frühwirth *et al.*, "Vertex reconstruction and track bundling at the LEP collider using robust algorithms," *Comp. Phys. Comm.*, vol. 96, pp. 189–208, 1996.
[16] E. Chabanat *et al.*, "Vertex reconstruction in CMS," *Nucl. Instrum. Meth. A*, vol. 549, p. 188, 2005.
[17] K. Van Driessen and P. Rousseeuw, "Computing LTS regression for large data sets," *Data Mining and Knowledge Discovery*, vol. 12/1, pp. 29–45, 1999.
[18] R. Frühwirth *et al.*, "Adaptive Vertex Fitting," *J. Phys. G: Nucl. Part. Phys.*, vol. 34, p. 343, 2007.
[19] R. Frühwirth and W. Waltenberger, "Redescending M-estimators and Deterministic Annealing, with Applications to Robust Regression and Tail Index Estimation," *Austrian Journal of Statistics*, vol. 37, p. 301, 2008.
[20] A. G. Frodesen, O. Skjeggestad, and H. Tøfte, *Probability and Statistics in Particle Physics*. Oxford Univ Press, 1979.
[21] K. Prokofiev, "Study of the $B_s^0 \to (J/\psi)\,\phi \to \mu^+\mu^- K^+ K^-$ Decay with the CMS Detector at LHC," Ph.D. dissertation, Zürich, 2005, CERN-THESIS-2009-109.
[22] G. Piacquadio and C. Weiser, "A new inclusive secondary vertex algorithm for b-jet tagging in ATLAS," *J. Phys. Conf. Ser.*, vol. 119, p. 032032, 2008.
[23] S. Hillert, "LCFI Vertex Package." presented at VERTEX2008, pos VERTEX2008:023,2008.
[24] A. Strandlie and R. Frühwirth, "Track and vertex reconstruction: From classical to adaptive methods," *Rev. Mod. Phys.*, vol. 82, pp. 1419–1458, 2010.
[25] T. Sjöstrand, S. Mrenna, and P. Skands, "A Brief Introduction to PYTHIA 8.1," *Comp. Phys. Comm.*, vol. 178, p. 852, 2008.
[26] T. Stelzer and W. F. Long, "Automatic generation of tree level helicity amplitudes," *Comp. Phys. Comm.*, vol. 81, pp. 357–371, 1994.
[27] S. Agostinelli *et al.*, "GEANT4: A simulation toolkit," *Nucl. Instrum. Meth. A*, vol. 506, pp. 250–303, 2003.
[28] "ROOT - A C++ framework for petabyte data storage, statistical analysis and visualization," *Comp. Phys. Comm.*, vol. 180, p. 2499, 2009.
[29] L. Lönnblad, "CLHEP – a project for designing a C++ class library for high energy physics," *Comp. Phys. Comm.*, vol. 84, p. 307, 1994.
[30] B. C. Allanach, "SOFTSUSY: a program for calculating supersymmetric spectra," *Comp. Phys. Comm.*, vol. 143, p. 305, 2002.
[31] Sourceforge – the world's largest open source software development web site. [Online]. Available: http://sourceforge.net
[32] A. Buckley *et al.*, "HepForge: A lightweight development environment for HEP software," 2006, arXiv:hep-ph/0605046.
[33] G. P. Salam and G. Soyez, "A practical Seedless Infrared-Safe Cone jet algorithm," *JHEP*, vol. 05, p. 086, 2007.
[34] A. Cervera Villanueva, J. A. Hernando, and J. J. Gomez-Cadenas, "RecPack – a general reconstruction toolkit," *Nucl. Instrum. Meth. A*, vol. 534, p. 180, 2004.
[35] C. Höppner, S. Neubert, B. Ketzer, and S. Paul, "A Novel Generic Framework for Track Fitting in Complex Detector Systems," *Nucl. Instrum. Meth. A*, vol. 620, pp. 518–525, 2010.
[36] S. Gorbunov and I. Kisel, "Reconstruction of decayed particles based on the Kalman filter," Darmstadt, Tech. Rep. CBM-SOFT-note-2007-003, 2007.
[37] J. P. Palacios, "VELO vertexing and tracking algorithms of the LHCb trigger system," *Nucl. Instrum. Meth. A*, vol. 560, p. 84, 2006.
[38] E. Gamma *et al.*, *Design Patterns*. Addison-Wesley, 1995.
[39] P. Billoir and S. Qian, "Fast vertex fitting with a local parametrization of tracks," *Nucl. Instrum. Meth. A*, vol. 311, pp. 139–150, 1992.
[40] ——, "Erratum to 'fast vertex fitting with a local parametrization of tracks'," *Nucl. Instrum. Meth. A*, vol. 350, p. 624, 1994.
[41] RAVE at HepForge. [Online]. Available: http://projects.hepforge.org/rave
[42] V. Karimäki, "Effective vertex fitting," *CMS Note*, vol. 1997/051.
[43] W. Waltenberger, "Development of Vertex Finding and Vertex Fitting Algorithms for CMS," Ph.D. dissertation, Vienna University of Technology, 2004, CMS-TS-2006-012.
[44] W. Waltenberger, "Adaptive Vertex Reconstruction," *CMS Note*, vol. 2008/033.
[45] W. Adam *et al.*, "Reconstruction of electrons with the Gaussian-sum filter in the CMS tracker at the LHC," *J. Phys. G: Nucl. Part. Phys.*, vol. 31, p. N9, 2005.
[46] D. Jackson, "A topological vertex reconstruction algorithm for hadronic jets," *Nucl. Instrum. Meth. A*, vol. 388, p. 247, 1997.
[47] "Precision electroweak measurements on the Z resonance," *Phys. Rept.*, vol. 427, pp. 257–454, 2006.
[48] F. Abe *et al.*, "Observation of top quark production in $\bar{p}p$ collisions," *Phys. Rev. Lett.*, vol. 74, pp. 2626–2631, 1995.
[49] S. Lowette, "B-Tagging as a Tool for Charged Higgs Boson Identification in CMS," Ph.D. dissertation, Vrije Universiteit Brussel, 2006.
[50] "Prospects for the first Measurement of the $t\bar{t}$ Cross Section in the Muon-plus-Jets Channel at $\sqrt{s} = 10\,\mathrm{TeV}$ with the CMS Detector," *CMS Physics Analysis Summary*, vol. CMS-PAS-TOP-09-003, 2009.
[51] "Algorithms for b Jet Identification in CMS," *CMS Physics Analysis Summary*, vol. CMS-PAS-BTV-09-001, 2009.
[52] "Impact of Tracker Misalignment on the CMS b-Tagging Performance," *CMS Physics Analysis Summary*, vol. CMS-PAS-BTV-07-003, 2007.
[53] F. Moser, "Implementation of a General-Purpose Kinematic Estimator in the Software Environment of the International Linear Collider and Application to Pair Production of W Bosons," Master's thesis, Vienna University of Technology, 2008.

[54] F. Moser, W. Waltenberger, M. Regler, and W. Mitaroff, “Implementation and application of kinematic vertex fitting in the software environment of ILD.” presented at the 11th Int. Linear Collider Workshop (LCWS’08), UIC, Chicago, 2008.
[55] D. Beazley, “SWIG: An Easy to Use Tool for Integrating Scripting Languages with C and C++.” presented at the Fourth Annual USENIX Tcl/Tk Workshop, Monterey, USA, 1996.
[56] RAVE “Glue” Code for Marlin. [Online]. Available: http://stop.itp.tuwien.ac.at/websvn/listing.php?repname=marlinrave
[57] RAVE “Glue” Code for org.lcsim. [Online]. Available: http://stop.itp.tuwien.ac.at/websvn/listing.php?repname=lcsimrave
[58] Belle-2 Software Framework. [Online]. Available: http://wiki.kek.jp/display/sbelle/SuperBelle+Wiki+Home