

A detector-independent library for reconstruction of interaction vertices (RAVE)

Wolfgang Waltenberger,
Fabian Moser

Outline

- What do we want to do?
- How do we want to do it?
- Caveat - no official approval of this project
- Why do we want to do it?
- What have we done already?
- When do we want to do what?
- What novel algorithms will the toolkit introduce?
- References

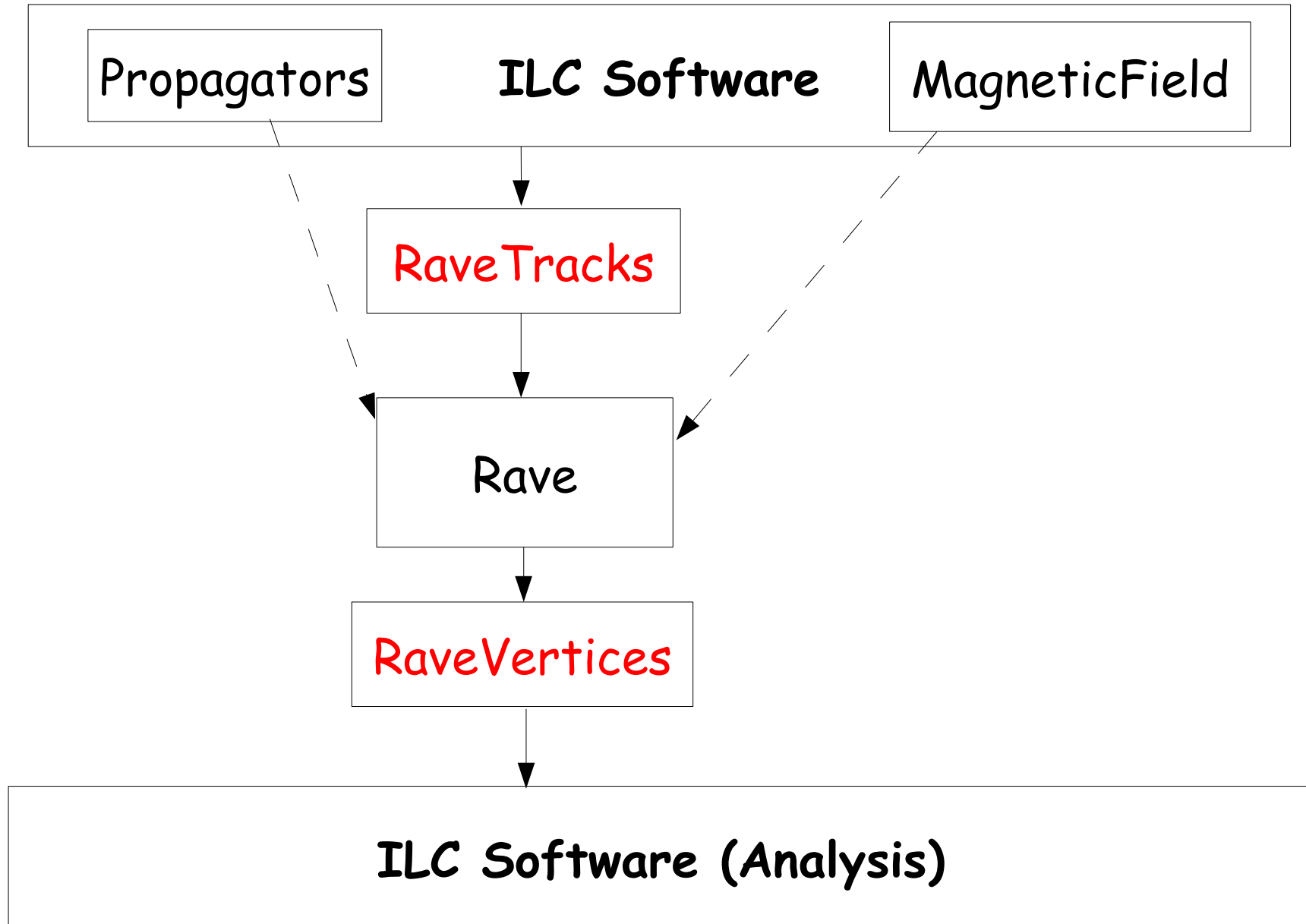
What do we want to do?

The project's primary goal is to produce **an experiment-independent library that finds and fits interaction vertices**, given a set of reconstructed tracks, access to the magnetic field, and an abstract mechanism that can be used to propagate tracks through the detector (i.e. a propagator).

The working title is: RAVE

Reconstruction in an Abstract Vertexing Environment

What do we want to do? (2)

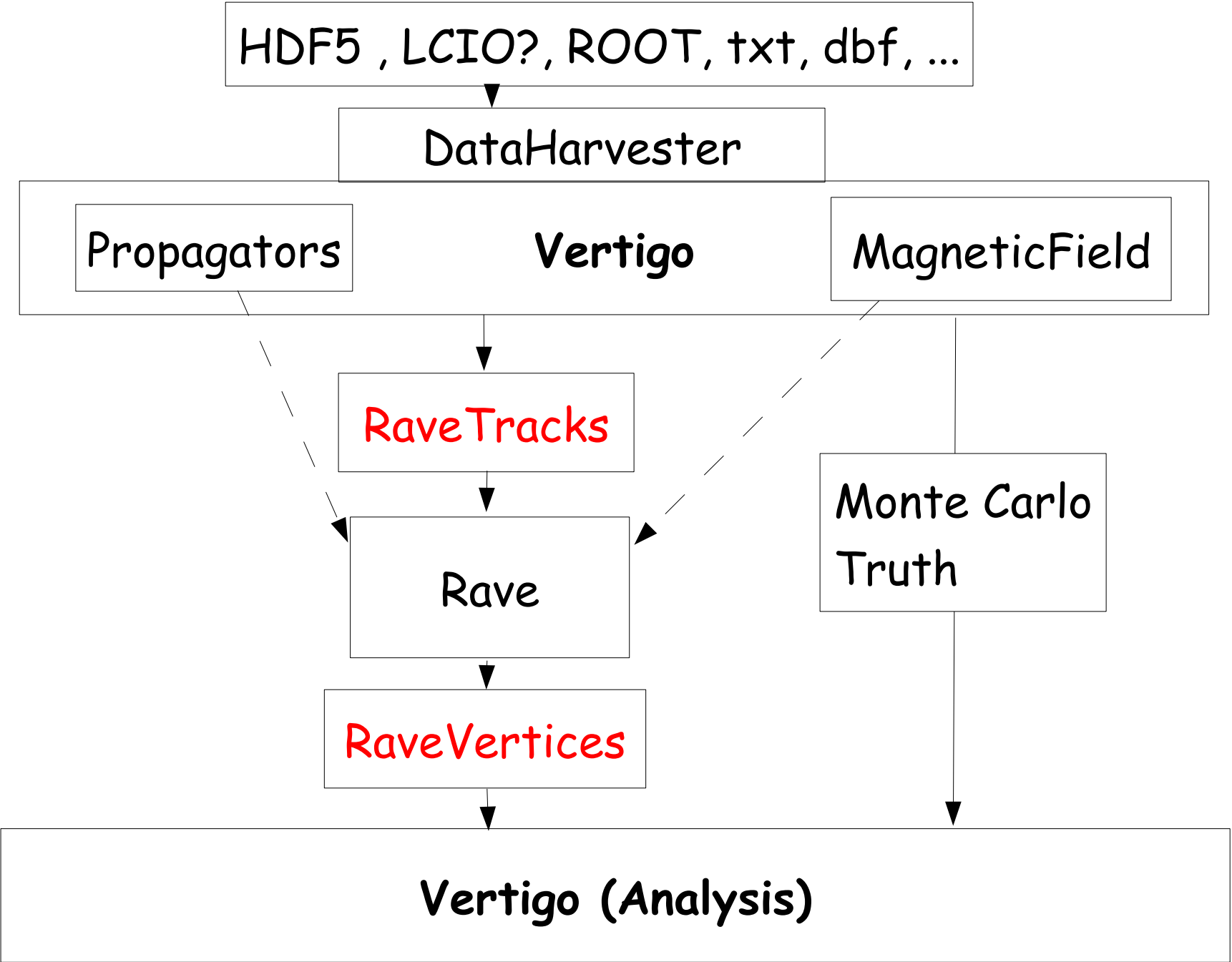


What do we want to do? (3)

A vertex reconstruction library ("rave") is (for us) only the first step.

We want to be able to also have **a complete standalone toolkit** ("vertigo") **that is packaged separately from the library**. It will serve as a simple framework that can be used to test algorithms and/or analysis code. The concept of skins will make it easy to simulate different experiment-specific setups.

Our DataHarvester will be employed to be able to read reconstructed tracks plus optional MonteCarlo information (tracks and vertices) that have been produced by the different experiments. An LCIO backend for the DataHarvester could make ILC data accessible for Vertigo. **HDF5** is currently our favorite file format.



How we want to do it

We want to 'rip' vertexing code out of CMS. This has several advantages:

- CMS Vertexing is a high-quality code base. The concepts of abstraction and encapsulation have already been pushed quite far.
 - The code base is algorithmically extremely strong and innovative. CMS vertexing is the "home" of the adaptive vertex fitter, the multi vertex fitter, and the gaussian sum vertex fitter. Three novel concepts which are in the process of being implemented in many other experiments.

But ...

Caveat

So far, we have only focussed on the (technical) feasibility.

The "political" implications are not yet resolved.

The code that needs to be 'ripped', comes from many authors.

The CMS community, the people involved, have not yet been asked collectively about their consensus.

There is

no official CMS approval for this project.

Although, HEPHY Vienna fully supports this idea. Also, a few individuals have already given their consensus.

Why do we want to do it

The potential benefits for ILC are obvious.

But, there are also several reasons why **we** (CMS vertexing people) want to create a vertexing toolkit:

- **Faster development cycles** - Algorithms can be implemented and studied in a much faster way than within a full integration in CMSSW.
- **More users = better code** - debugging efforts are contributed backwards to CMS
- **Potential backcontributions** - we might see algorithms developed for ILC, used in CMS
- **Portable analysis code** - a vertexing toolkit automatically introduces a few important base classes for physics analysis. We want to be able to "port" our analyses from one experiment to the next.

What we have already done

Many parts of CMS vertexing already compile and run in a standalone fashion. Development of a "rave" library is progressing well. Interfaces are getting defined. First complete vertex reconstruction is a matter of weeks.

What is “rave”?

“rave” = Reconstruction in an Abstract Vertexing Environment.

“rave” =

- build files (scons files now!)

- + most of CMS SW vertexing (100 % code compatible)

- + a few parts of CMS SW tracking / framework
(100 % code compatible)

- + a few “hacked” CMS SW classes

- + a few rave-specific classes, mostly “facades” and “adaptors”.

How will “rave” be used?

Easy to
interface to
Java and
python!

The interface is not yet fully defined.

However, usage should look something like this in C++:

```
// user produces RaveTracks
```

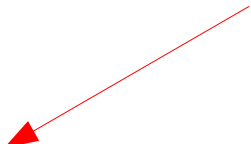
```
std::vector < RaveTrack > trks;
```

```
// user wants reconstruction to happen with MyPropagator,
```

```
MyMagneticField, and the “default” method.
```

```
RaveFactory factory ( MyMagneticField(),  
                      MyPropagator(), "default" );
```

description
of reconstruction
strategy- as a string.



```
std::vector < RaveVertex > vertices = factory.create ( trks );
```

Which algorithms will the toolkit contain?

The toolkit will contain vertex fitters and vertex finders and algorithms that cannot be categorised in the aforementioned way.

Fitters:

- Least-squares kalman vertex fitter
- trimmed kalman vertex fitter, discarding outliers
- adaptive (iterative, weighted) vertex fitter (powerful novel algorithm)
- adaptive multi vertex fitter, fitting several vertices at once (powerful novel algorithm)
- gaussian sum vertex fitter, dealing with non-gaussian error model (powerful novel algorithm)

Which algorithms will the toolkit contain? (2)

On the finders side we will see e.g.:

- trimmed kalman vertex finder
- adaptive vertex finder
- d0-phi vertex finder, clustering tracks in the d0-phi plane
- deterministic annealing algorithm

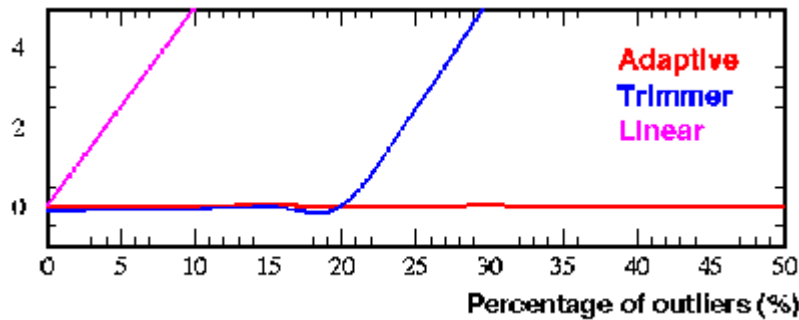
...

ZVTOP is, of course, a very interesting algorithm for the rave library.

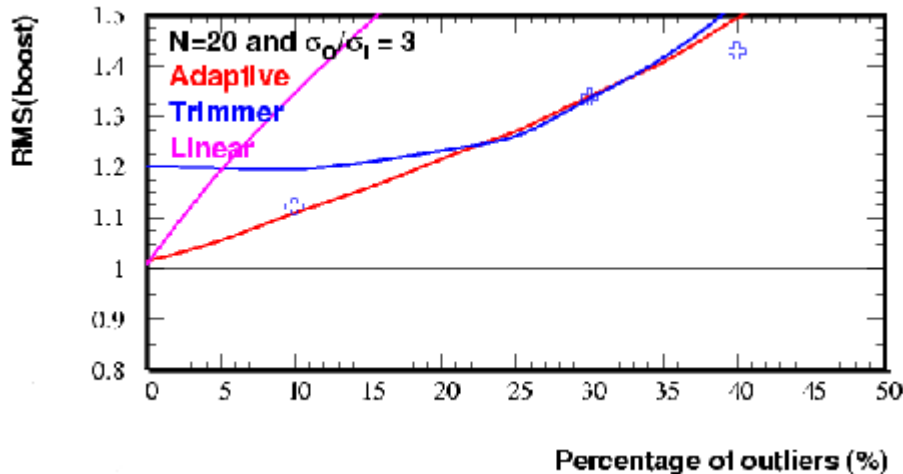
Which algorithms will the toolkit contain(3)

Demonstration of the assets of the adaptive vertex fitter.
On highly "artificial" data only.

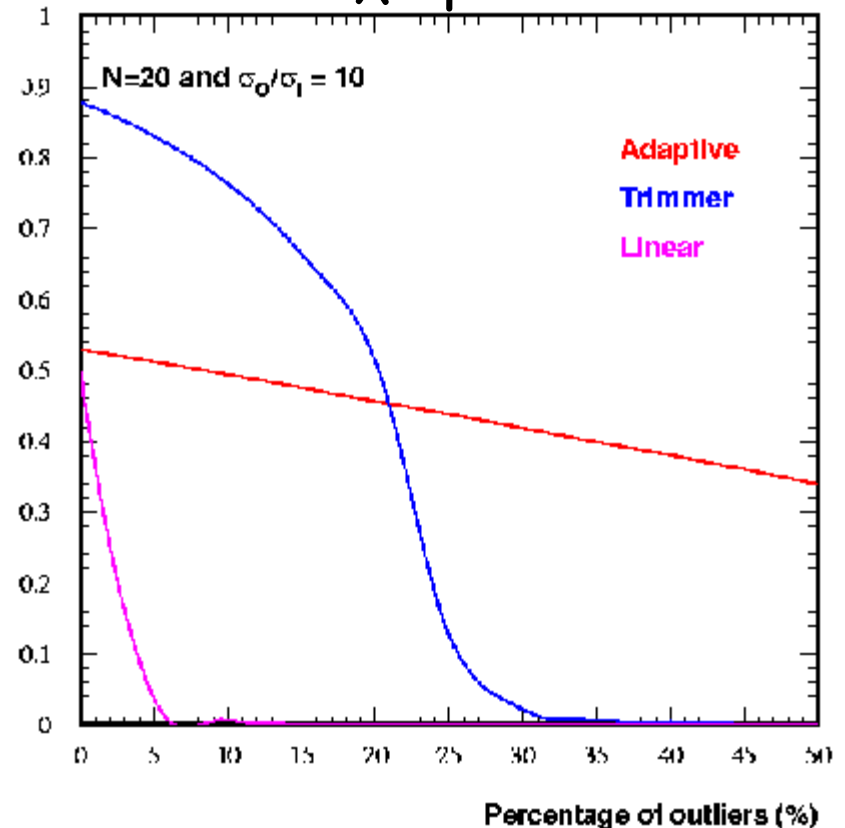
bias



standardized residuals



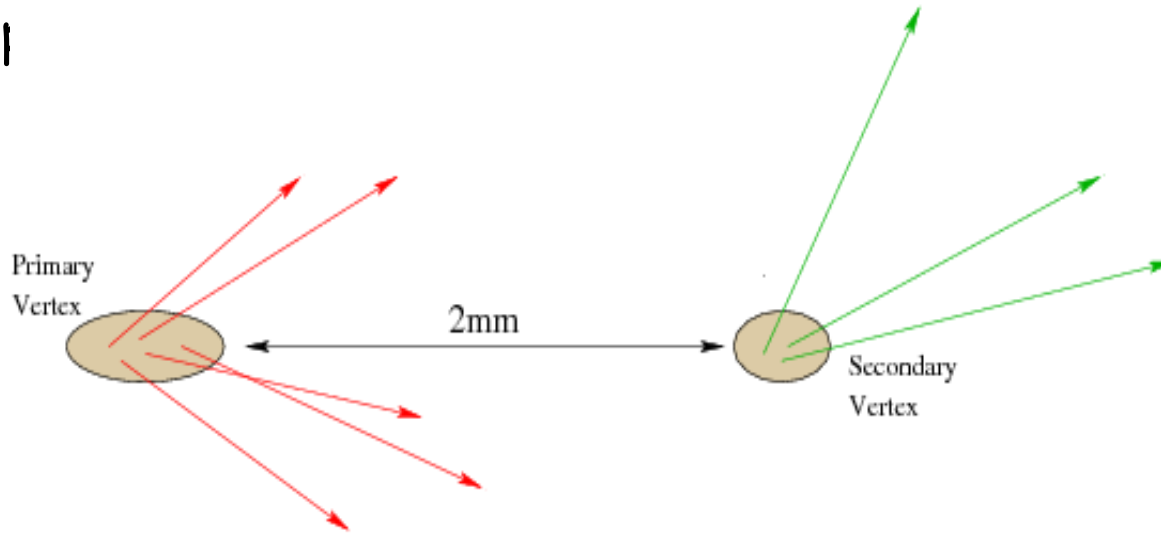
χ^2 -prob



•"Sensitivity of Robust Vertex Fitting Algorithms" J. D'Hondt et al., IEEE Trans. Nucl. Science 51, 2037 (2004) CMS NOTE-2004/002

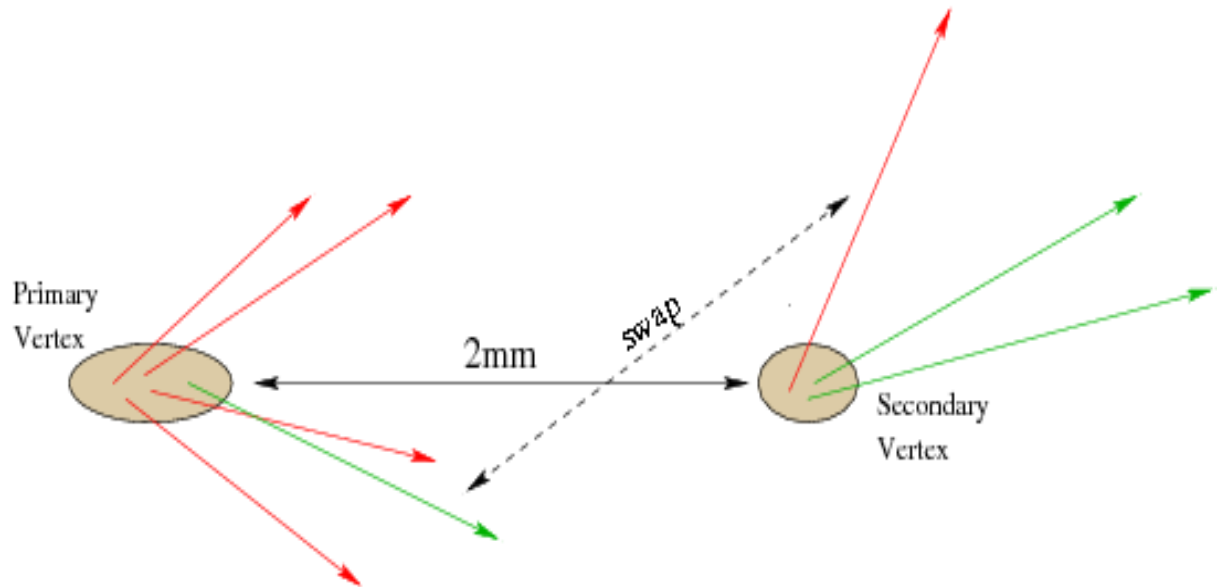
Which algorithms will the toolkit contain(4)

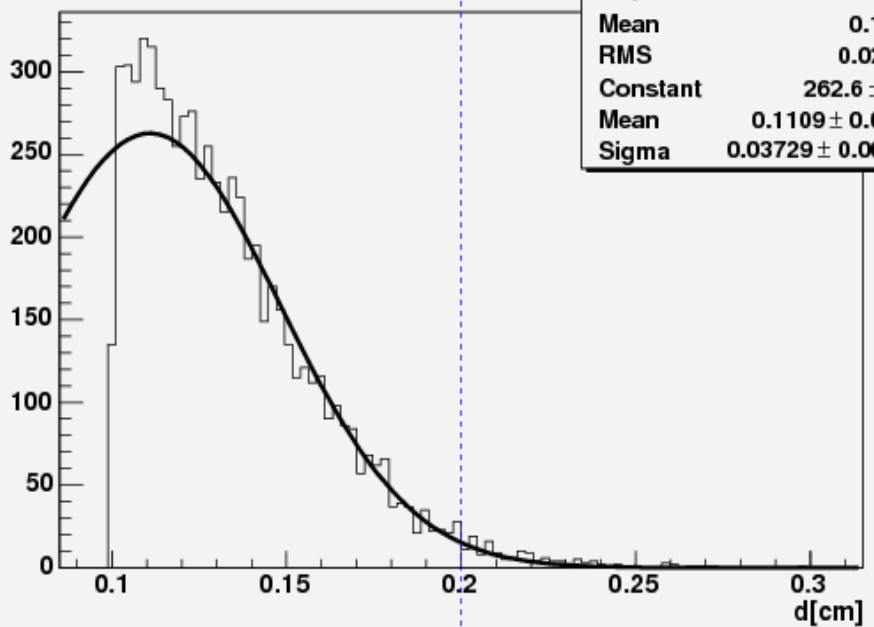
Demonstration of how the multi vertex fitter "corrects" errors of the way.



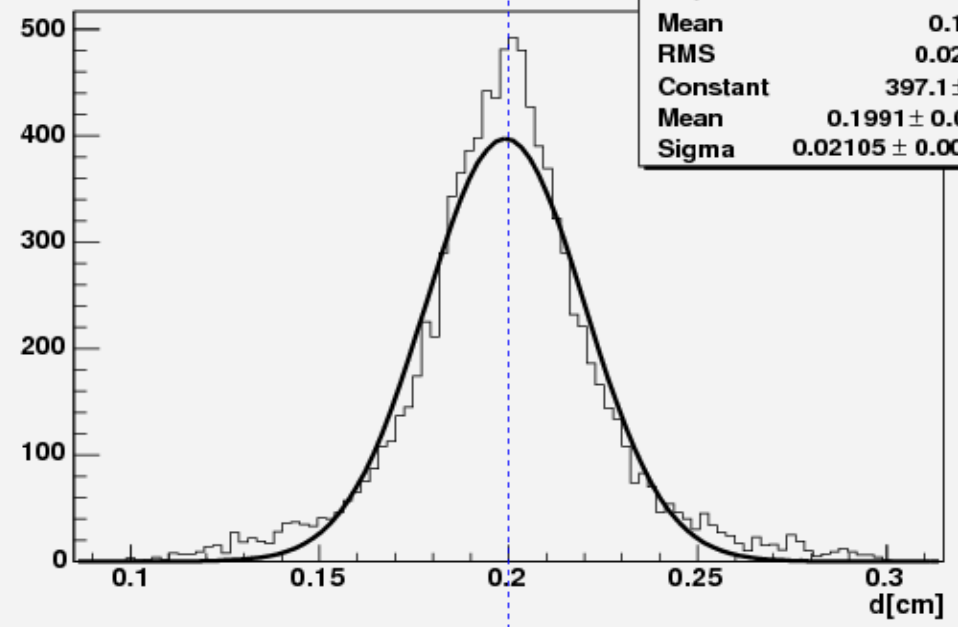
Example for multivertexfitter

track swapping

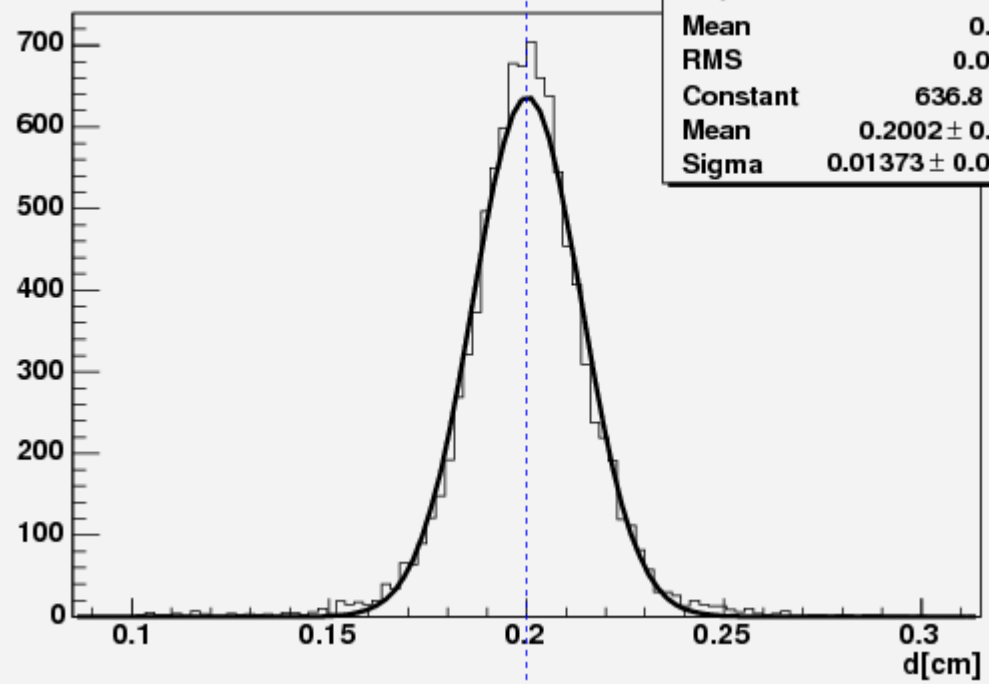


Flightpath resolution, KVF

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

Flightpath resolution, AVF

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

Flightpath resolution, MVF

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

Options for the future

Of course, writing an e.g. MarlinReco processor for RAVE must be one of its primary goals.

Kinematic finding / fitting code could easily be ported, as well.

Interfacing with **PAX** (Physics Analysis Experts) might be an interesting goal.

Also, interfacing with **RecPack** (track reconstruction toolkit) is an option.

References

- "Modern vertex reconstruction methods", R. Fruehwirth and W. Waltenberger and T. Speer. HEPHY-PUB-818/06. November 2005
- "Adaptive Multi-Vertex Fitting", CMS CR-2004/062
- "VERTIGO", W. Mitaroff, W. Waltenberger, 7th Int. Conf. on Linear Colliders, Paris, April 2004
- "Development of Vertex Finding and Vertex Fitting Algorithms for CMS", PhD thesis, W. Waltenberger, Nov 2004
- "Sensitivity of Robust Vertex Fitting Algorithms" J. D'Hondt et al., IEEE Trans. Nucl. Science 51, 2037 (2004) CMS NOTE-2004/002
- "Robust Vertex Fitters", T. Speer and R. Fruehwirth and P. Vanlaer and W. Waltenberger. Proceedings for the Time Workshop, Zurich, October 2005.

When will we do what

Roadmap (modulo politics):

Summer 2006 - A first version of a vertexing library ("rave")

Fall 2006 - A first version of a vertexing toolkit ("vertigo").

- First attempts to interface with other experiments?
- Interfacing with ILC?