Vertex Reconstruction in CMS

Rudi Frühwirth, Kirill Prokofiev, David Schmitt,
Gabriele Segneri, Thomas Speer,
Pascal Vanlaer, Wolfgang Waltenberger

CHEP 2003 @ La Jolla, CA, USA , March 2003

Vertex reconstruction in CMS

Outline

- Introduction
- Vertex fitting
  - Linear methods
  - Robustifications
- Inclusive vertex finding
  - Problem definition
  - Algorithms
  - Results

Introduction (recap)

Vertex reconstruction can be decomposed into:

- Vertex finding:

given a set of tracks, separate it into clusters of compatible tracks, i.e. vertex candidates

  - inclusively: not related to a particular decay channel
    search for secondary vertices in a jet
  - exclusively: find best match with a decay channel.
    general solution: combinatorial search → not discussed here.

- Vertex fitting
  - find the 3D point most compatible with a vertex candidate (i.e. a set of tracks).
  - track smoothing: additional vertex information is used to re-estimate track momenta

Vertex Fitting

Vertex Fitting

The Task of estimating a point in 3d space that is most compatible with a given set of reconstructed tracks.

Least Squares Methods:

- LinearVertexFitter
- KalmanVertexFitter

sensitive to outliers and non-Gaussian tails in the track errors!

Robustified Methods:

- TrimmingVertexFitter
- AdaptiveVertexFitter
- LMSVertexFitter

Least square methods

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

LinearVertexFitter

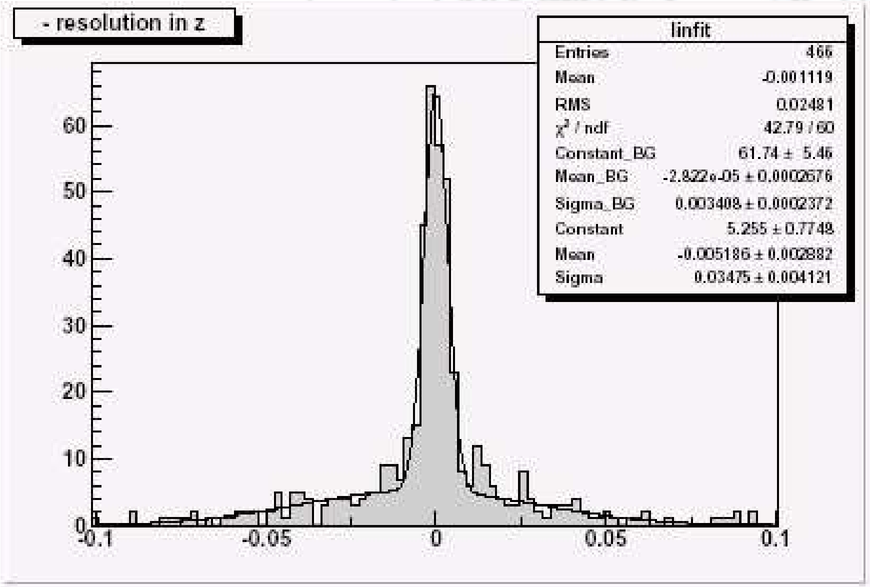
V.Karimäki, CMS Note 1997/051

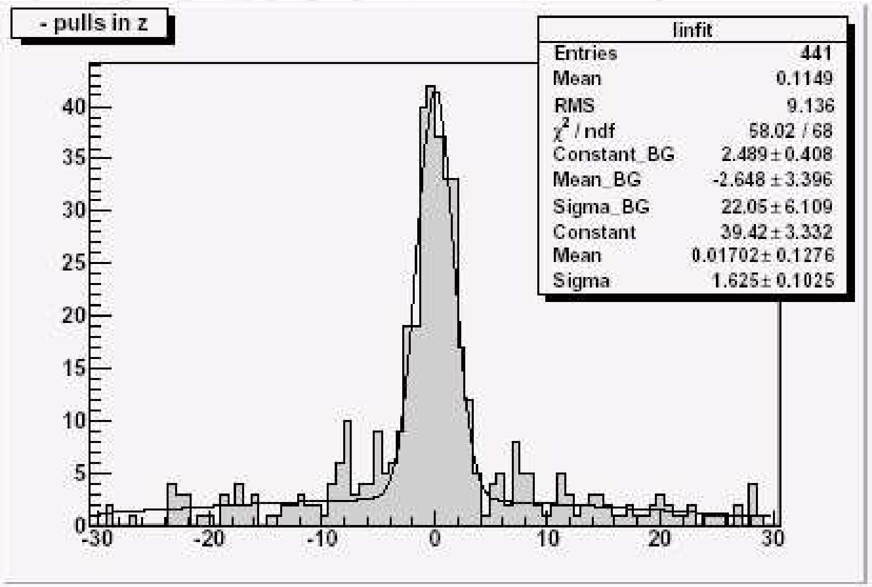
KalmanVertexFitter

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

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

Least squares fit, z-resolution and pull





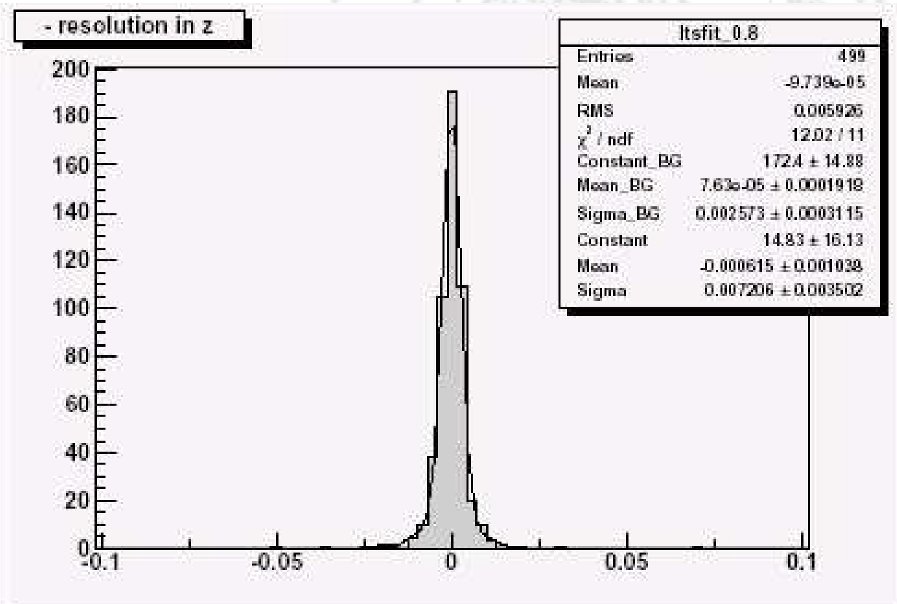
Trimming Vertex Fitter

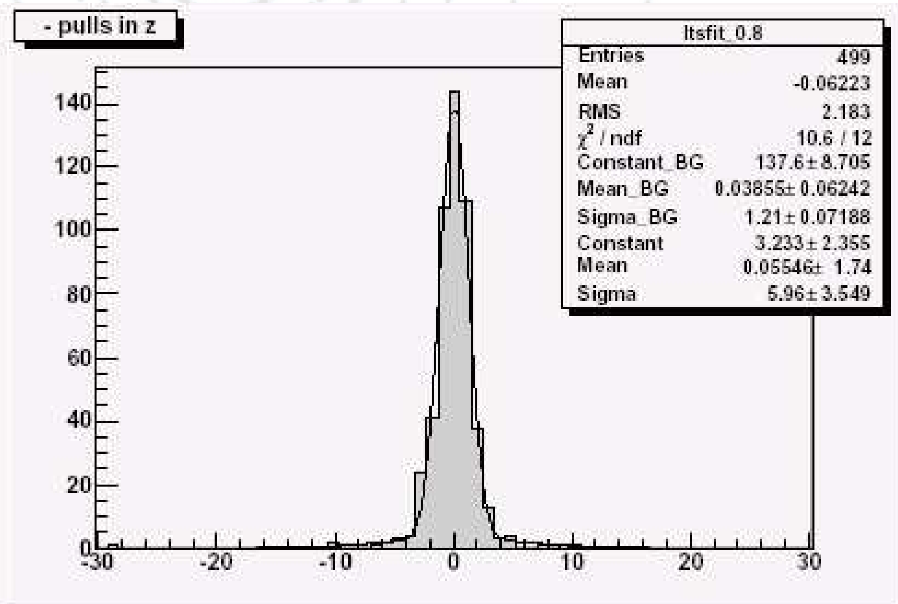
$$\hat{\beta}_{LTS} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{h<n} r_i^2(\beta)$$

LTSVertexFitter: fast Least Trimmed (sum of) Squares

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

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

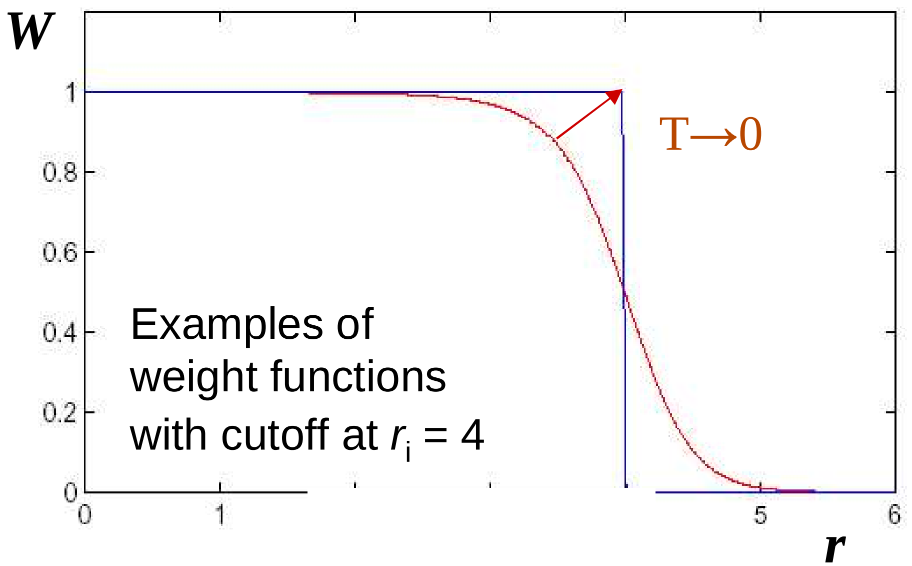


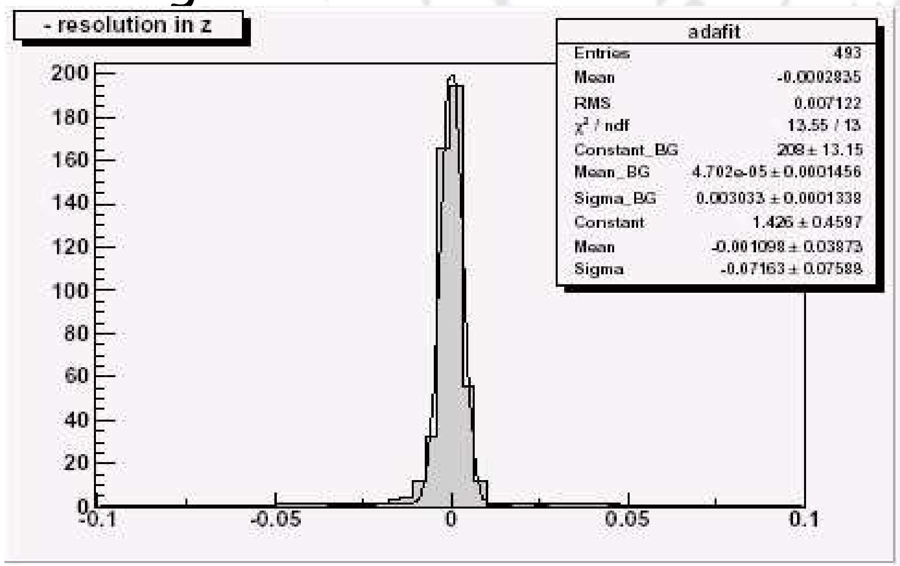


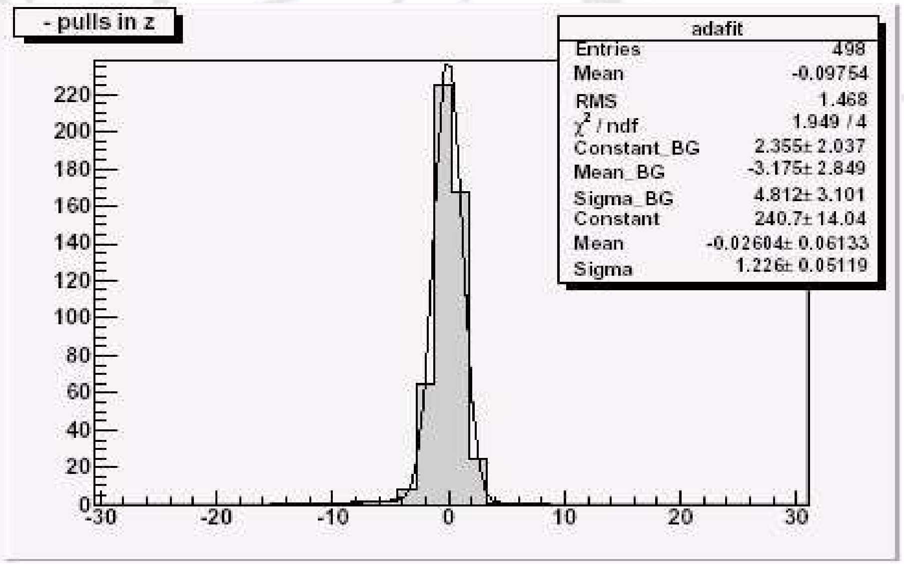
AdaptiveVertexFitter

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

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





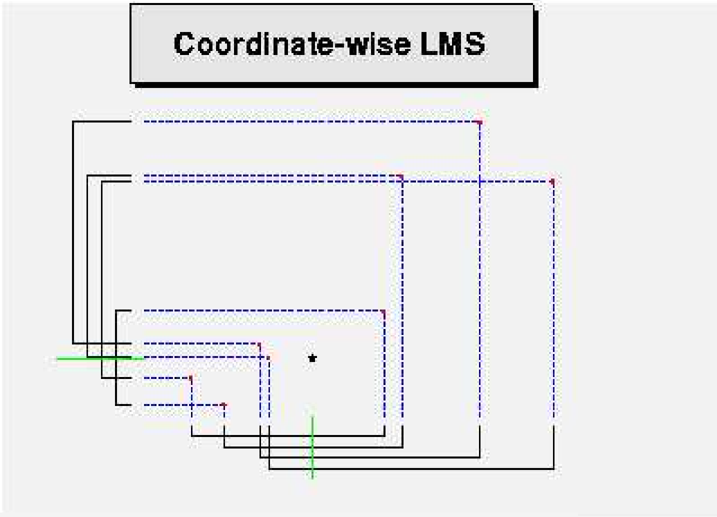


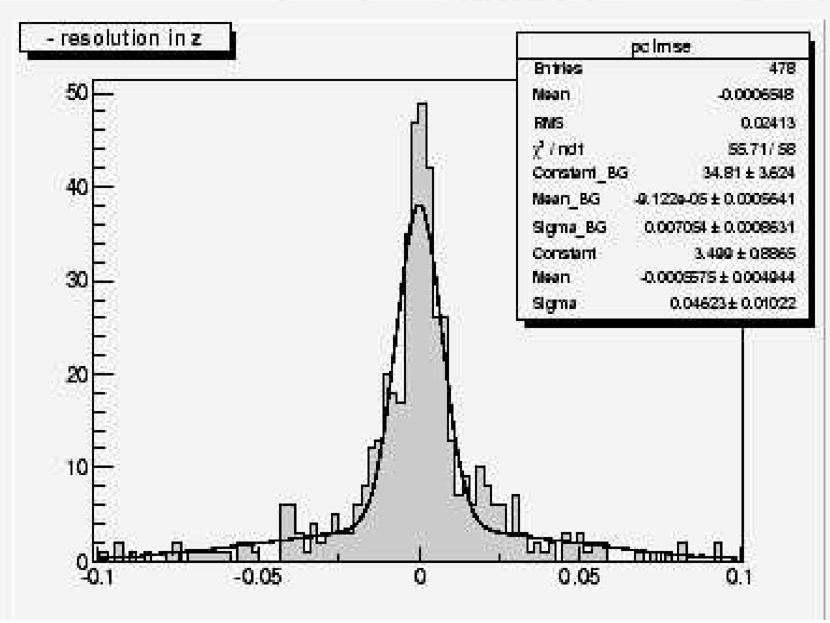
LMS Vertex Fitter

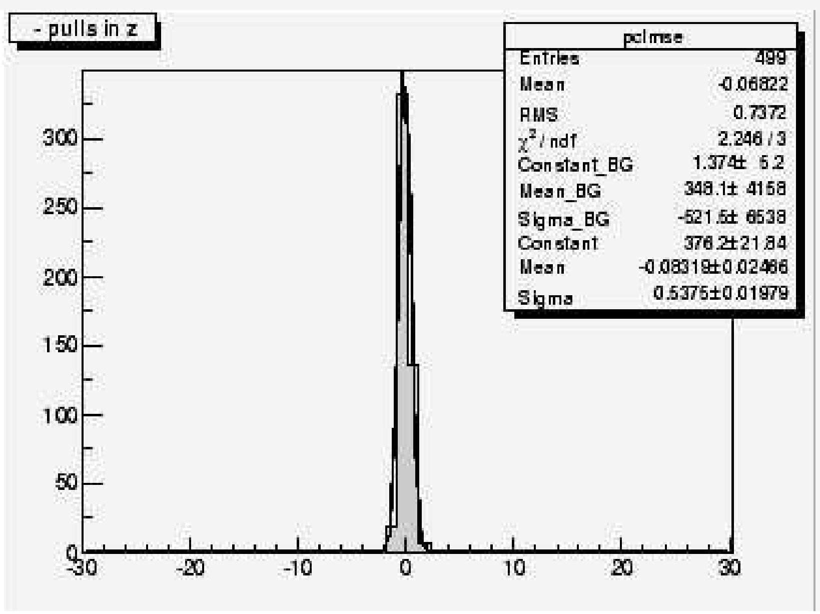
$$\hat{\beta}_{LMS} = \underset{\beta}{\operatorname{argmin}} \operatorname{med}\left(r_i^2(\beta) \right)$$

LMSVertexFitter: Least Median of Squares

- minimizes median of the squared distances – coordinate wise
- very robust (breakdown point = 0.5)
- very fast implementation
- inefficient; poor error estimate







Vertex Finding

Algorithms

Hierarchic vertex finding algorithms can be classified in:

- Agglomerative algorithms
  - at first iteration, each track constitutes a vertex candidate
  - merge compatible candidates
    - until stopping condition is met
- Divisive algorithms
  - initial vertex candidate made of the whole set of tracks
  - split into incompatible candidates
    - until stopping condition is met

There are also non-hierarchic methods (e.g. vector quantisation).

Divisive Clusterers - Principal Vertex Finder

Divisive algorithm, search for primary and secondary vertices

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

fit all tracks to a common vertex

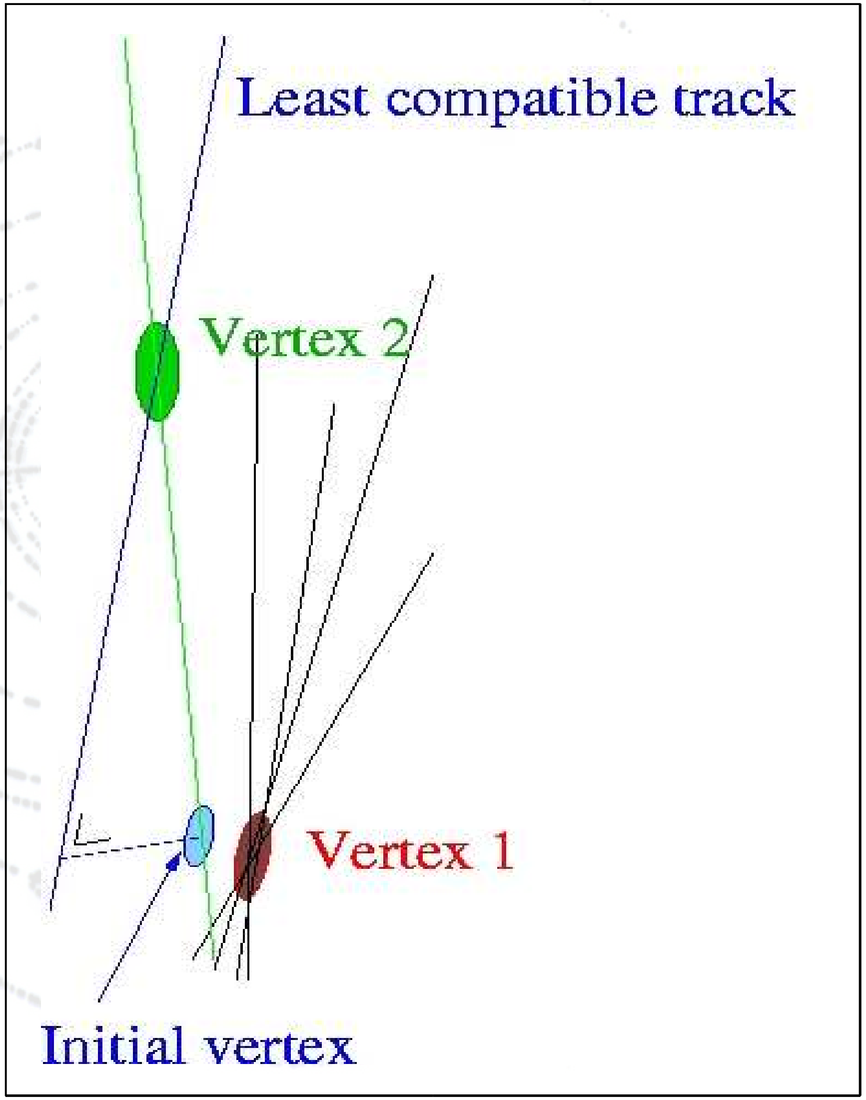
remove least compatible track

refit vertex.

1 vertex candidate

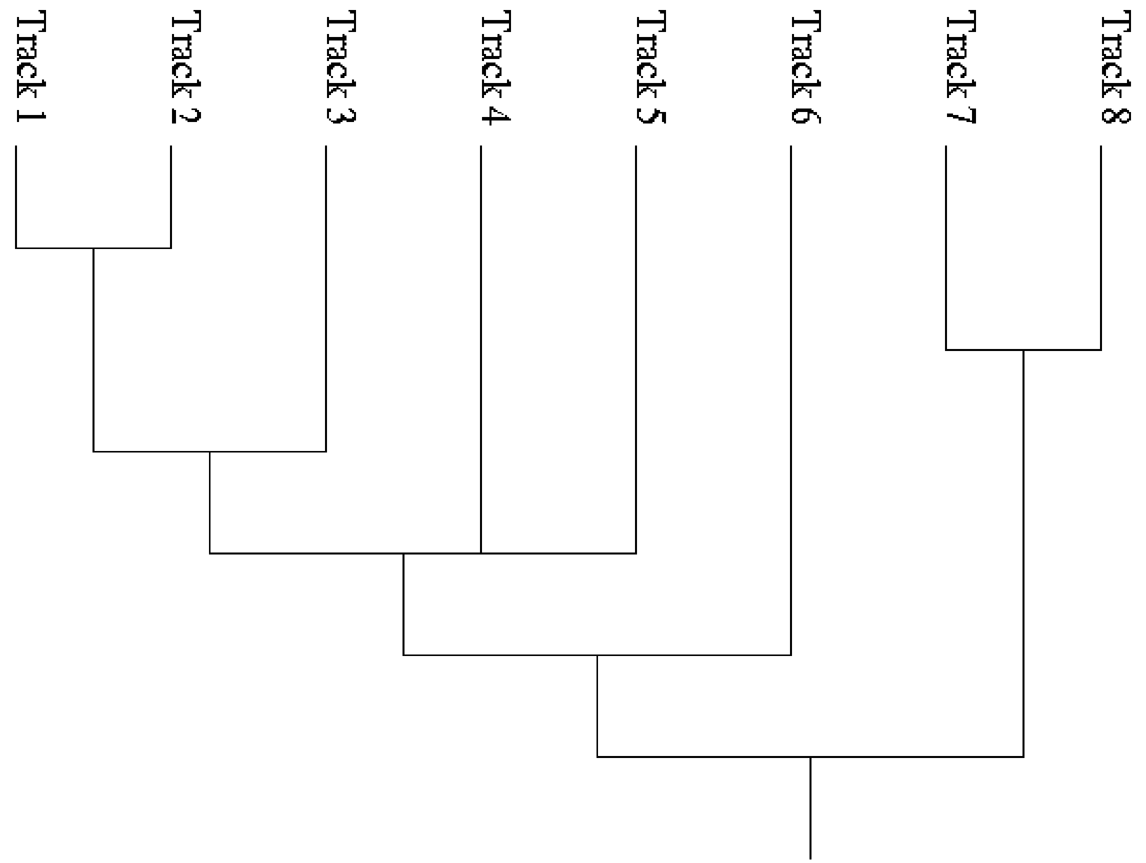
+ 1 set of discarded tracks

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



Agglomerative Finders

Agglomerative clustering algorithms start with singleton groups, and then proceed with iteratively merging pairs of groups with the minimal distance. The properties of the algorithm depend on the distance metric.

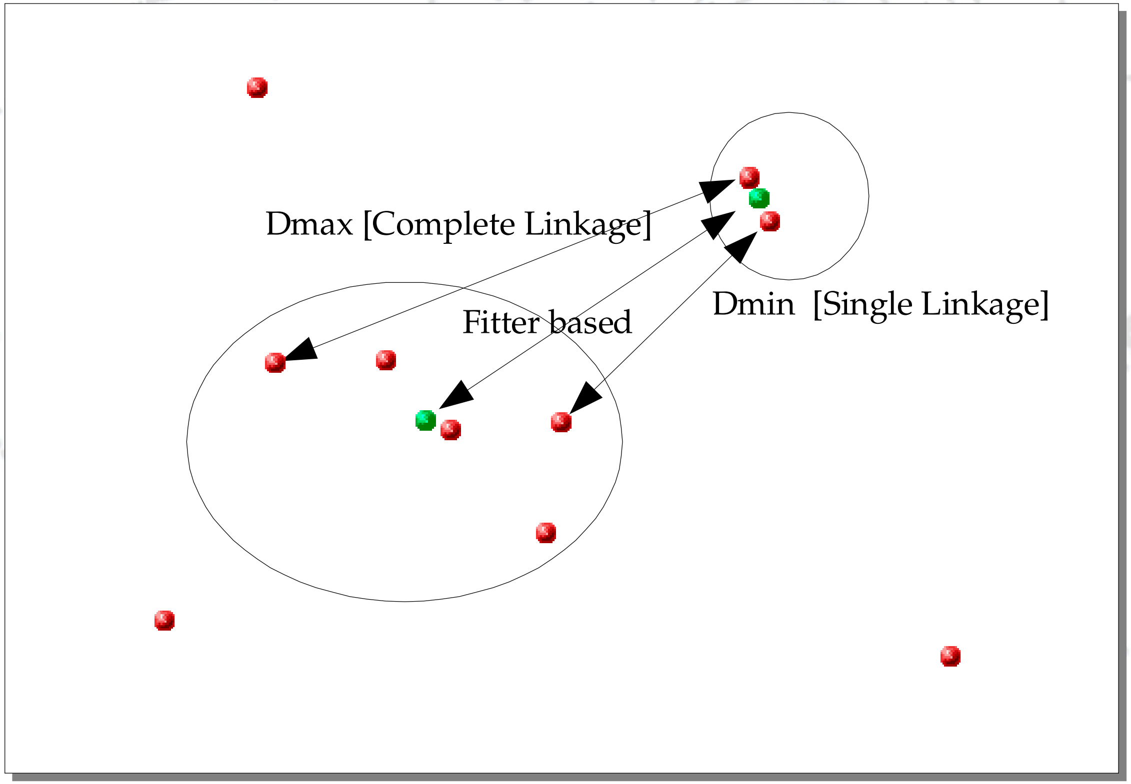


Agglomerative Finders (2)

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

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

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



Agglomerative Finders (3)

Theorem: The triangle inequality does not hold for the distance matrix between the PCA's of n tracks.

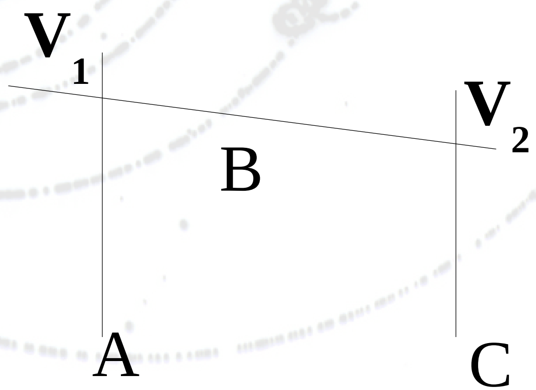
Proof: Let A, B, and C denote three tracks. Let A and B share one common vertex V_1; let further B and C also share one common vertex V_2. Then:

$$\overline{AB} = \varepsilon, \ \overline{BC} = \varepsilon, \ \overline{AC} = d \gg \varepsilon$$

Hence:

$$\overline{AB} + \overline{BC} \ll \overline{AC}$$

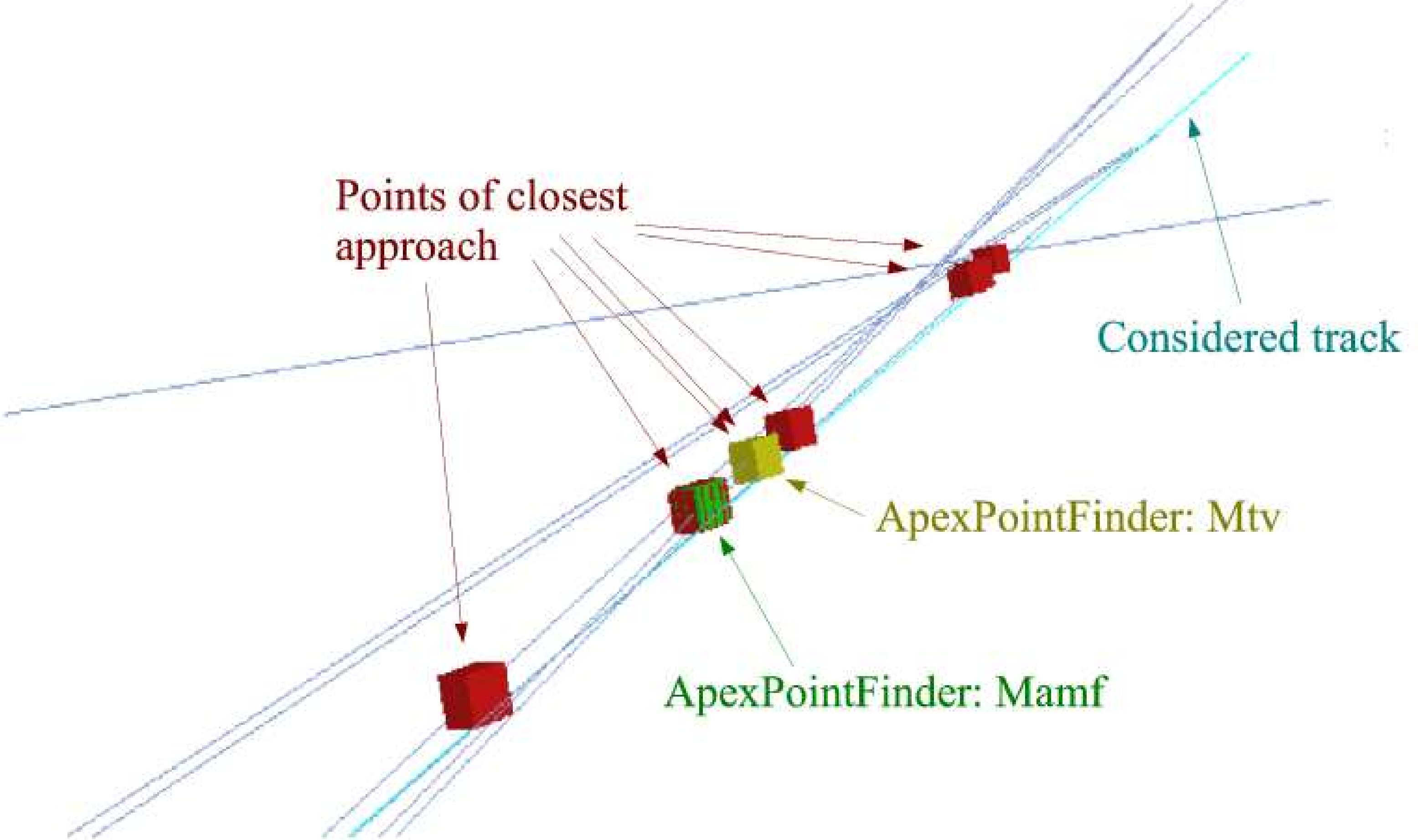
q.e.d.



→ Dmin (a.k.a. single linkage, minimum spanning tree) = bad choice

Apex Points

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



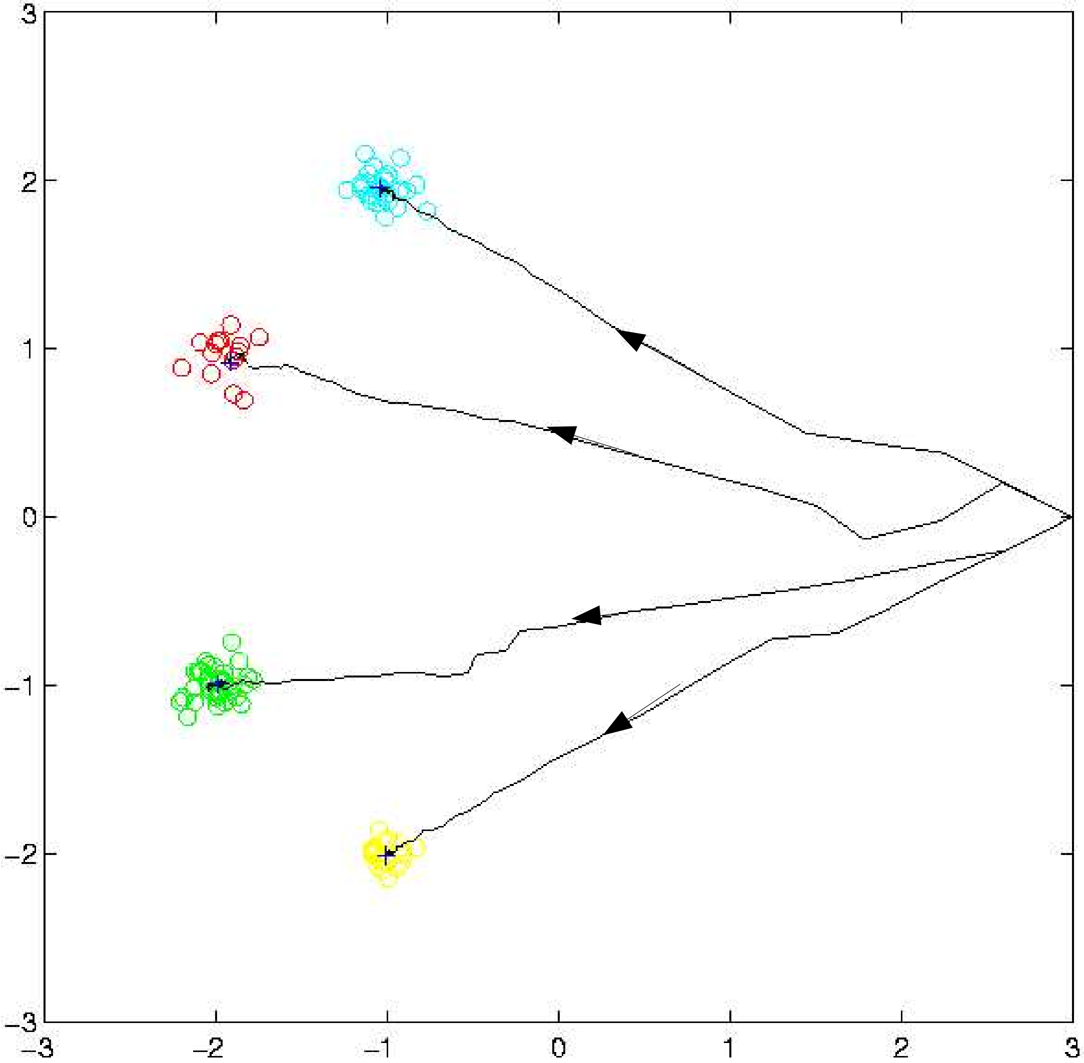
Vector Quantisation

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

Learning algorithm:

frequency sensitive competitive learning.

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



Global Association Criterion

The "weights" as we have defined them for the AdaptiveVertexFitter, can also be used to define a global "plausibility" criterion of the result of a vertex reconstructor: the GlobalAssociationCriterion:

Let w_{ij} be the weight of track i with respect to vertex j.

The "penalty" p_{ij} can then be defined as:

$$p_{ij} = \begin{cases} 1 - w_{ij} & \text{if } i \in j \\ w_{ij} & \text{if } i \notin j \end{cases}$$

The average of all penalties then makes up the GlobalAssociationCriterion.

Global Association Criterion (2)

What can the GlobalAssociationCriterion be used for?

- exhaustive vertex finding algorithm

  information-theoretic limit?

  equivalence to Minimum Encoding Length [MEL]?

- stopping criterion for other algorithms.
  - "SuperFinder" algorithms

    combines the results of two finders into one better finder
  - ...
  - work in progress, no detailed results yet.

Vertex Finding Results

Recap: tuning and score

Finetuning process needs a score.

Score needs PerformanceEstimators:

VertexFindingEfficiencyEstimator: How many reconstructible simulated vertices were found.

VertexPurityEstimator: How many wrong tracks are in the reconstructed vertices.

VertexTrackAssignmentEstimator: How many assigneable tracks were assigned to a vertex.

FakeRateEstimator: how many fake vertices were found.

-> Score:

$$Eff_P^{\,a} \,.\, Eff_S^{\,b} \,.\, Pur_P^{\,c} \,.\, Pur_S^{\,d} \,.\, Ass_P^{\,e} \,.\, Ass_S^{\,f} \,.\, (1\text{-}Fake)^g$$

Simulation experiments

Performance was analysed against:

full fledged Monte Carlo events

50 GeV b-jets (one primary vertex, one 'signal' secondary vertex)

$\eta \leq 1.4$

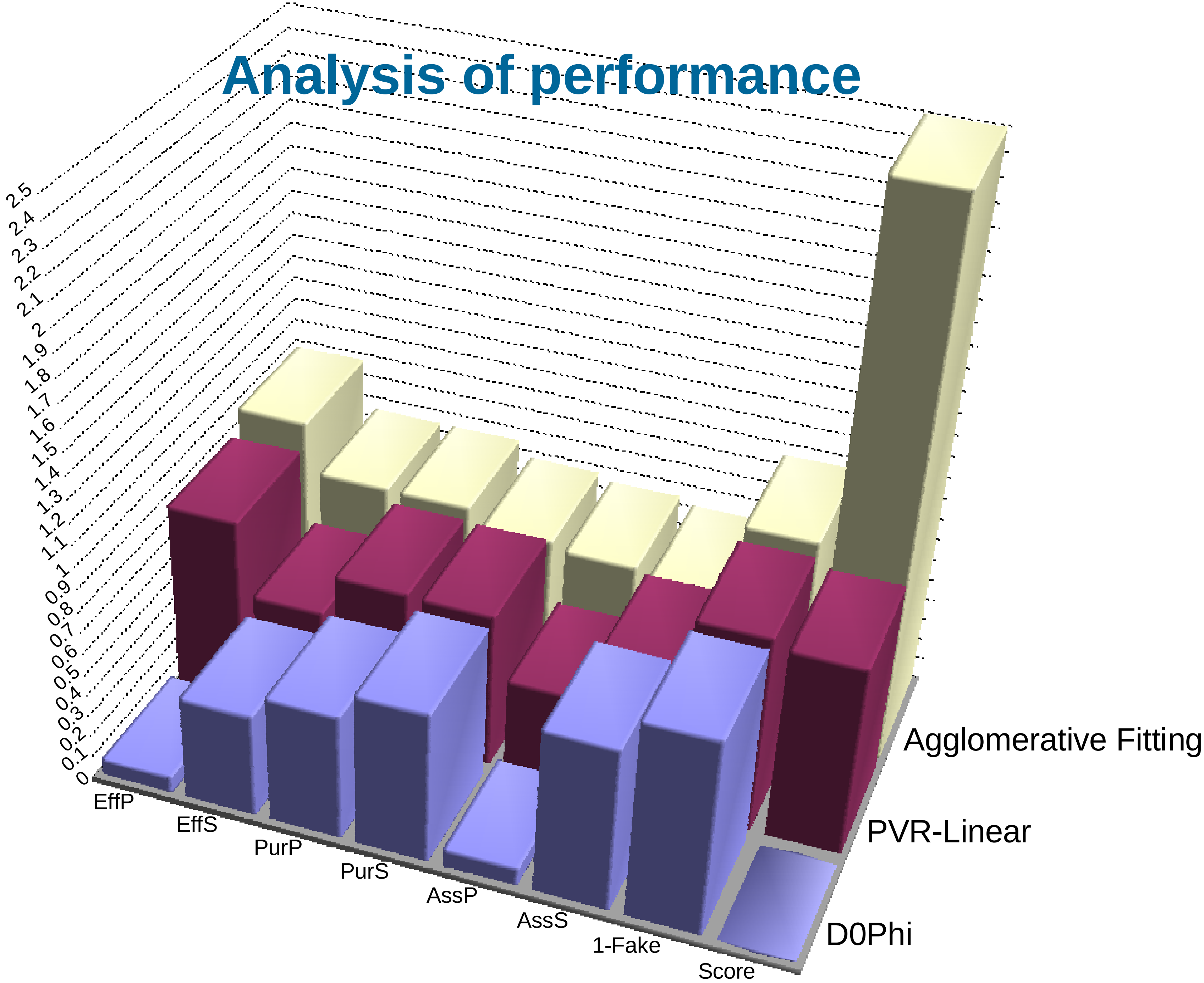
finetuning:

1000 events final round, 200 events per pre-round.

score :

$Eff_P^{\,2} \, . \, Eff_S^{\,2} \, . \, Pur_P^{\,0.5} \, . \, Pur_S^{\,0.5} \, . \, Ass_P^{\,0.5} \, . \, Ass_S^{\,0.5} \, . \, (1\text{-}Fake)^1$

Analysis of performance



Conclusions

Current algorithms seem to be quite good already.
But: can we do even better?

Future plans:

- another 'learning algorithm':
  Potts neurons or super-paramagnetic clustering (SPC)
- GlobalAssociationCriterion:
  theoretical and practical exploration of GlobalAssociationCriterion and its potential applications.
- and, most importantly:

Tests, performance analyses, case studies

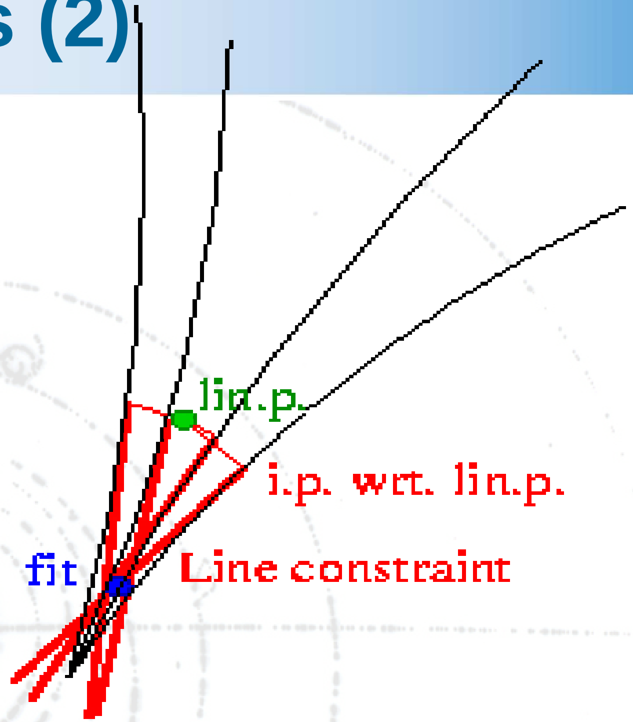
Backup slides

Least square methods (2)

LinearVertexFitter

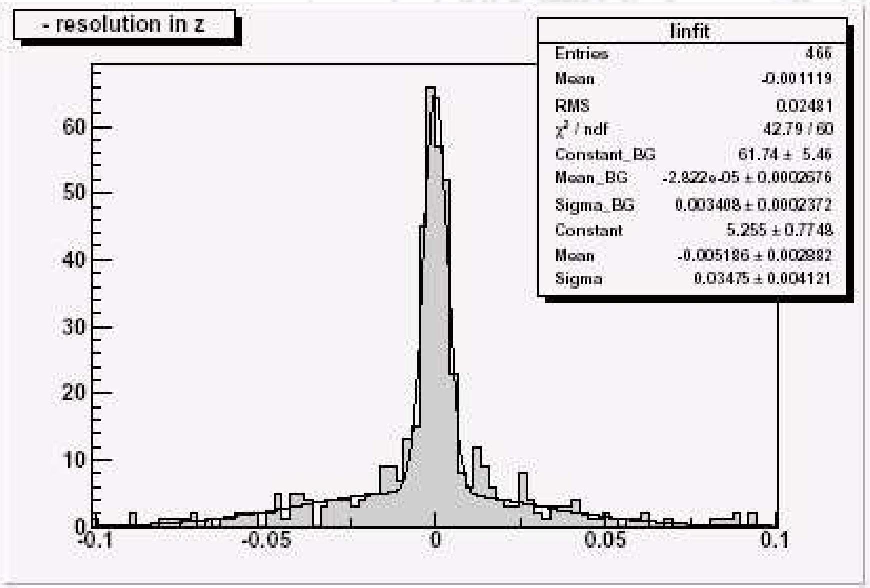
V.Karimäki, CMS Note 1997/051

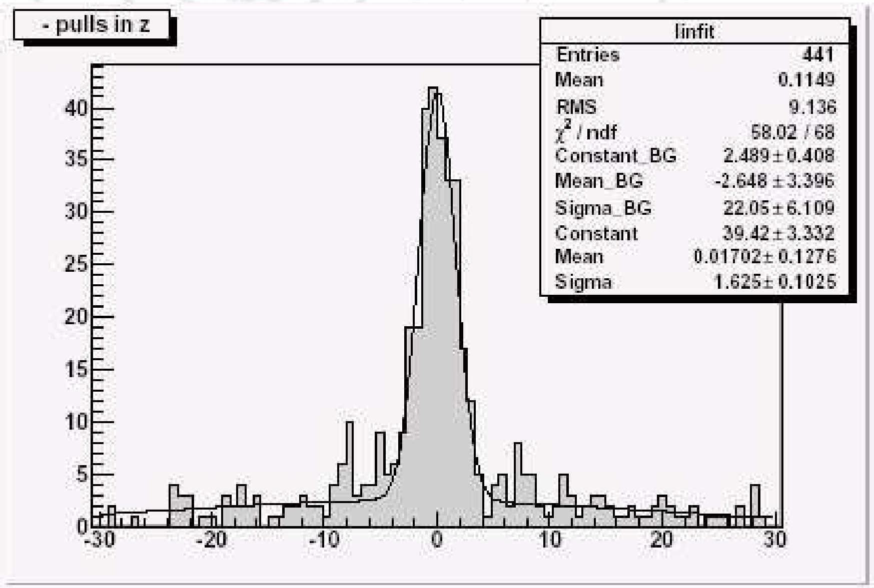
- works with p.c.a.'s in 3D
- Straight line approximation of tracks at linearization point



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

Least squares fit, z-resolution and pull





Least squares methods(1)

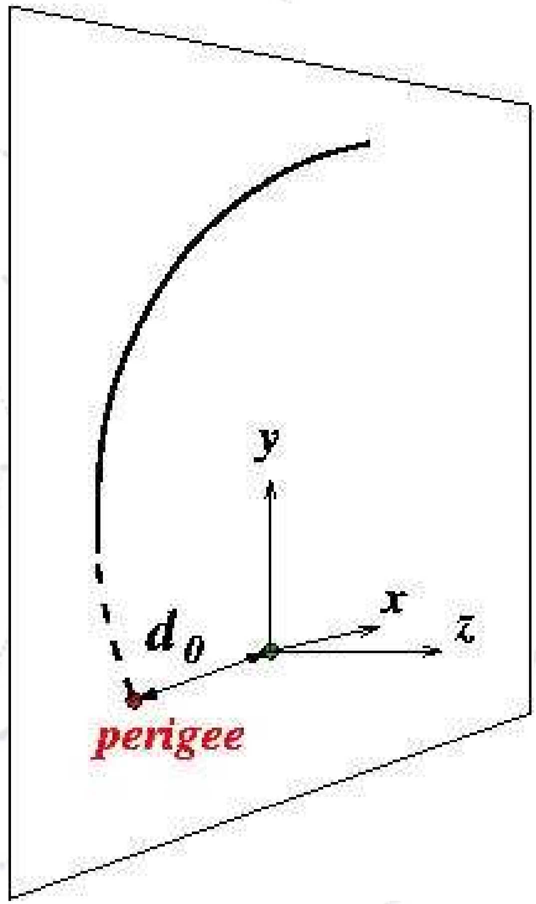
KalmanVertexFitter

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

- works with track parameters at perigee

P.Billoir et al., NIM A311(1992) 139-150

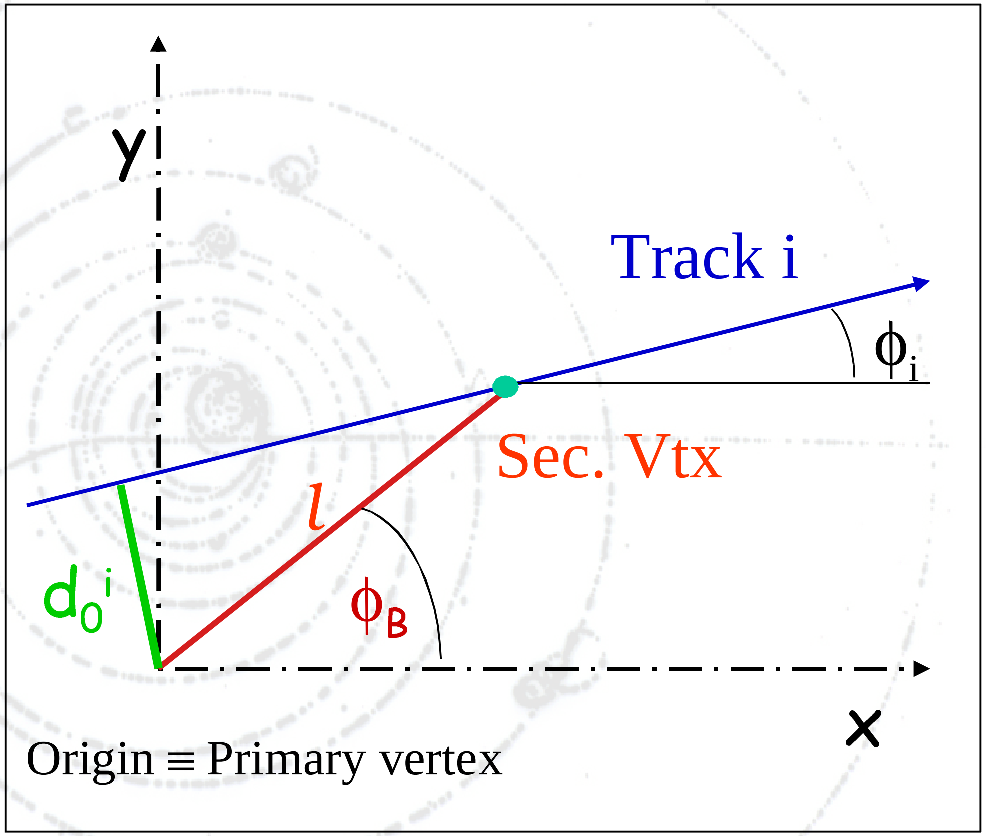
- helix approximation of tracks
  - 5 parameters at the perigee:
  - $(\rho, \theta, \varphi_p, \varepsilon, z_p)$
  - ρ signed transverse curvature
  - θ polar angle
  - φ_p azimuthal angle at perigee
  - ε signed d_0
  - z_p z-coordinate at perigee



d_0-ϕ algorithm (CDF)

Agglomerative algorithm,
search for secondary vertices

- geometrical correlations of tracks from same secondary vertex
  - track ≡ straight line around the primary vertex
  - projection onto well-chosen plane
  - tracks from same secondary vertex have same l and ϕ_B in that plane



$$D_0^i = l \sin(\phi_i - \phi_B) \approx l.(\phi_i - \phi_B)$$

d_0-ϕ algorithm (cont.)

In (d_0,ϕ) plane:

- 1 point for each track
- tracks from same secondary vertex:
  - have $d_0 \neq 0$
  - aligned along segment of positive slope ($l > 0$)
- tracks from primary vertices:
  - have $d_0 \approx 0$