



# Vertex reconstruction in CMS

T.Boccali, E.Chabanat, N.Estre, R.Frühwirth, V.Karimäki, D.Kotlinski, S.Moreau, K.Prokofiev, R.Ranieri, G.Segneri, T.Speer, N.Stepanov, T.Todorov, P.Vanlaer, W.Waltenberger

*Workshop on B/ $\tau$  Physics at LHC in Helsinki, May 30 - June 1, 2002*

## Outline

- Introduction: tasks, boundary conditions
- Vertex fitting
- Vertex finding
- Testing
- Conclusions



# Introduction

## Tasks

- **Fitting of a single vertex**
  - most precise vertex position and track parameters at vertex
- **Vertex finding**
  - separation of primary vertices (presentation from D.Kotlinski)
  - search for (at least 1) secondary vertex inside a jet
    - typical application: *b*-jet tagging
  - reconstruction of decay chains



## Boundary conditions

### Separation of primary vertices along beam axis

- At high LHC luminosity:  $\langle 17 \rangle$  primary vertices, 1 hard event, spread with  $\sigma_z = 5.3$  cm

- CMS Tracker resolution on longitudinal impact parameter  $z_0$ :

- $\sigma(z_0) = f(p_T, \eta)$

- $p_T = 1 \text{ GeV}/c: 90 \rightarrow 800 \mu\text{m}$

- high  $p_T: 20 \rightarrow 80 \mu\text{m}$

$\Rightarrow$  Hard event easy to separate in  $z$

$\Rightarrow$  Search for secondary vertex can proceed in Region Of Interest defined by (CALO jet cone + primary vertex)

$\Rightarrow$  Association of tracks to primary vertex: less easy



## Boundary conditions (2)

### Search for secondary vertex inside $b$ -jet

- Resolution on transverse impact parameter  $d_0$ :

- $\sigma(d_0) = f(p_T, \eta)$

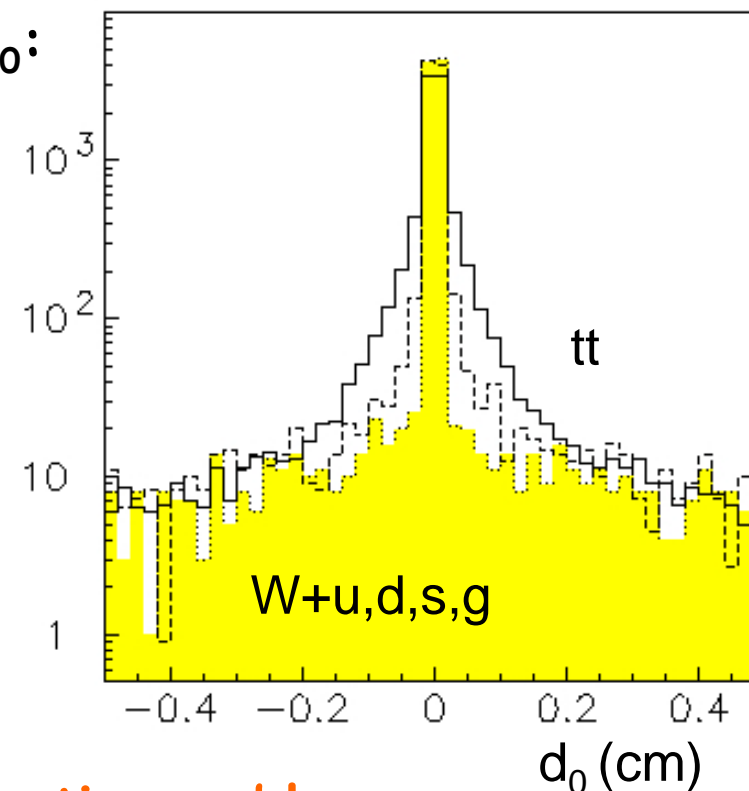
- $p_T = 1 \text{ GeV}/c$ :  $80 \rightarrow 200 \mu\text{m}$

- high  $p_T$ :  $10 \rightarrow 20 \mu\text{m}$

- $d_0$  in top pair,  $W+c$ ,  $W+u,d,s,g$ -events:

- impact parameter of secondaries  $\sim \text{mm}$

- A few percent of mismeasured tracks, with much worse resolution



⇒ Interesting association / minimization problem

⇒ 3D reconstruction should work better than reconstruction in transverse plane only



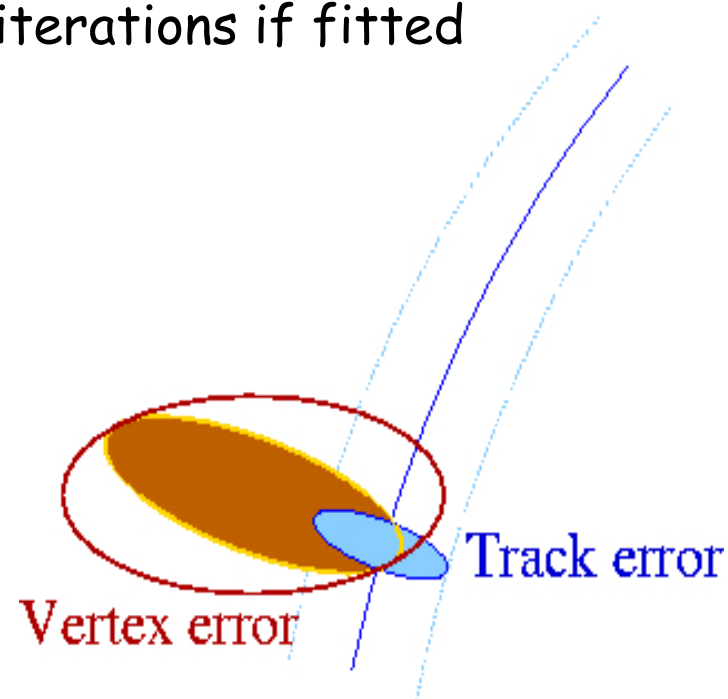
# Vertex fitting

## Principle

- **Minimization problem**
  - minimize a function of track-to-vertex distances
- Each track constrains the vertex by a helix + error (2D constraint)
- Non-linear, but explicit linear solution exists if tracks are linearized in the vicinity of the vertex position
  - requires a first guess of vertex position, and iterations if fitted position is too far from first guess

## Techniques

- **Least sum of Squares (LS):**
  - all tracks are used unweighted
  - function to minimize is usual total  $\chi^2$
  - biased if vertex contains an outlier, i.e. track from another vertex or badly measured track
- **Robust estimators:**
  - insensitive to outliers

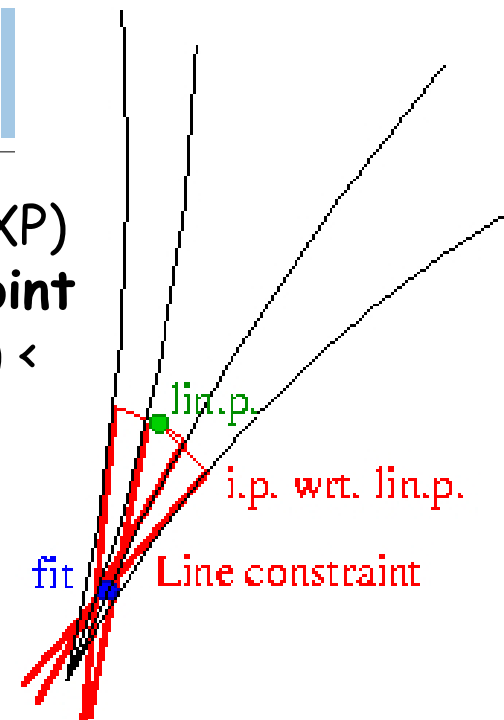




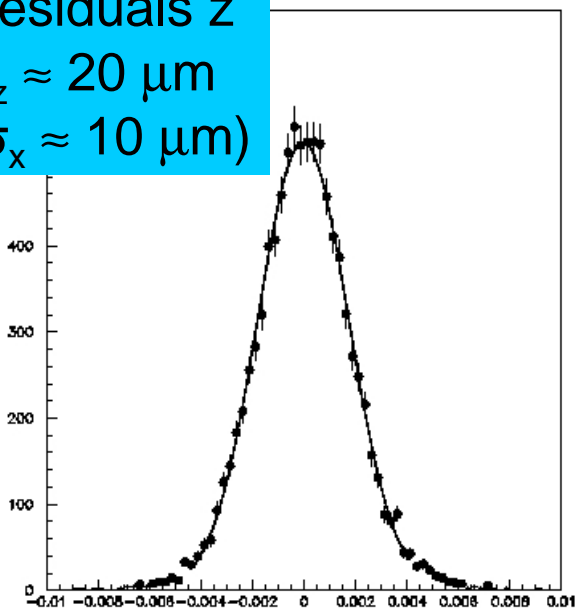
## LS: Linearized vertex fit

**Very fast, precise algorithm** (V.Karimäki, HIP-1997-77/EXP)

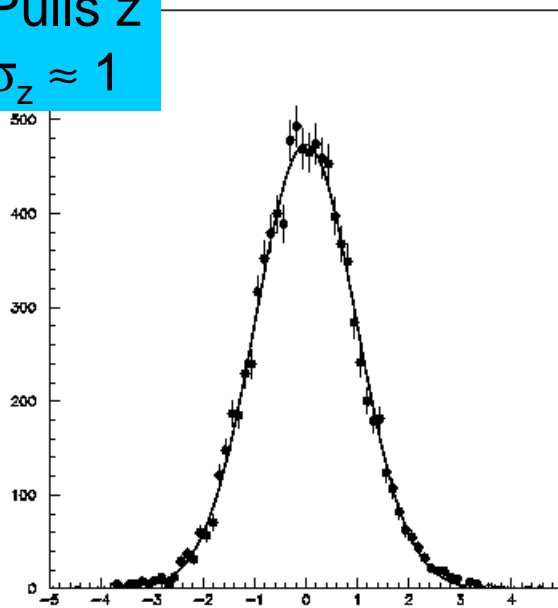
- **straight line approximation of tracks at linearization point**
  - correct to a few  $\mu\text{m}$  for  $p_T > 1 \text{ GeV}/c$  if  $d(\text{lin.point}-\text{vtx}) < \text{few mm}$
- **track error matrix  $\sim$  constant around lin.point**
- **explicit solution involving only 3x3 matrix algebra**



Residuals  $z$   
 $\sigma_z \approx 20 \mu\text{m}$   
( $\sigma_x \approx 10 \mu\text{m}$ )



Pulls  $z$   
 $\sigma_z \approx 1$



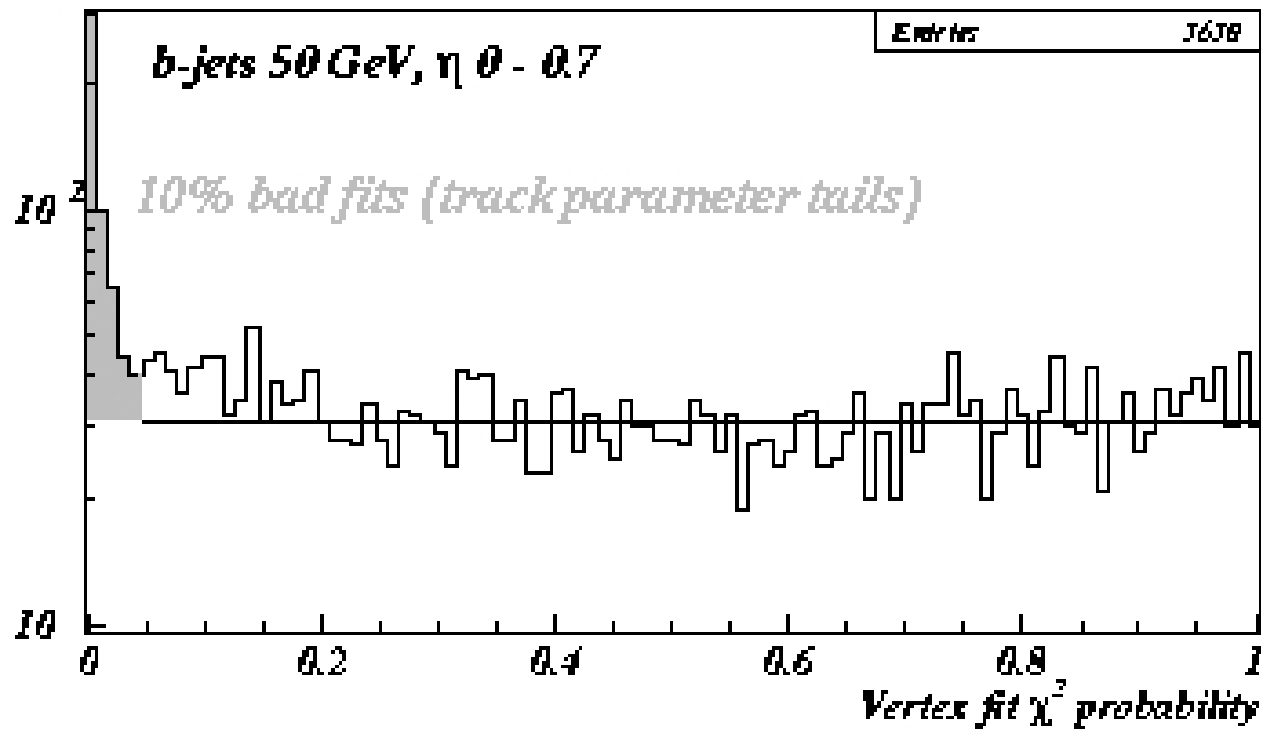
10000  $H \rightarrow 4\mu$  events  
LinearVertexFitter

Kirill Prokofiev,  
Thomas Speer



## Linearized vertex fit (2)

- **Distribution of  $\chi^2$  probability**
  - ~flat as expected
  - powerful test to reject fake vertices
  - 10% bad fits due to tails of  $d_0$  and  $z_0$  pull distributions



All vertices in *b*-jets  
 $P(\chi^2)$ , tracks selected using  
MC information

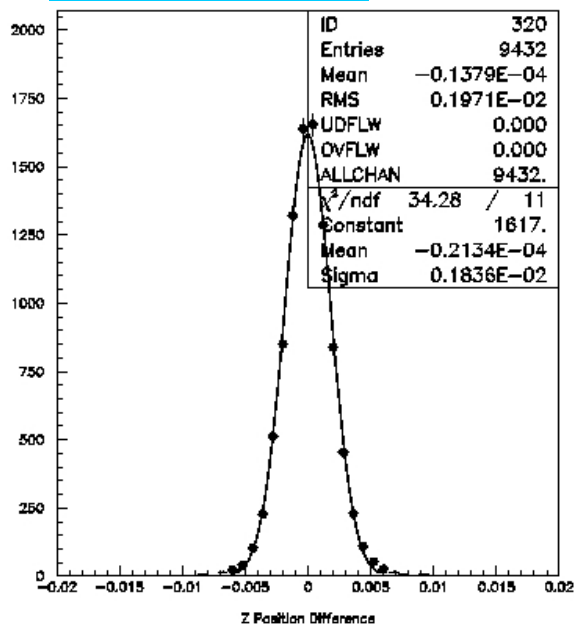


# LS: Kalman vertex fit

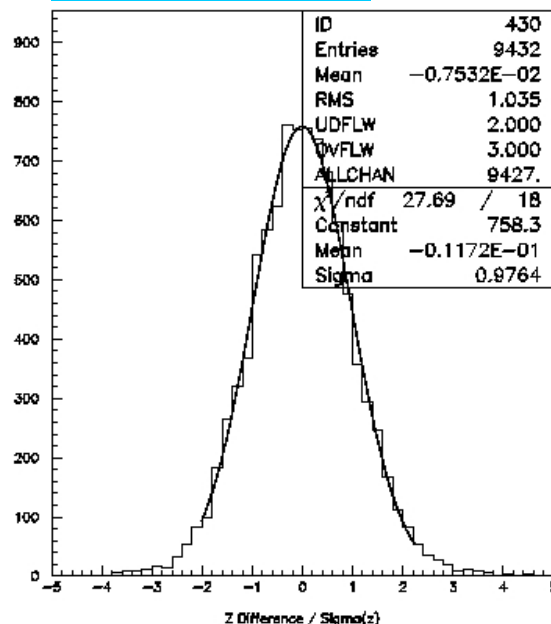
## Kalman formalism (R.Frühwirth et al.)

- allows parabolic track approximation instead of straight line
  - should be more precise for low  $p_T$  tracks
  - should require less iterations

### Residuals z



### Pulls z



10000  $H \rightarrow 4\mu$  events  
KalmanVertexFitter

Identical performance as  
LinearVertexFitter for high  
 $p_T$  tracks

Kirill Prokofiev, Thomas Speer





## Robust estimators

**Robust** = insensitive to outlying tracks

- **most distant tracks discarded** (Least Trimmed Squares, Least Median of Squares, Minimum Volume Ellipsoid, Minimum Covariance Determinant,...)
- **distant tracks downweighted** (M-estimator, adaptive algorithms,...)
- CPU-expensive but fast, approached algorithms exist (tried here)

**Least Trimmed Squares** (P.Rousseeuw et al.)

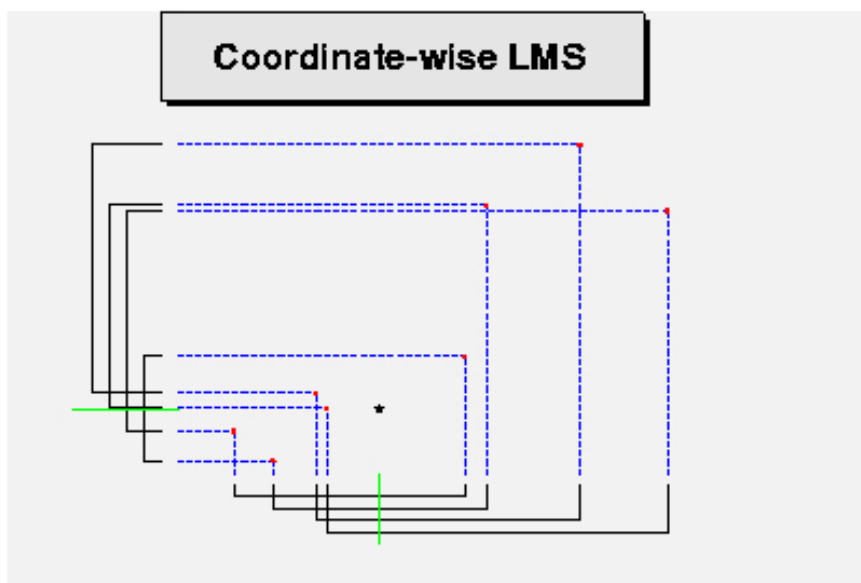
- use  $k$  most compatible tracks out of  $N$  ( $k/N$ : trimming fraction)
- **breakdown point**  $\approx 1 - k/N$
- has good statistical properties
  - estimator has normal distribution, precision improves as  $1/\sqrt{N}$
- Exhaustive: try all possible combinations of  $k$  out of  $N$
- FAST-LTS: good approached iterative algorithm



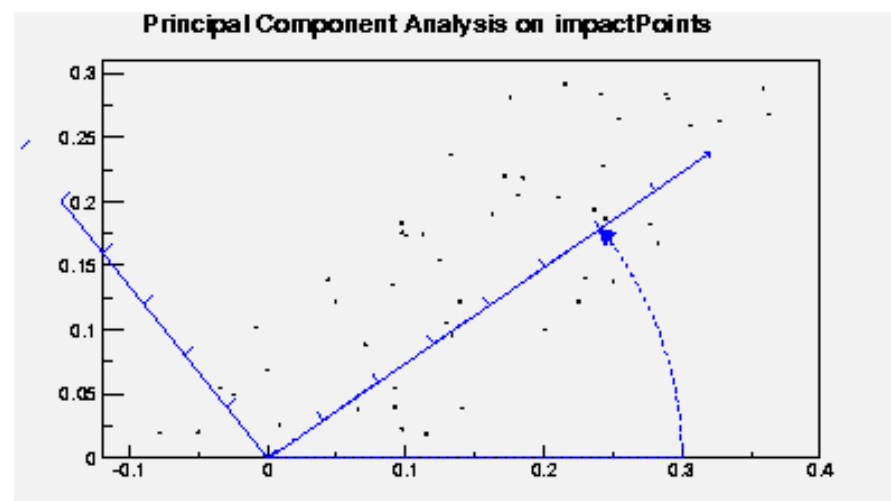
## Robust estimators (2)

### Least Median of Squares (P.Rousseeuw et al.)

- Fast implementation: medians of coordinates of track impact points wrt. linearization point:



### + Principal Component Analysis



- very robust (breakdown point = 0.5)
- worse statistical properties
  - not normal, precision improves as  $N^{-1/3}$

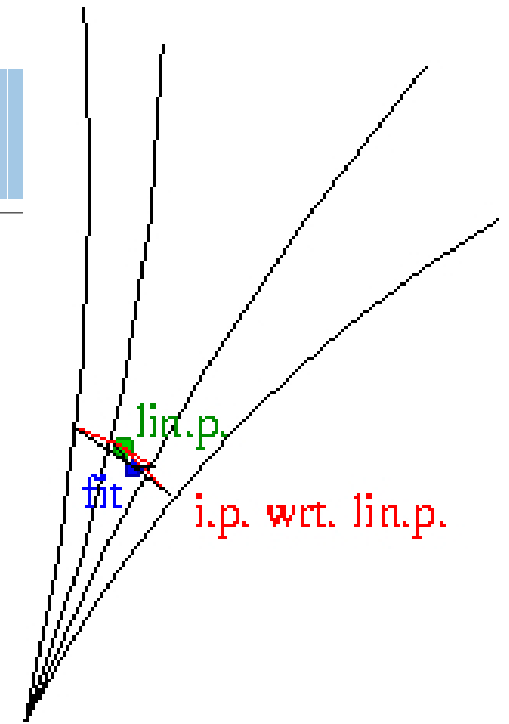


## Robust estimators (3)

### Least Median of Squares (cont.)

- this implementation: sensitive to choice of initial linearization point

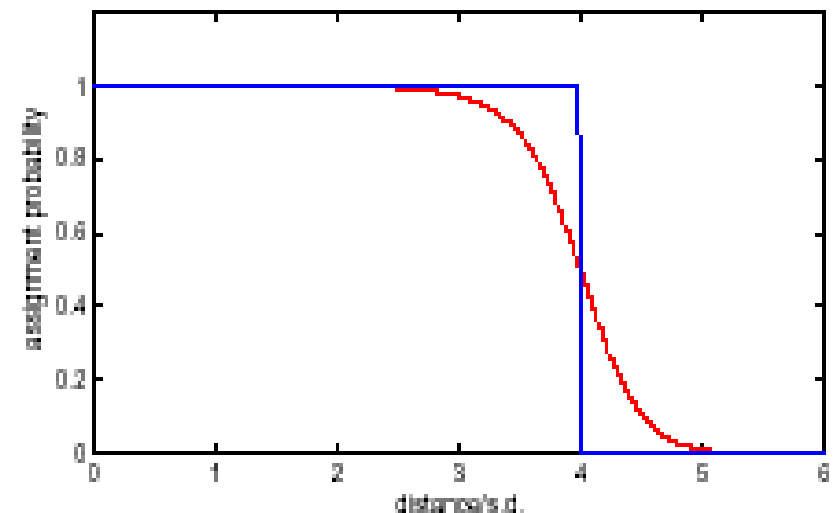
⇒ Fast, precise linearization point finder:  
LMS on crossing points of *track pairs*



### Adaptive vertex fit

- iterative, re-weighted LS fit
- weight  $w$  of track  $i$  at iteration  $k$  depends on distance  $r$  to vertex at iteration  $k-1$ 
  - $w(r) \equiv$  assignment probability
- same good properties as LS estimators

Examples of weight functions  $w(r)$   
for a cut at  $r = 4$





# Robust estimators: results

	Mean (mkm)	RMS (mkm)	Failed	msec./fit 1GHz PC
Linear	50	45	none	15
LMS**	85	160	none	13
LTS (0.8)	16	26	none	65
MCD (0.8)	21	19	none	62
Adaptive**	50	45	none	29

300 vertices from VertexGun  
50 tracks + 5 outliers  
Mean and RMS of distance(sim - rec)

*\*no error matrix estimation*

*"2tk lin. point finder*

*\*\*very preliminary*

Wolfgang Waltenberger

- **robustness pays in precision**
- **CPU not so bad (only factor 4 slower than linearized LS fit)**

VertexFitter / LinPointFinder	Mean (mkm)	RMS (mkm)	msec./fit 1GHz PC
LTS(0.95) / 2tk	43	50	76
LTS(0.95) / LMSLP	15	10	58

50 vertices from VertexGun  
50 tracks + 2 outliers

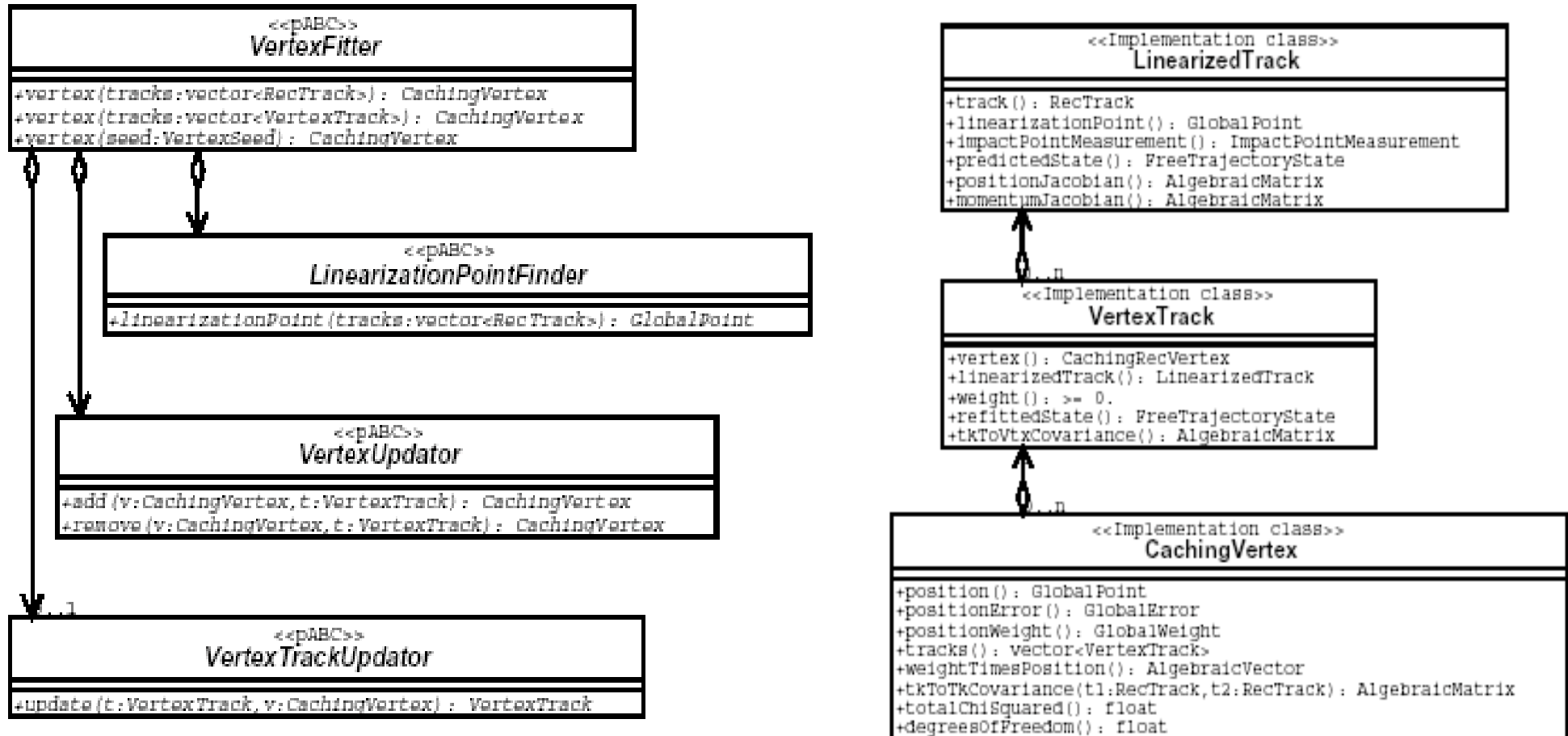
Wolfgang Waltenberger

- **robust estimators sensitive to linearization point**
- **LMS on crossing points of track pairs seems to work well**



# Decomposition in classes

## UML diagrams:



All functional components are clonable



# Vertex finding

**Variety of algorithms** (R.Frühwirth, CMS mini-workshop on vtx reconstruction)

**Two simple working algorithms in CMS**

- **D0 $\phi$  algorithm**

- exploits correlation between  $d_0$  and  $\phi$  for tracks from same vertex

- **Principal vertex finding**

- finding-trough-fitting: fits all tracks to a common principal vertex, discards incompatible tracks, then looks for a secondary vertex among discarded tracks

- **More evolved algorithms (competitive learning, multi-vertex fitting, ...)** need guess of number and position of vertices

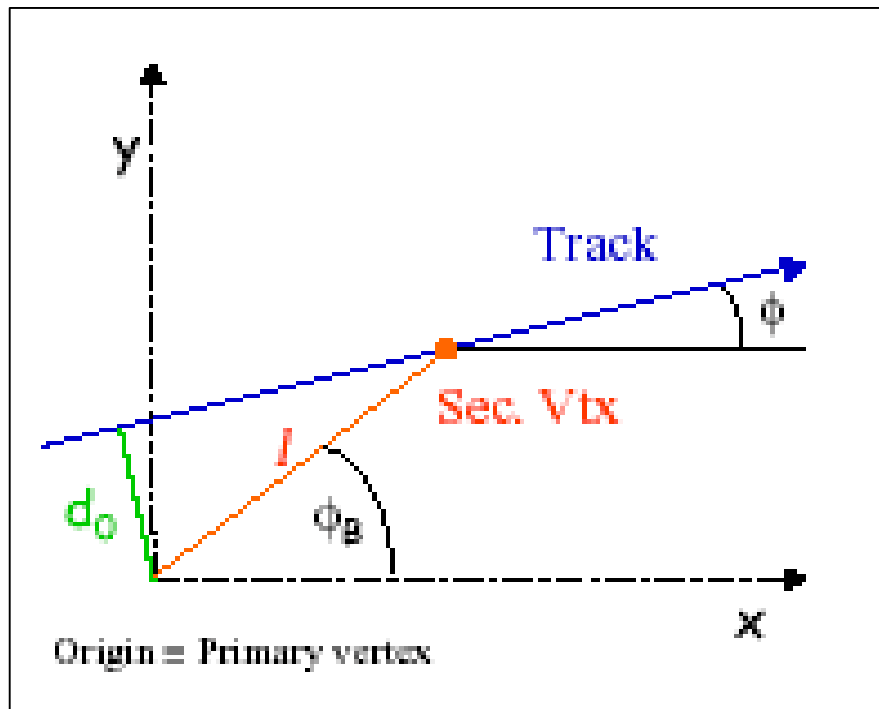
⇒ basic building blocks are:

- a vertex fitting algorithm (LS, LTS, adaptive, ...)
- a vertex seed generator
  - Minimum Spanning Tree, Self-Organizing Map, ...
  - development started (Eric Chabanat, Nicolas Estre)

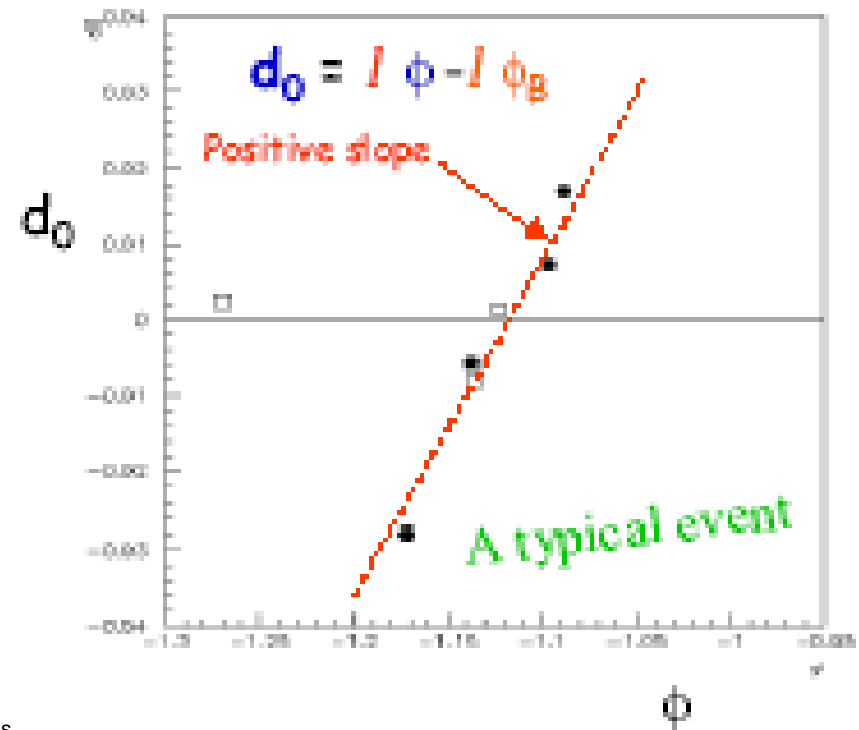


# D0 $\phi$ algorithm

- all tracks from same vertex have same  $l$  and  $\phi_B$   
 $d_0 = l \sin(\phi - \phi_B) \approx l \sin(\phi - \phi_B)$

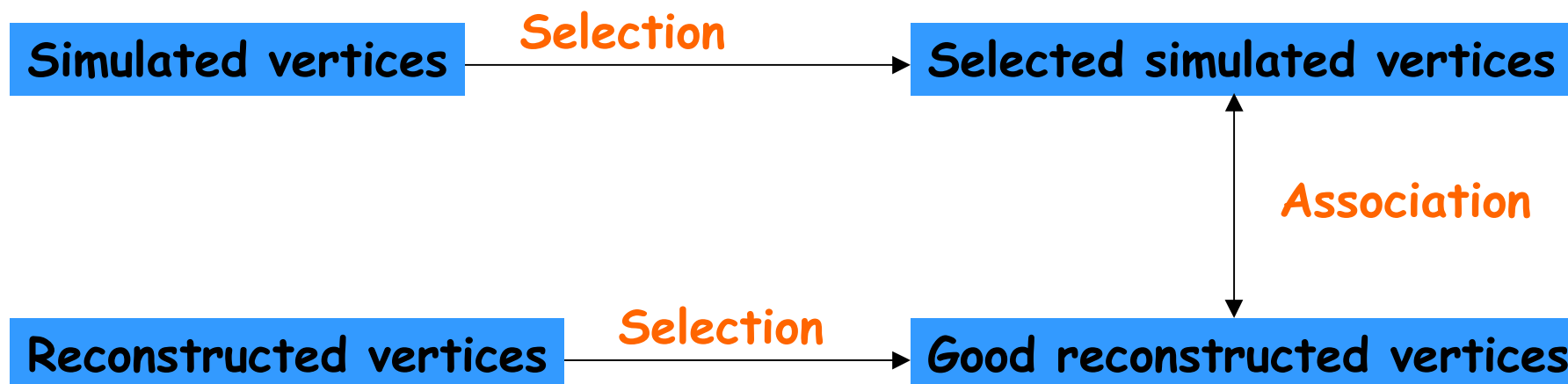


- in  $(d_0, \phi)$  plane
  - 1 point for each track
  - points from same secondary vertex aligned
  - form segments of positive slope and cluster them





# Analysis



Tommaso Boccali

- A simulated vertex is **found** if there is an associated reconstructed vertex
- A reconstructed vertex is **a fake** if there isn't any associated simulated vertex
- Association is done by tracks
  - a **reconstructed vtx** is associated to the **simulated vtx** from which the largest fraction of its tracks originate





## Analysis (2)

### Vertex selection

- simulated vertices:  $\geq 2$  reconstructed tracks
- reconstructed vertices:  $> 55\%$  of tracks from same sim. Vtx

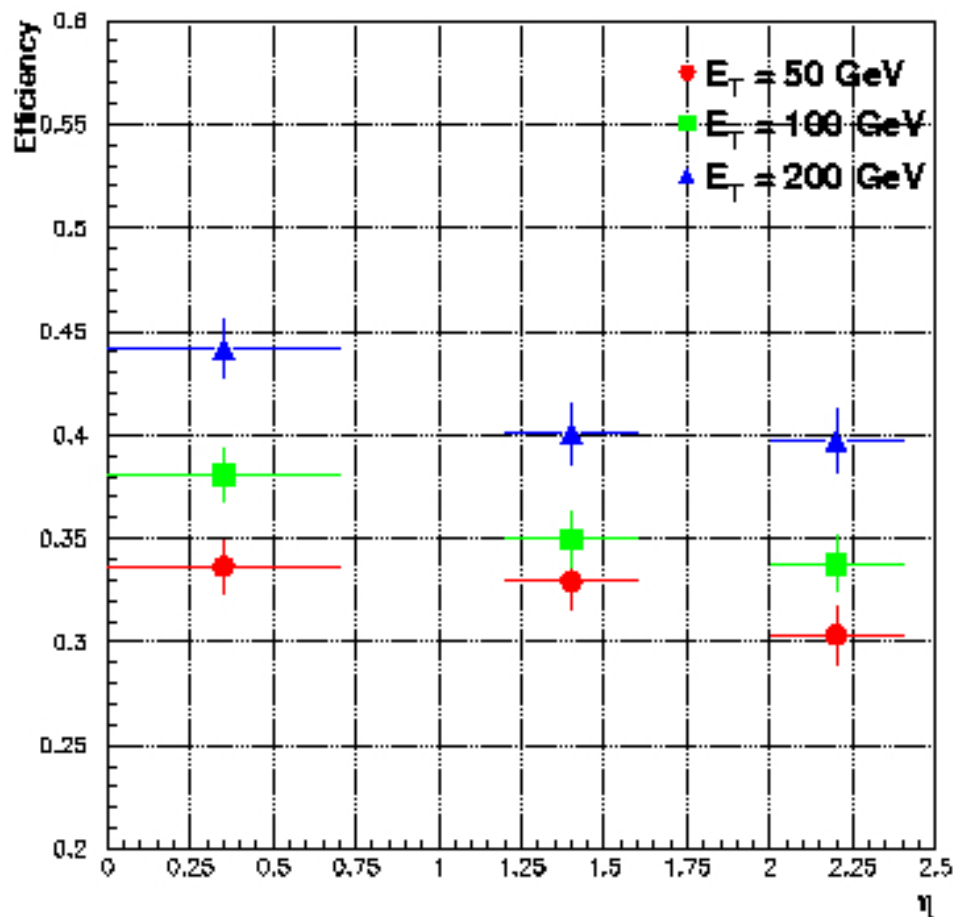
### Event samples

- di-jet events,  $E_T = 50, 100$  and  $200$  GeV,  $0 < \eta < 2.4$
- $b$ -jets: secondary vtx finding efficiency
- $u$ -jets: fake rate



# Results

## D0φ algorithm



Gabriele Segneri

## Principal vertex reconstruction

Principal vertex reconstructor	S.v. eff. (%)	Fake rate (fake/uu ev.)
ET = 50 GeV, all eta	33	0.006
100 GeV	36	0.009
200 GeV	32	

Pascal Vanlaer

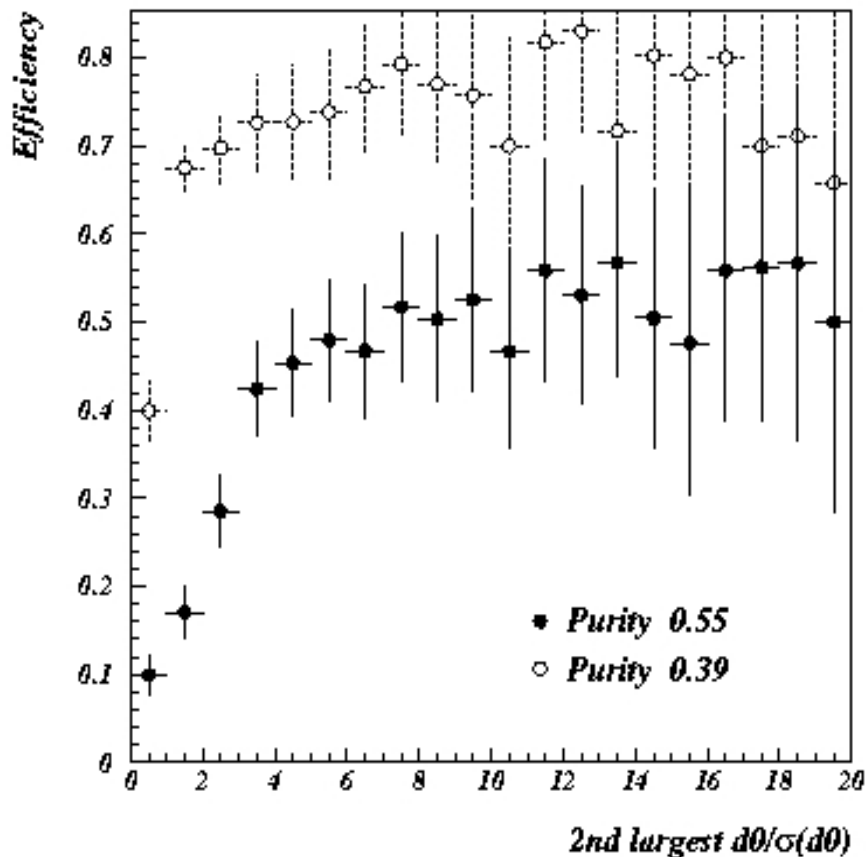
- Comparable results
- Fake rate very low
- Low efficiency... see next slide



## Results (2)

### Efficiency vs.

- separation of secondary vertex from beam
- selection of reconstructed vertices



S.V. finding efficiency vs. 2nd largest transverse impact parameter, 50 GeV b-jets, all  $\eta$   
Principal vertex reconstructor

### In summary:

- secondary vertices with small separation from beam ARE found (efficiency  $\approx 70\%$ )
- but up to 60% of their tracks do not originate from the right vertex

### Replace LS by robust fitter to improve:

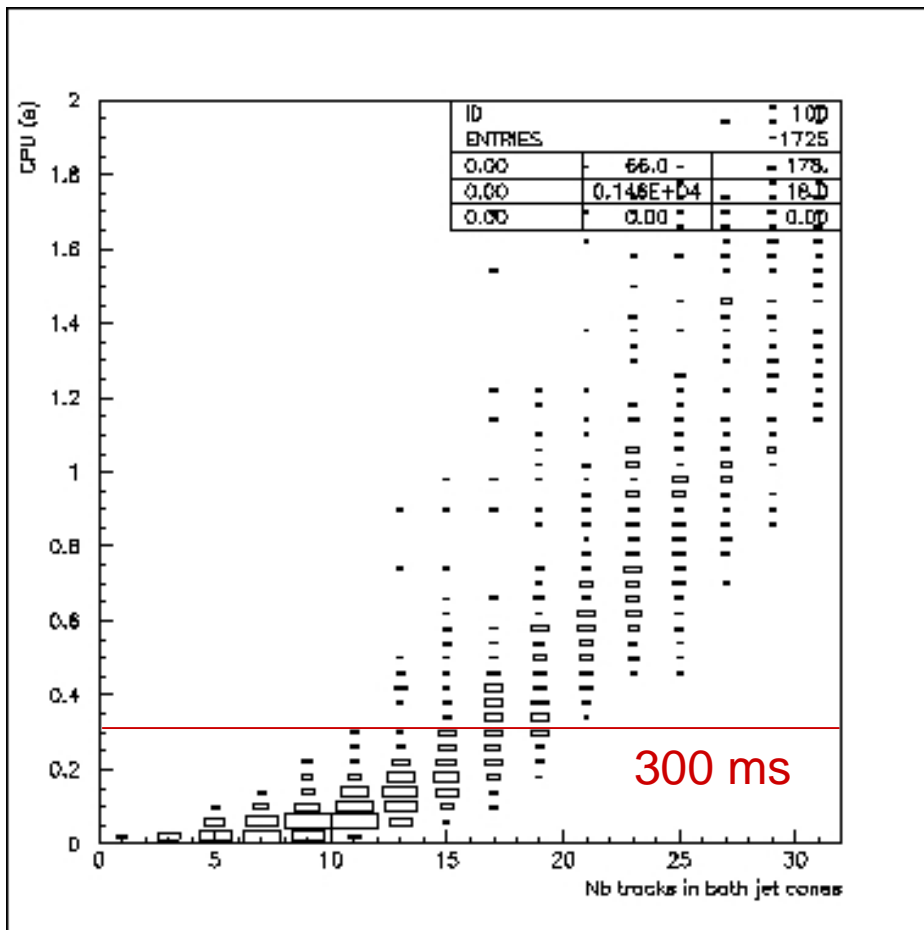
- primary vertex position thus separation of secondaries from primary vertex
- secondary vertex purity



## Results (3)

### Use of secondary vertex finding online

- Principal vertex reconstruction applied jet per jet
- CPU vs. number of tracks in both jets (1GHz PC, 50 GeV b-jets)



**Even inclusive search seems affordable online (after some optimization)**

- CPU time available @ CMS L2  $\equiv$  300 ms on today 1GHz CPU
- sizeable fraction of events processed in less than 300 ms

Pascal Vanlaer



## Testing

### Vertex reconstruction is at the end of event reconstruction chain

- affected by problems upstream
- use of full chain (database access; event simulation and track reconstruction if not done yet) for simple algorithm tests too slow

### VertexGun facility:

- allows generation of **user-defined kinematics** at vertex
- provides coarse track parameter smearing + error matrix
- ideal for e.g. **code release tests**

Wolfgang Waltenberger

### Fast Tracker Simulation (FTSim) facility:

- parametrizes track reconstruction performance
- tuned on full reconstruction
- provides **handles to deteriorate / improve track parameter resolutions and tails**
- ideal for e.g. **stability tests** wrt. tracker performance



# Conclusions

## CMS vertex reconstruction code offers:

- **fast, precise and reliable vertex fitting tools**
- **secondary vertex finding algorithms with satisfactory performance**
  - improvements on **CPU time for online**: in progress
  - improvements on **vertex purity for offline**: in progress
- **many performance analysis tools, still being completed**
- **code gets more and more used in analyses**

## Developing rapidly:

- **robust vertex fitting algorithms**
  - building blocks for advanced vertex finding algorithms
- **several tried for the first time in HEP**
- **testing facilities for fast algorithm development**

**Material, references,...:** CMS  $b/\tau$  page -> Activities -> Vertex; CMS  $b/\tau$  page -> Mini-workshops