Track and Vertex Reconstruction with the CMS Detector

Susanna Cucciarelli

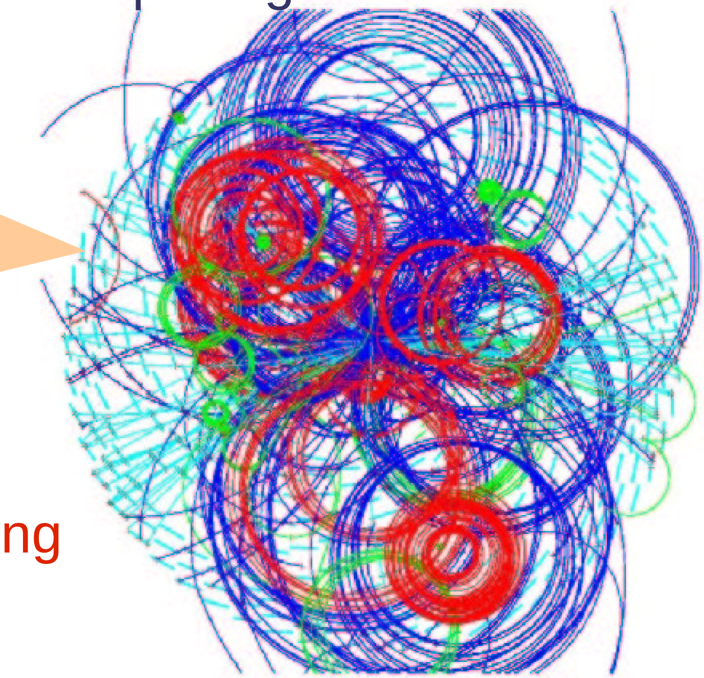
BEAUTY 2005
10th International Conference in B-Physics at Hadron Machines
Assisi (Italy) 20th-24th June 2005

Introduction

- proton-proton collisions at $\sqrt{s}$ = 14 TeV with bunch spacing of 25 ns
- Luminosity:
  - low-luminosity: $2 \cdot 10^{33}$ cm^{-2} s^{-1}
  - high-luminosity: 10^{34} cm^{-2} s^{-1}
    - ~ 20 minimum bias events per bunch crossing
    - ~ 5000 charged tracks per event

➔ Fast response time to resolve bunch crossing

➔ High precision to resolve nearby tracks



- Reconstruction of narrow heavy objects:

  4 T field - ~1.1m Tracker radius: 1.80 mm sagitta for 100 GeV/c p_T track!

- Ability to tag b jets through secondary vertices: good impact parameter resolution

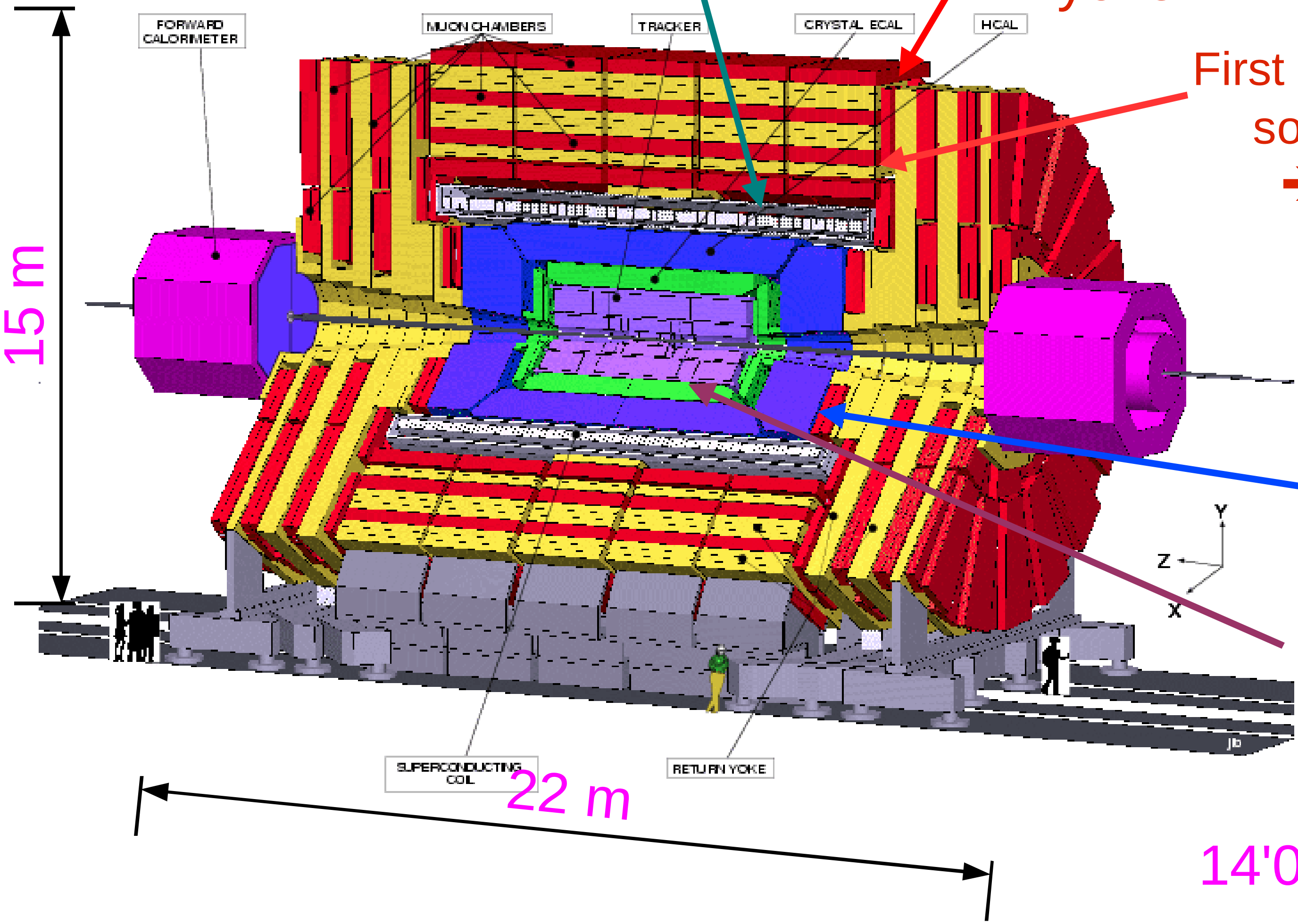
The CMS Detector

13x6 m Solenoid: 4 Tesla Field

Muon system in return yoke

First muon chamber just after solenoid

➔ extend lever arm for p_T measurement



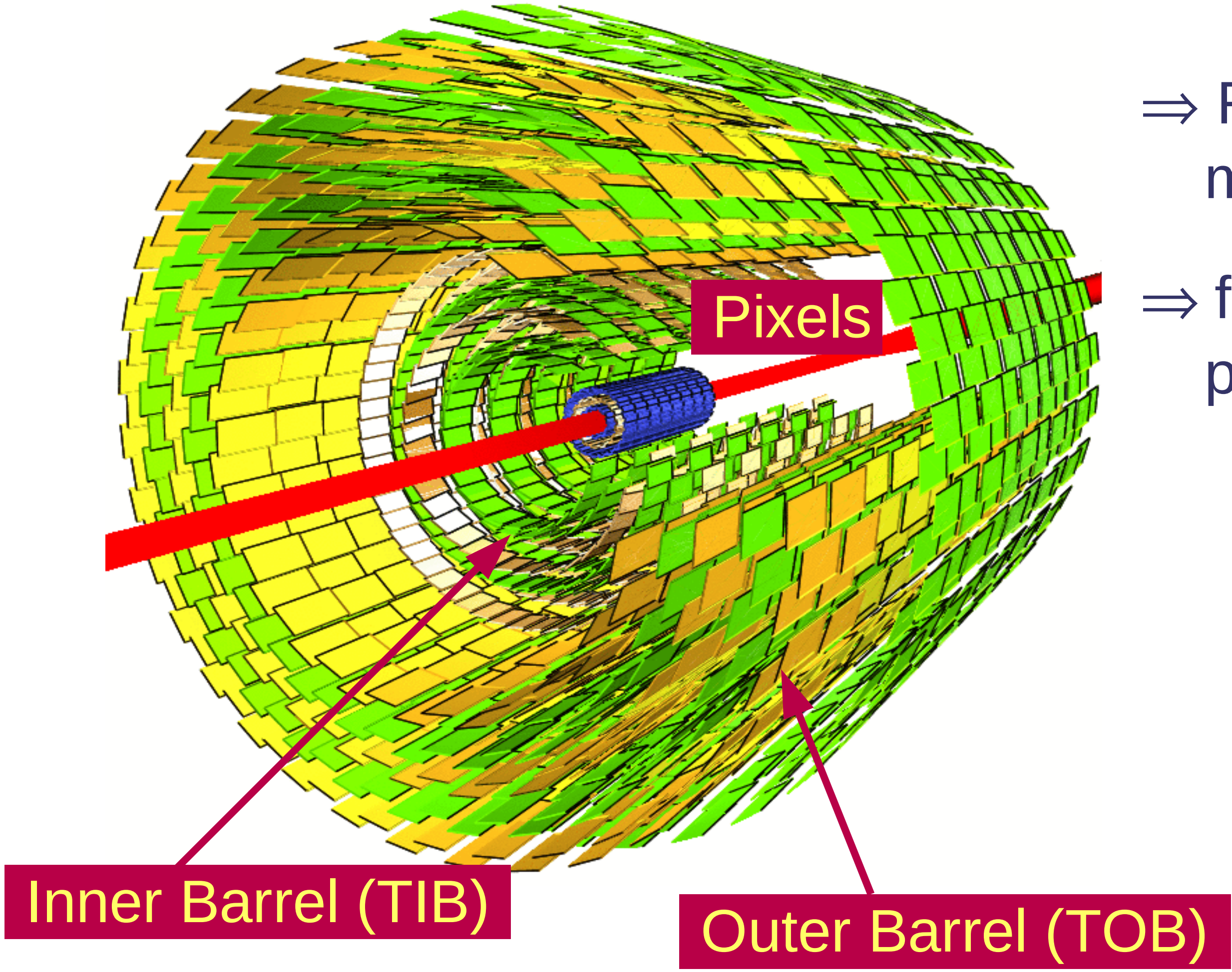
Calorimetric System (ECAL+HCAL)

Tracker System (Pixel+Silicon Strip)

14'000 Ton Detector

The CMS Tracker System

Configuration with all-silicon for the CMS Tracker System

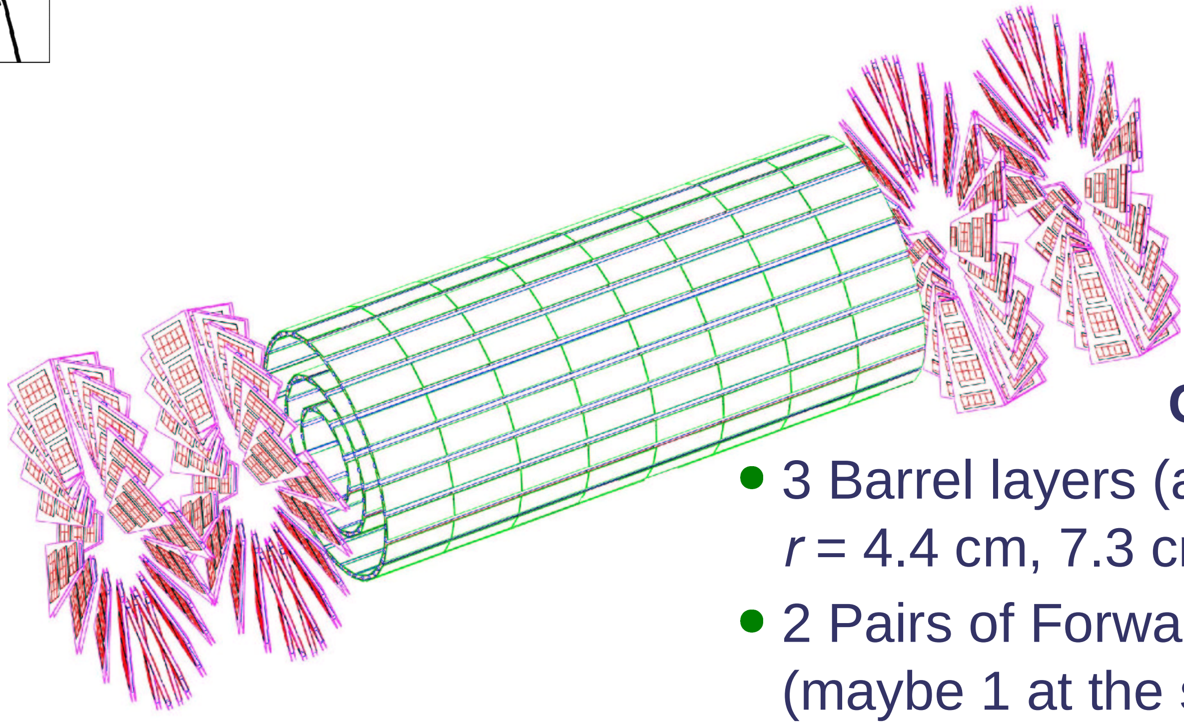


⇒ Rely on relatively few measurement layers

⇒ few and precise measurement points per track:

- 2 - 3 points from the Pixel Detector
- 10 - 14 points from the Silicon Strip Tracker

The CMS Pixel Detector



Geometry:

- 3 Barrel layers (at least 2 at the start up)
  r = 4.4 cm, 7.3 cm, 10.2 cm
- 2 Pairs of Forward/Backward Disks
  (maybe 1 at the start up)
  r = 6 cm-15 cm ; z = 34.5cm, 46.5cm

Active area ~ 1 m^2 with 66×10^6 pixels 100x150 mm^2 sized

$\Rightarrow$ 2-3 high resolution **3D measurement** points

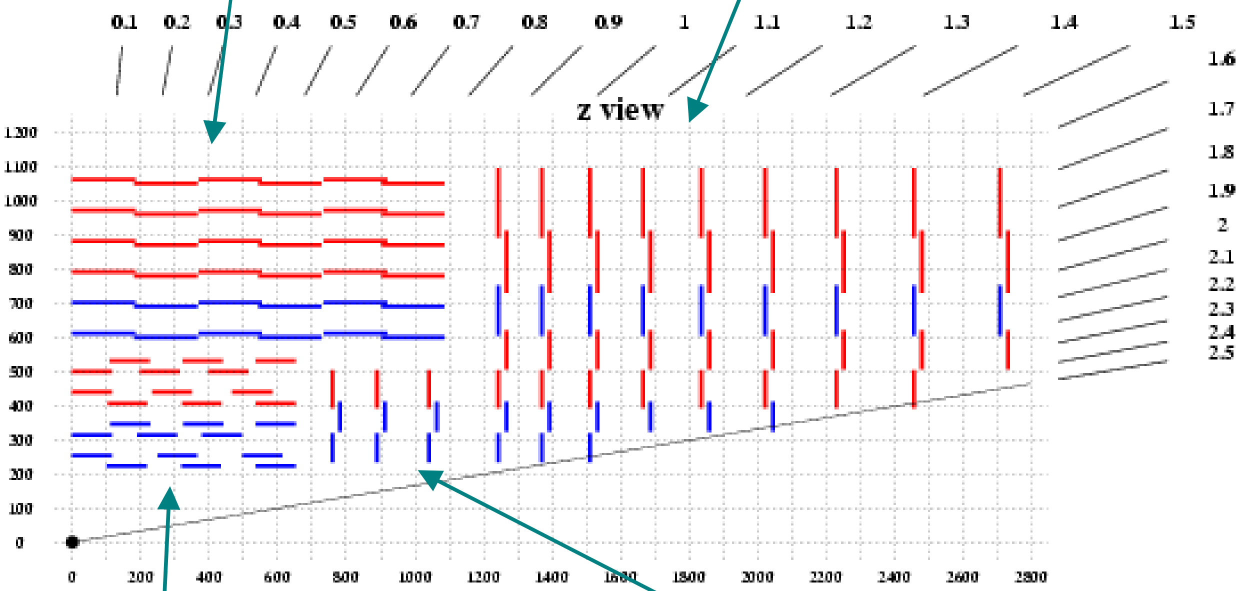
for **$|\eta|$ < 2.2** Lorentz angle ~ 23^o

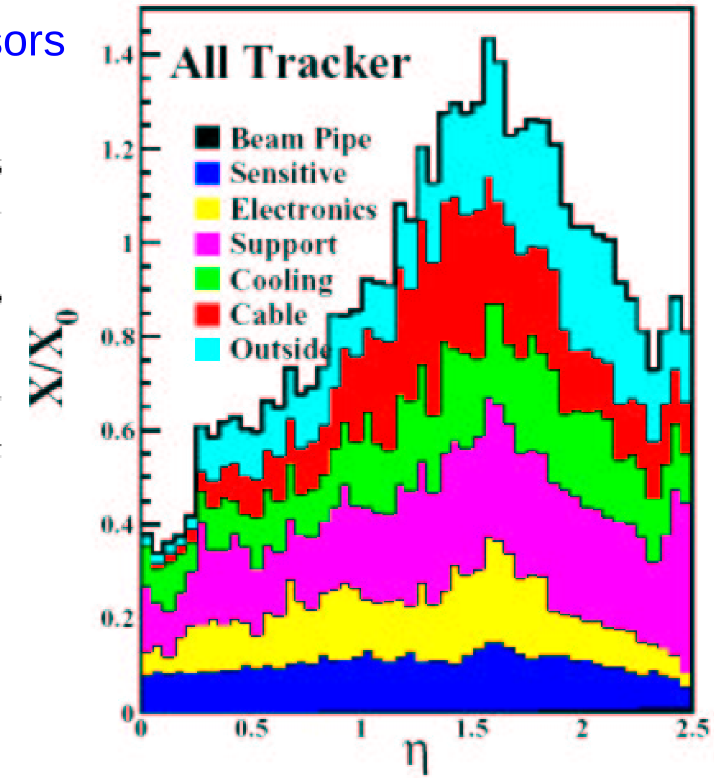
Spatial resolution: r-ϕ ~10μm and r-z ~ 15 μm

The CMS Silicon Strip Detector

Outer Barrel (TOB): 6 layers
- Thick (500 µm) sensors
- Long Strips

Endcap (TEC): 9 disks pairs
- r<60cm: Thin sensors
- r>60cm: Thick sensors



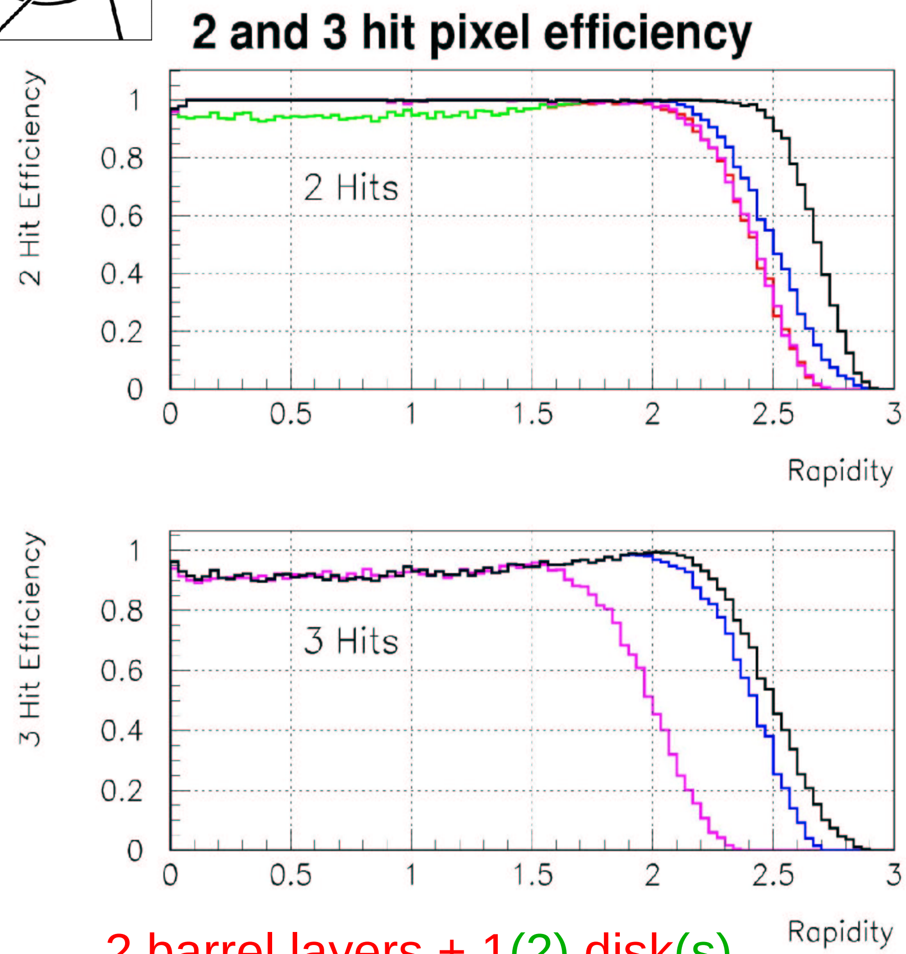


Inner Barrel (TIB): 4 layers
- Thin (320 µm)sensors
- Short Strips

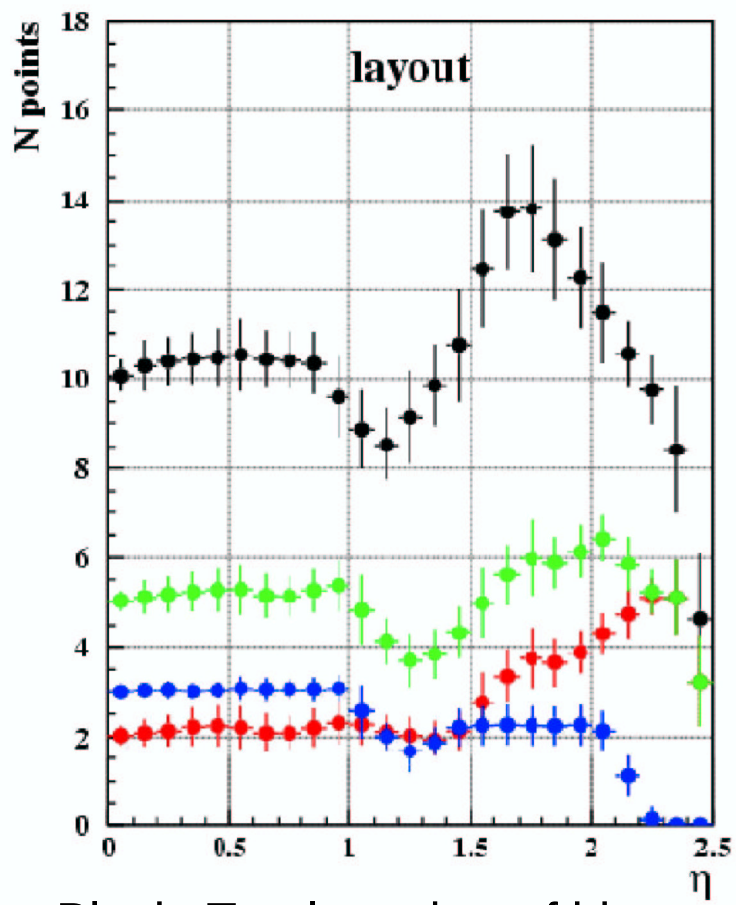
Inner Disks (TID): 3 disks pairs
- Thin sensors

Material Budget of the Tracker: significant contribution due to cabling, electronics and cooling

The CMS Tracker: Points per Track



2 barrel layers + 1(2) disk(s)
3 barrel layers + 1 disk
3 barrel layers + 2 disks
3 barrel layers + 3 disks



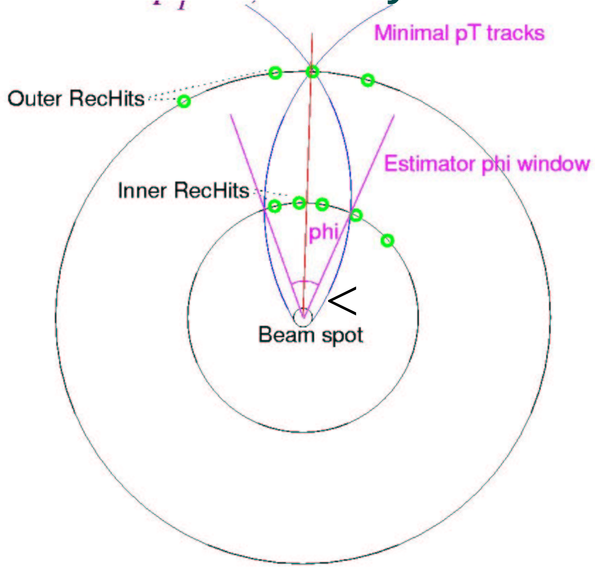
Black: Total number of hits
Green: double-sided hits
Red: double-sided hits in thin detectors
Blue: double sided hits in thick detectors.

Track Reconstruction

- The **Combinatorial Kalman Filter** is the main algorithm used to reconstruct charged tracks
  - Local method: one track reconstructed at a time, starting from an initial trajectory.
  - Recursive procedure: track parameters estimated from a set of reconstructed hits
  - Takes into account the energy loss and multiple-scattering between layers
  - Integrates pattern recognition and track fitting
- The Kalman Filter is mathematically equivalent to **a global least square minimization** (LSM), optimal when
  - model is linear
  - random noise Gaussian
- For **non-linear models or non-Gaussian noise => Adaptive Filters** . The Kalman Filter is still the optimal linear estimator.

Track Seeding

- ***Inside-out tracking:*** start in the first Pixel layers, grow tracks layer by layer to the outer layer of the SST.



Initial trajectories (*seeds*) made of **pixel hit pairs:** every combination of 2 pixel layers, compatible with the beam spot and a minimum p_T cut.

  - The Pixel detector ensures: 3 dimensional measurement points with good spatial resolution. The closest to the interaction point => minimal multiple scattering and low occupancy.

- ***Outside-in tracking:***

  - Muons Reconstruction: seeds in the outer layers based on Muon- Chamber seeds.
  - Electrons from γ conversion: seeds in the outer layers based on ECAL clusters.

Combinatorial Kalman Filter
The Method

1) **Trajectory Building:** construction of trajectories for a given seed

- Trajectories are extrapolated from layer to next layer, accounting for multiple scattering and energy loss
- On the new layer, new trajectories are constructed, with updated parameters (and errors) for each compatible hit in the layer.
- All trajectories are propagated to the next layer in parallel to avoid bias.
- The number of trajectories to grow is limited according to their χ^2 and the number of missing hits.
- Only the estimate in the last layer is based on the full track candidate information

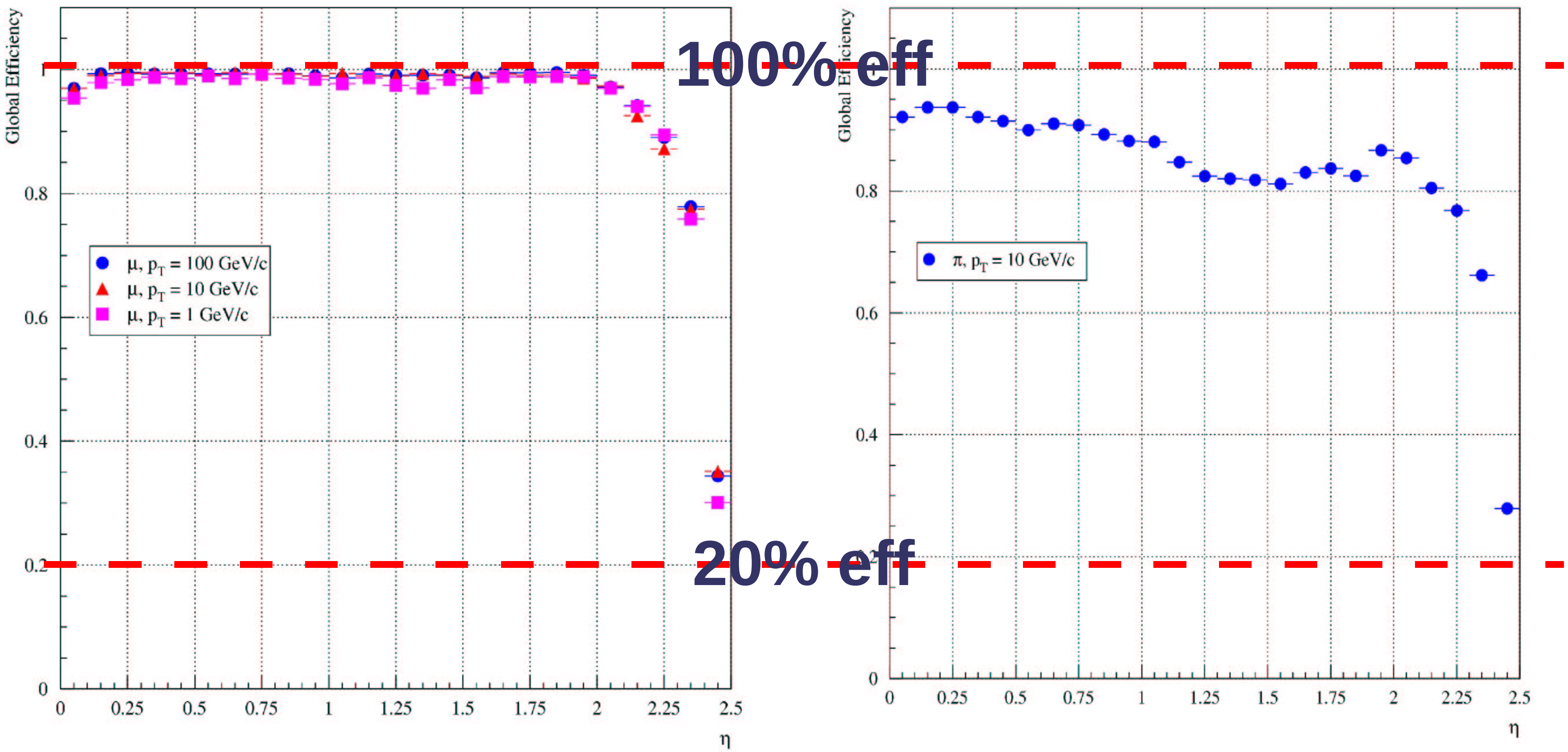
2) **Trajectory Cleaning:** hit assignment ambiguity resolution

3) **Trajectory Smoothing:** final fit of trajectories

- Obtain optimal estimates at every measurement point along the track.
- In addition to providing tracks accurate at both ends this procedure provides more accurate rejection of outliers

Combinatorial Kalman Filter Track Reconstruction Efficiency

Muons p_T 1-100GeV/c **Pions p_T 1-100GeV/c**

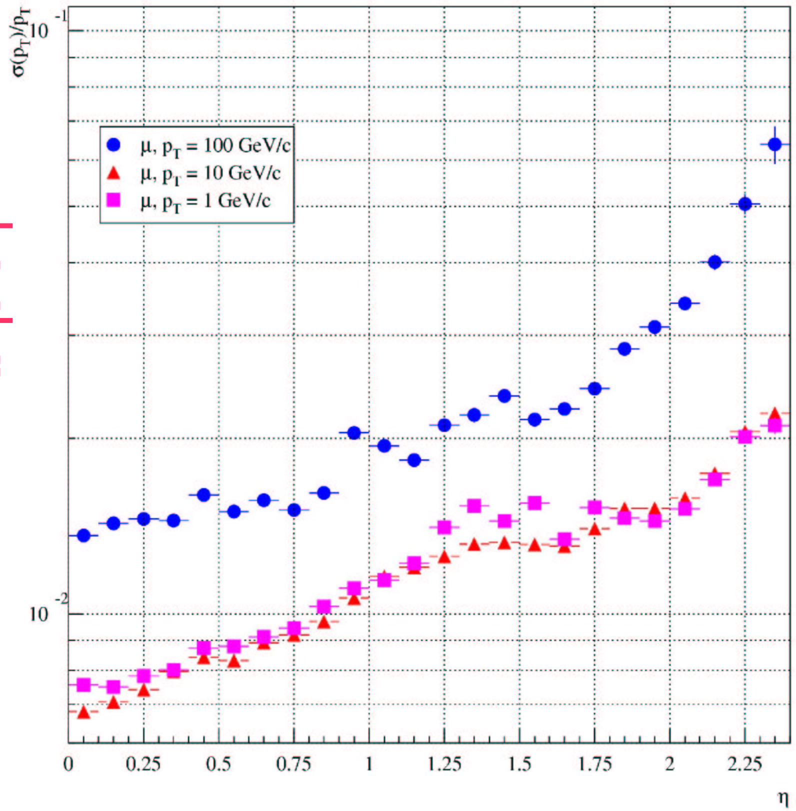


Lower reconstruction efficiency for Pions due to nuclear interactions inside the tracker (~20% of 1 GeV pions do not reach the outer layer).

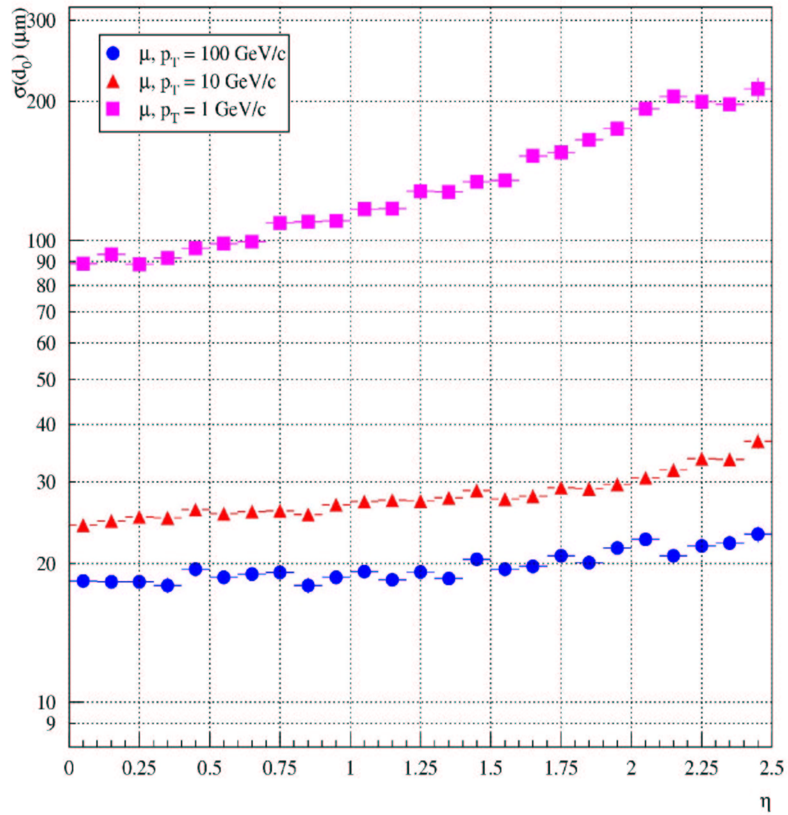
Combinatorial Kalman Filter Track Parameter Resolutions

Muons p_T 1-100GeV/c

$\sigma(p_T)/p_T$



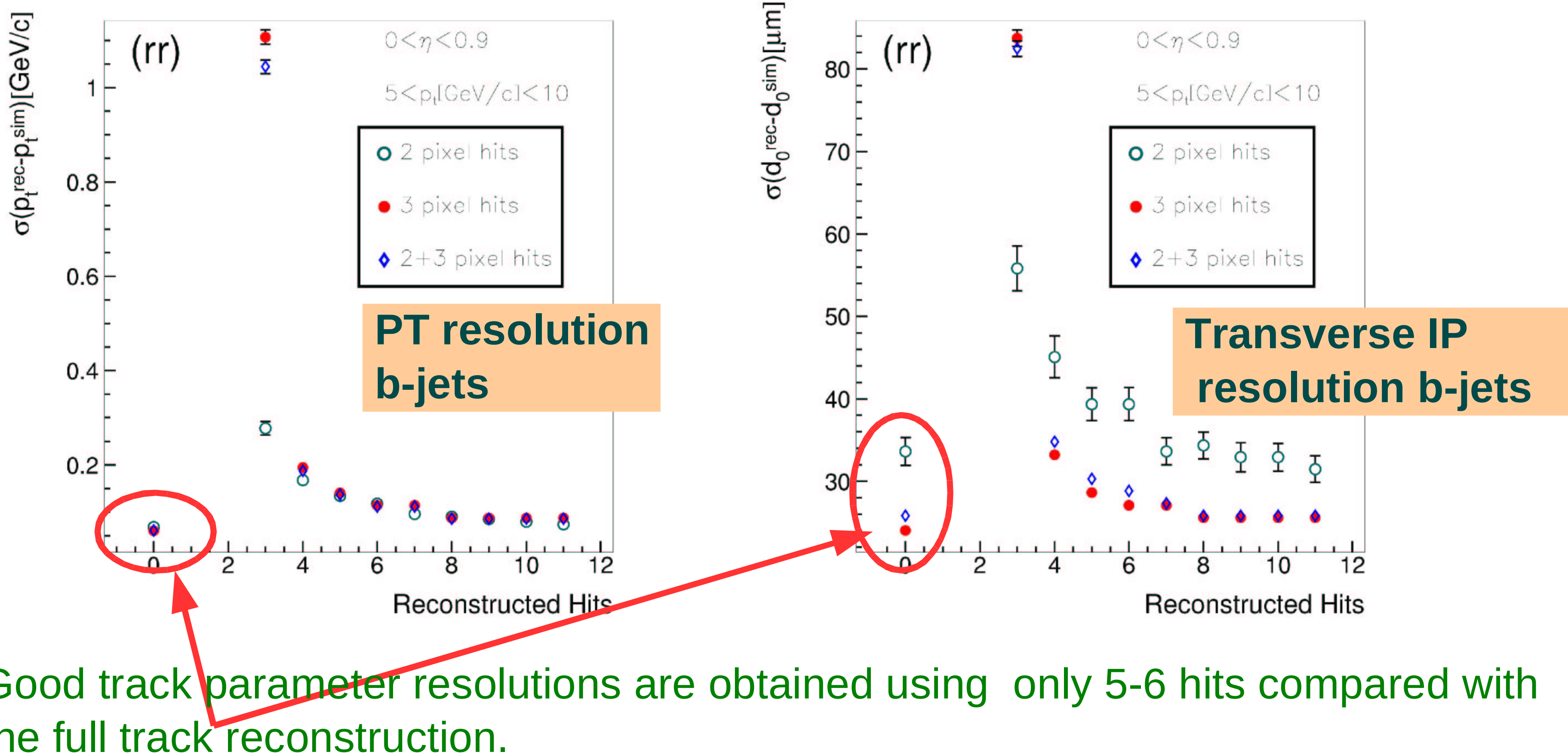
$\sigma(d_0)$



Resolutions mainly dominated by the level arm (p_T) and the spatial resolution in pixel measurements (d_0).

Combinatorial Kalman Filter
Partial Reconstruction

At the **High-Level Trigger** stage the same trajectory building and smoothing are used to reconstruct tracks with a partial information from the Tracker system
=> limit the number of hits

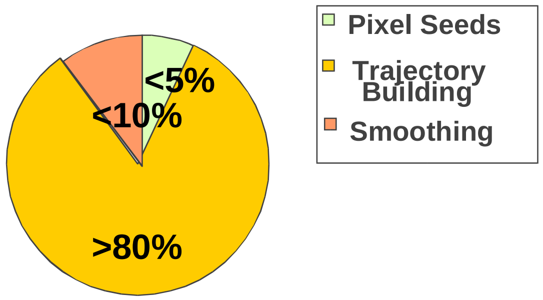


Good track parameter resolutions are obtained using only 5-6 hits compared with the full track reconstruction.

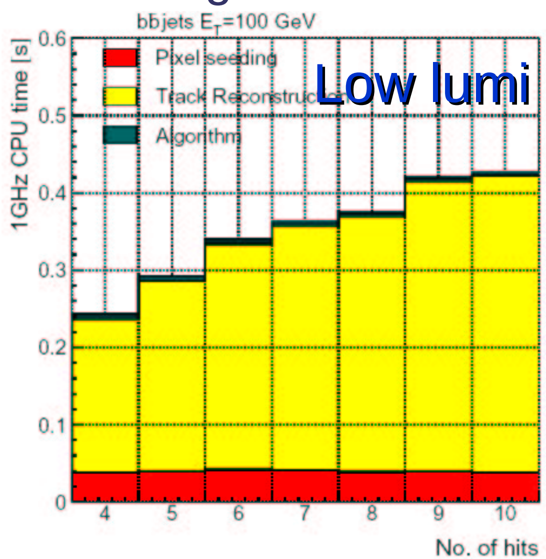
Partial Reconstruction
An Application

Timing:

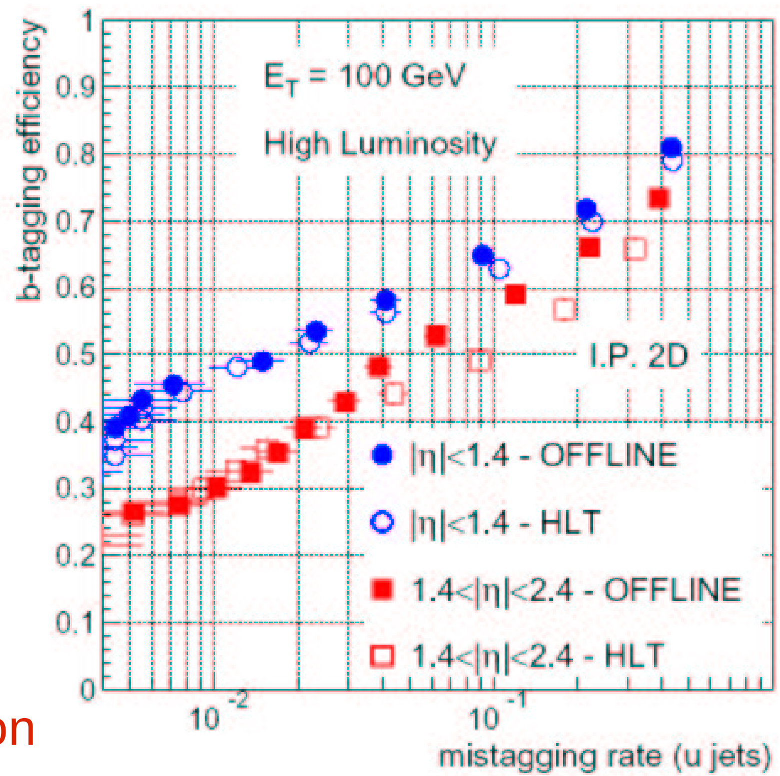
- ~85% of the track reconstruction time for trajectory building dominated by the search of compatible detectors.



- Limiting the number of hits to 5 increases speed by a factor ~1.4



Btagging performance at HLT comparable with offline reconstruction



Adaptive Filters

Least square methods are optimal when

- The model is linear
- Random noise is Gaussian (measurement errors, process noise)

To better describe non-linear models (specific applications), adaptive filters have been implemented.

- The Gaussian Sum Filter (GSF) suitable when:
  - ➔ Measurement errors have tails
  - ➔ non-Gaussian distribution of energy loss and bremsstrahlung
- The Deterministic Annealing Filter (DAF) and Multi-Track Fitting (MTF) suitable when:
  - ➔ Large background noise (electronics, low pT tracks, etc) produces hit degradation => error on the hit assignment
  - ➔ Not treated here

The Gaussian-Sum Filter

Motivation: *Pdfs* involved are usually non-Gaussian:

- Measurement errors have Gaussian core with tails
- Energy loss and multiple scattering (tails)

Method: the Gaussian-sum Filter (GSF): instead of single Gaussian,

model the pdf involved by mixture of Gaussians:

- Main component of the mixture describes the core of the distribution
- Tails are described by one or several additional Gaussians.

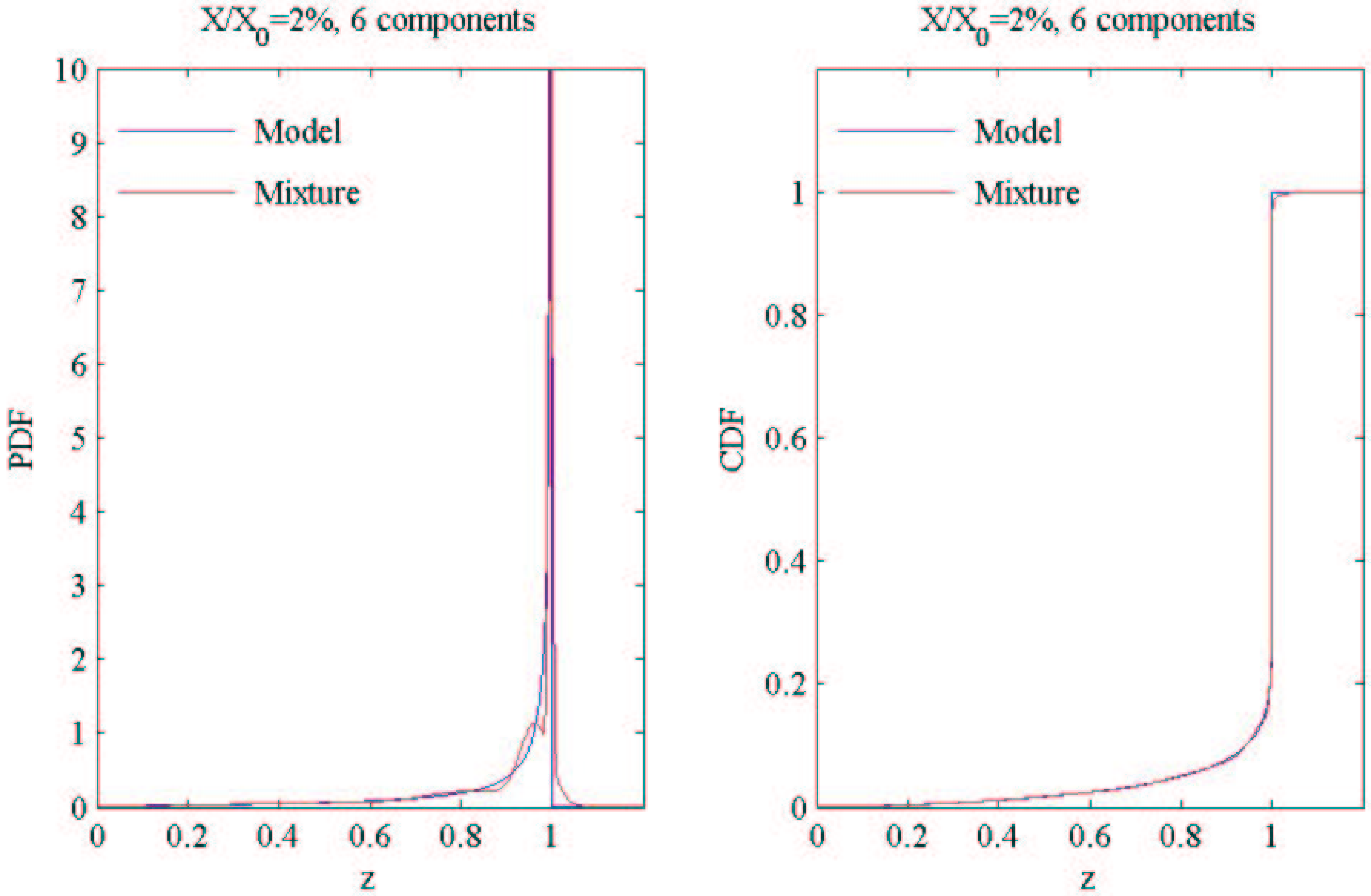
Application: electrons are good candidates to be reconstructed with the GSF

- Energy loss are dominated by bremsstrahlung
- Bethe and Heitler energy loss model is highly non-Gaussian (in the standard KF, distribution approximated by single Gaussian)

➔ The Bethe-Heitler distribution is modeled by a mixture of Gaussians

The Gaussian-Sum Filter

➔ The Bethe-Heitler distribution is modeled by a mixture of Gaussians



[R. Frühwirt 2003 *Comput. Phys. Commun.* **154** 131]

The Gaussian-Sum Filter

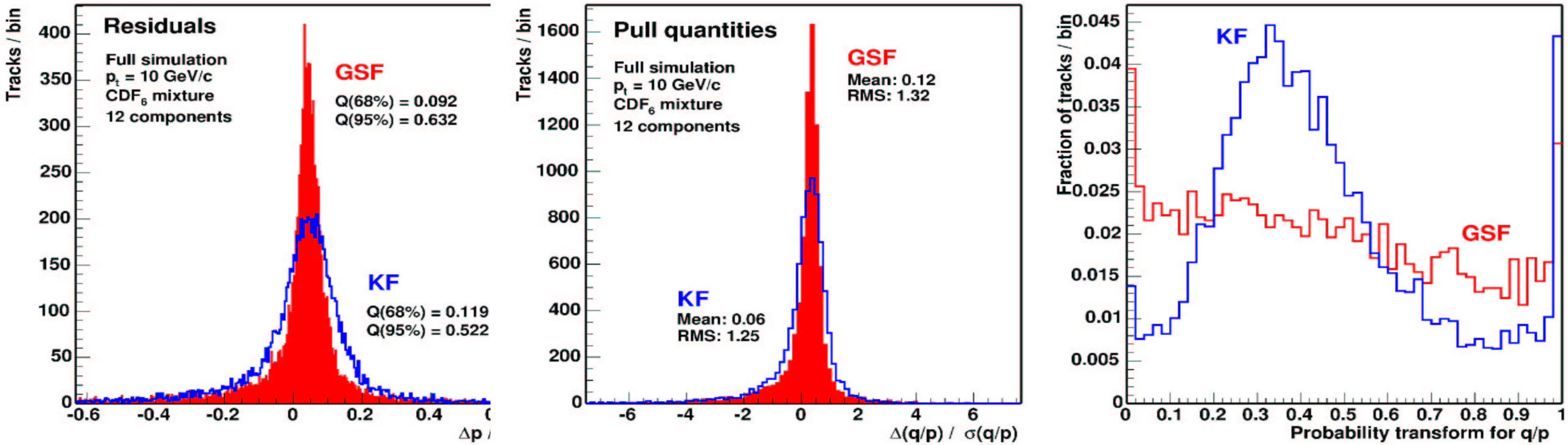
- All involved distributions are Gaussian mixtures
- State vector is also distributed according to a mixture of Gaussians

The GSF is a non-linear generalization of the Kalman Filter => Weighted sum of several Kalman Filters

  - GSF is implemented as a number of Kalman filters run in parallel
  - The weights of the components are calculated separately

- Limiting number of components: if the state vector has *n* components and the measurement density (energy loss) has *m* components, the updated state vector density has *mxn* components (exponential explosion)
  - Number of components have to be limited to a predefined number at each step => Cluster (collapse) components with the smallest 'distance' (according to a distance definition: Kullback-Leibler Distance or Mahalanobis Distance)
- Output is full Gaussian mixture of state vector => Can be used in subsequent application (=> GSF vertex fit)

The Gaussian-Sum Filter



➔ Significant improvement of the core of residuals

➔ Tails only slightly reduced:

_ GSF is sensitive to outliers (increase the weight of components far from the true values) => it could be solved modeling the meausured positions by a Gaussian mixture (meauserement errors of Pixel hits are non-Gaussian.

_ Radiation in the innermost layer cannot be evaluated => can be compensate by a vertex constraint.

➔ Most significant improvement wrt KF for low energy electrons (~10GeV), little gain at 100GeV. Perfect Pattern recognition is assumed.

Vertex Reconstruction

Reconstructed tracks are the inputs of vertex reconstruction
Performance directly depends on the of the track reconstruction

- A first measurement of the signal PV can be reached before the full track reconstruction using the **only hits in the Pixel Detector** (Pixel Primary Vertex Finder)

- Vertex reconstruction with **full reconstructed tracks** typically involves the following steps:

  - Vertex finding: given a set of tracks, individuate clusters of tracks compatible with the same vertex => produce vertex candidates

  - Vertex fitting: given a set of tracks, compute the most compatible vertex position and use it to constrain track parameters at vertex and covariance matrices

Vertex Finding Algorithms

Vertex finding algorithms can be classified in:

- *Agglomerative* algorithms:
  - At the first iteration, vertex candidates consists of only one track
  - Iteratively merge vertex candidates until the stopping condition is satisfied
- *Divisive* algorithms:
  - At the first iteration, only one vertex candidate made of the whole set of tracks
  - Iteratively split into incompatible candidates until the stopping condition is satisfied

Number of vertex candidates during iterations: $\begin{cases} \text{Agglomerative} & \Downarrow \\ \text{Divisive} & \Uparrow \end{cases}$

Pixel Primary Vertex Finding

Find sets of three hits compatible with a track ⇨ Evaluate Track Parameter ⇨ Search for z-component of Primary Vertex

- No Kalman Filter is used.
- Simple parameterization to evaluate track parameters
- First 1D measurement of the PV longitudinal coordinate (available before full track reconstruction)
- It can be used regionally

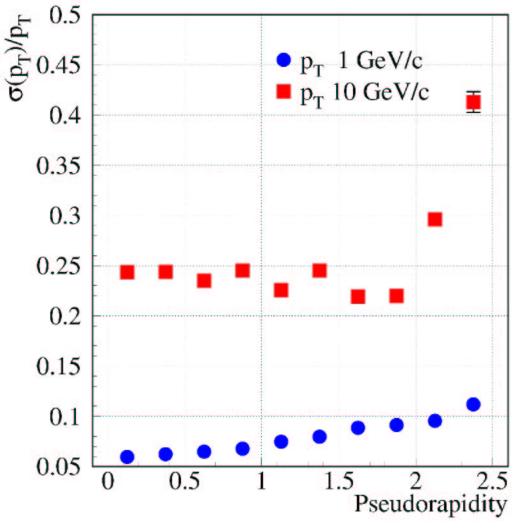
Main applications:

➔ Constraint the track seeding to the signal primary vertex

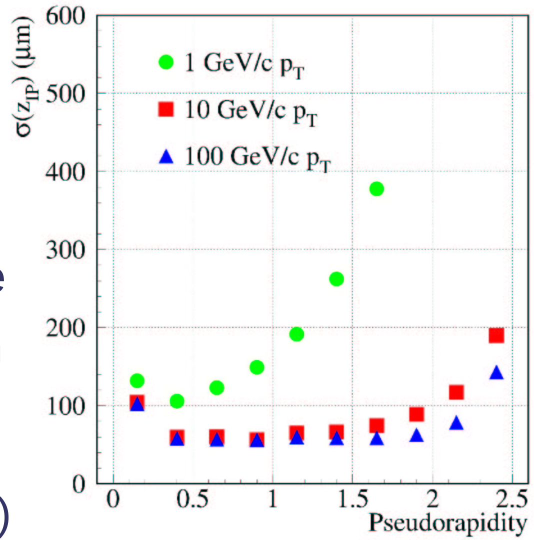
➔ Standalone pixel reconstruction used in many HLT applications

Pixel Primary Vertex Finding

Transverse momentum and impact parameter resolutions for hit triplets



Degradation of high p_T measurement due to the short level arm of the Pixel Detector. (Full track reconstruction~2-3%)



~60 μm of longitudinal impact parameter resolution for p_T = 1-10 GeV/c in the barrel. (Full track reconstruction ~40μm)

- Two algorithms of primary vertex finding:

An *agglomerative method* based on *histogramming*: clusterize tracks close on the longitudinal impact parameter.

A *divisive method*: iteratively discard tracks not compatible with the vertex estimate and recover discarded tracks to make a new vertex

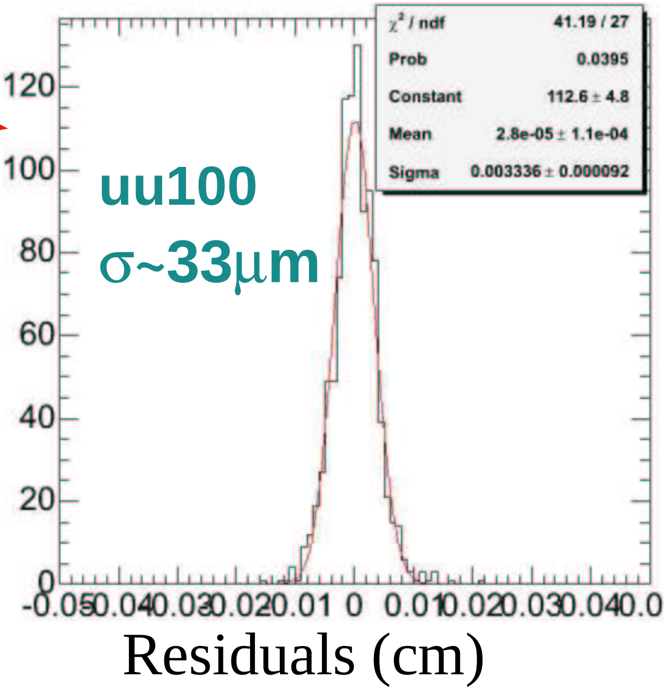
Pixel Primary Vertex Finding

PV finding efficiencies wrt to a 500μm windows around the simulated PV

eff(%)/event	Uu100	bb100	jet50-100	$b_0 J\psi$	h->γγ	h->eeμμ	tth->bb
Histo-PVtag	98	96	90	61	75	96	99
Divi-PVtag	99	99	94	78	80	100	100

Low Luminosity Results

- The Divisive method shows general better performance
- The signal PV is found and identified with an efficiency close to 100% in many physics cases
- Final states with low track multiplicity need "dedicated" algorithms



Primary Vertex Finding

Find primary vertices and identify the one from the trigger event.

Inputs are **fully reconstructed tracks** either in the whole Tracker or inside a region of interest (defined around a jet, a muon,...)

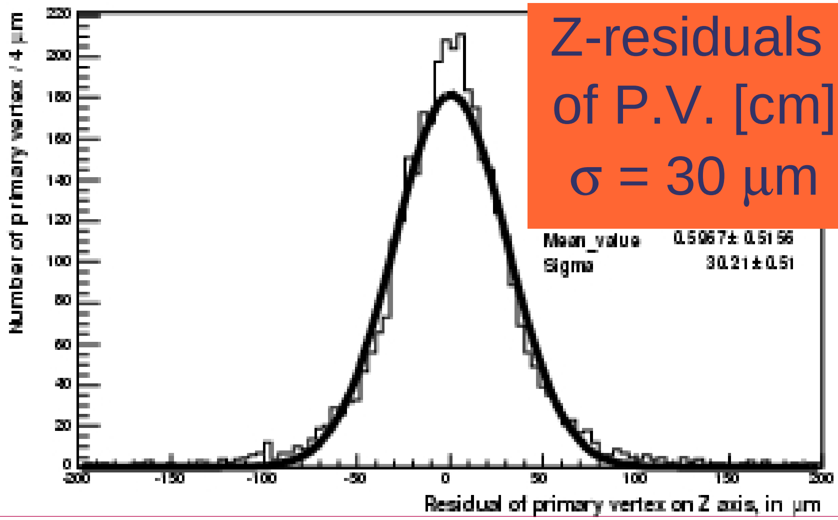
Principal Vertex Finder

Divisive algorithm: - reject tracks with less than 5% compatibility to the vertex
- vertex search among the discarded tracks

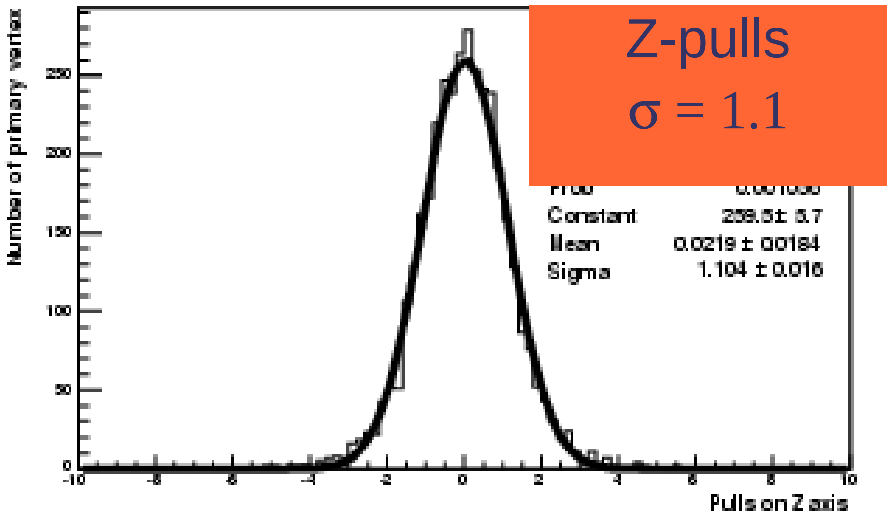
Efficiency to find P.V. Inside the beam spot with track purity>50%

bb jets
Et = 100 GeV, $|\eta| < 1.4$

Global search: 95% no PileUp – 92% Low Lumi.
Regional search inside a b-jet cone: 80%



Z-residuals of P.V. [cm]
σ = 30 µm



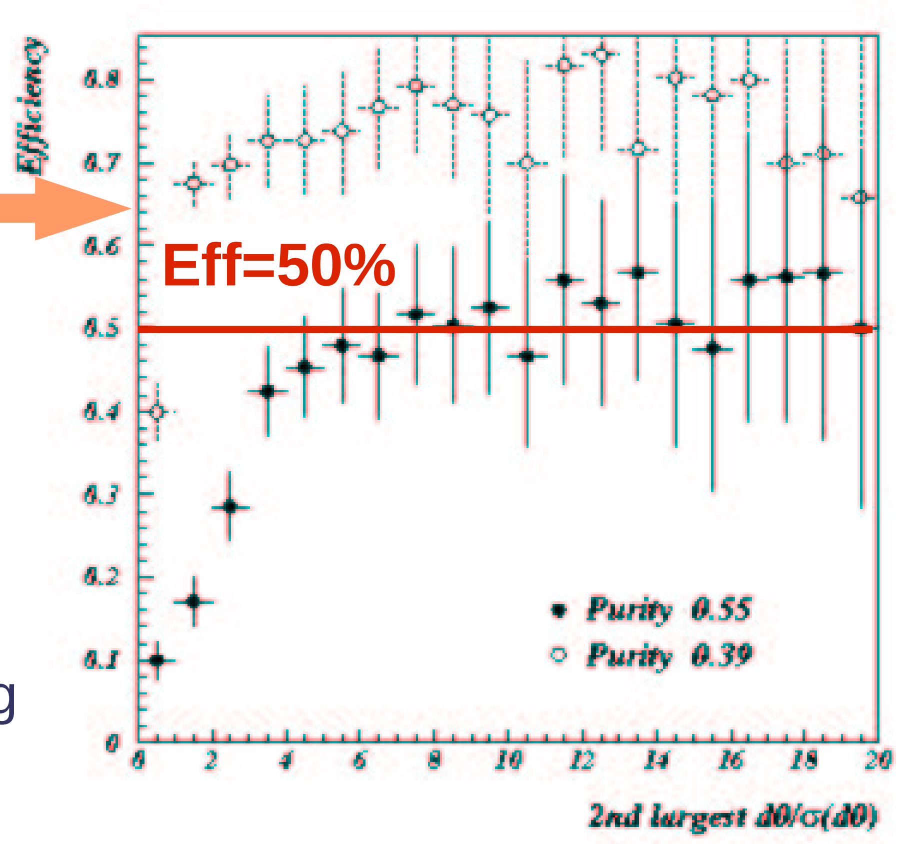
Z-pulls
σ = 1.1

Secondary Vertex Finding

Principal Vertex Finder

Find secondary vertices in a jet and optimize assignment of tracks to primary and secondary vertices

Efficiency to find SV inside a b-jet vs. the 2nd largest impact parameter significance, for different purity requirements



With a purity above 50%, the efficiency to find a SV in a b-jet is 48%
=> that corresponds to a b-tagging efficiency of ~ 50% with a mistagging rate of ~1% from light jets

Many others algorithms for SV finding implemented in CMS and under study!

Vertex Fitting

Most precise estimation of the 3D vertex position and
track parameters at vertex from a set of tracks

Discussed here:

- Kalman Vertex Fitter
  - A Least Square (LS) method, refit of the track with the vertex constraint
  - Sensitive to outliers and non-Gaussian tails in the track parameter errors
- LS Robust Fitters

  "robustifying" LS methods with respect to outliers:
  - Trimming Vertex Fitter
  - Adaptive Vertex Fitter
- Gaussian-Sum Vertex Fitter

  "Robustness" with respect to non-Gaussian tails of track errors

Kalman Vertex Fitter

Minimize as a function of the track reduced distance $\chi_i = (x-x_i)/\sigma_i$

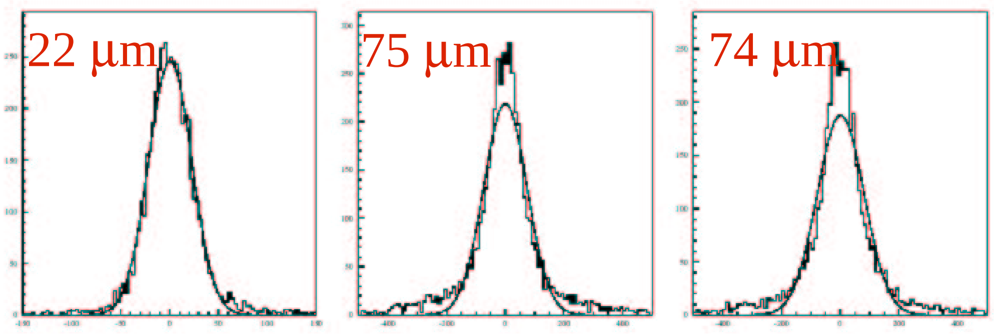
Vertex position residuals and pulls in z

$H \to ZZ \to 4\mu$ $B_s \to J/\psi\ \varphi$ $B_s \to \mu^+\mu^-$

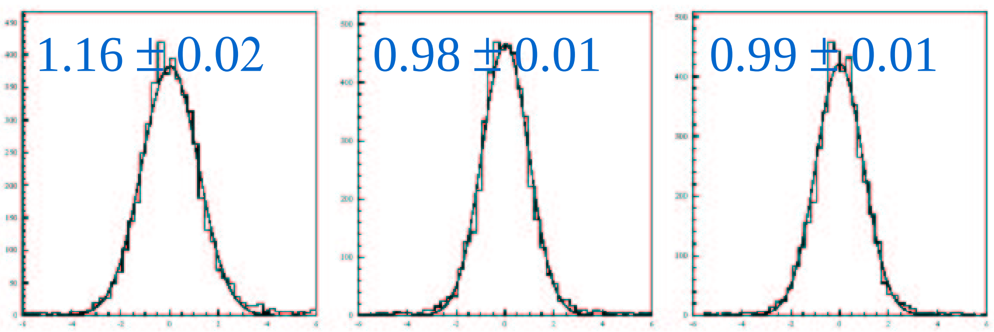
Residuals of f of tracks at vertex

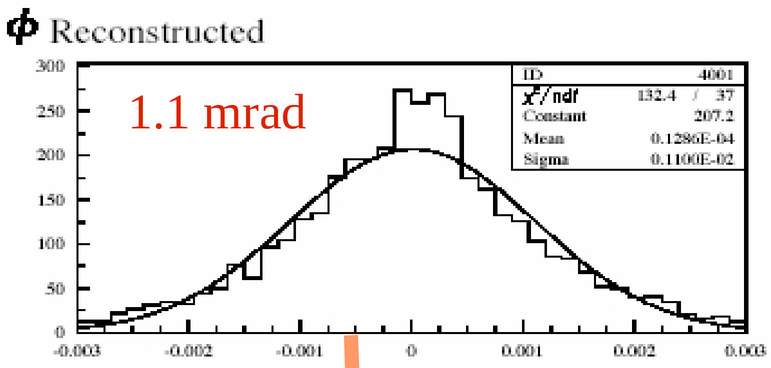
$B_s \to J/\psi\ \varphi \to \mu^+\mu^- K^+K^-$

z-residuals

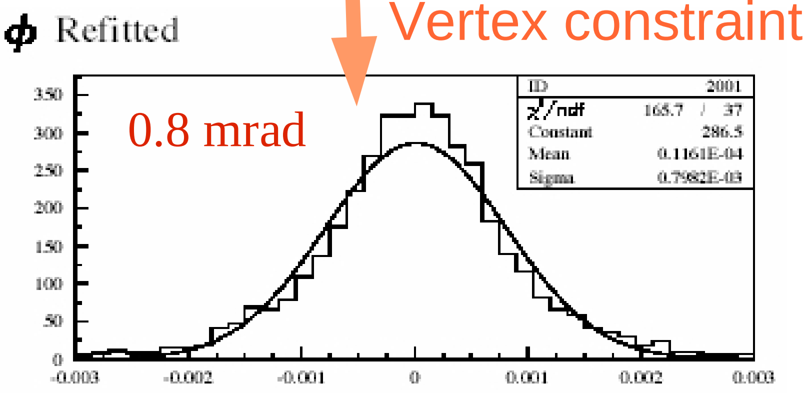


z-pulls





Vertex constraint



Robust LS Vertex Fitters

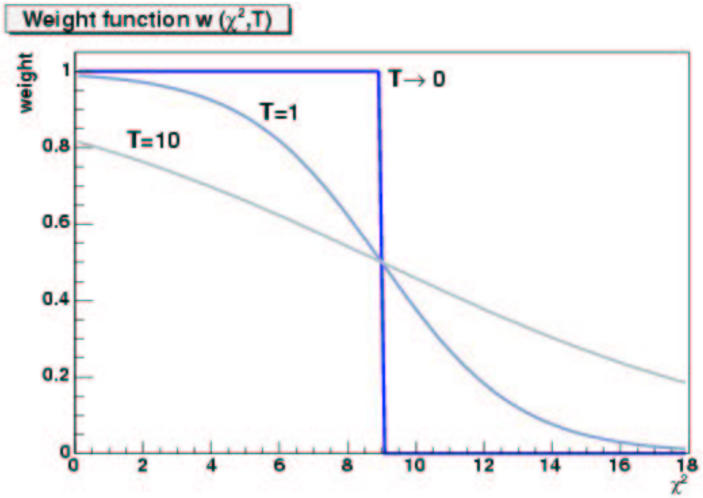
Least Trimmed sum of Square (LTS):

- Discards *m* out of *n* tracks which are the least compatible with the vertex
- The trimming fraction *m/n* is an input parameter (typically 20%)

Adaptive Vertex Fitter:

$$w_i = \frac{1}{1+\exp\left(\frac{\chi^2-\chi_c^2}{2T}\right)}$$

- Re-weighted LS fit with soft assignment
- Down-weights the reduced distance of the track *i* from the vertex estimate at *i-1* iteration by the weight function w_i
- Input parameters:
  - χ_c distance where the weight function drops to 0.5 (typically 4)
  - *T* controls the sharpness of the drop



Both the algorithms are iterative: they need an initial estimate and iterate until the vertex position converges

Comparison of the Fitters

qq jets Et=100GeV, |η|<1.4

	x-Res (μm)	x-Pulls	z-Res (μm)	z-Pulls
Linear Fitter	39	2.1	39	1.9
Trimming Fitter	25	1.1	29	1.1
Adaptive Fitter	21	1.1	28	1.1

cc jets Et=100GeV, |η|<1.4

	x-Res (μm)	x-Pulls	z-Res (μm)	z-Pulls
Linear Fitter	197	6.4	167	4.3
Trimming Fitter	21	1.4	30	1.3
Adaptive Fitter	18	1.2	23	1.3

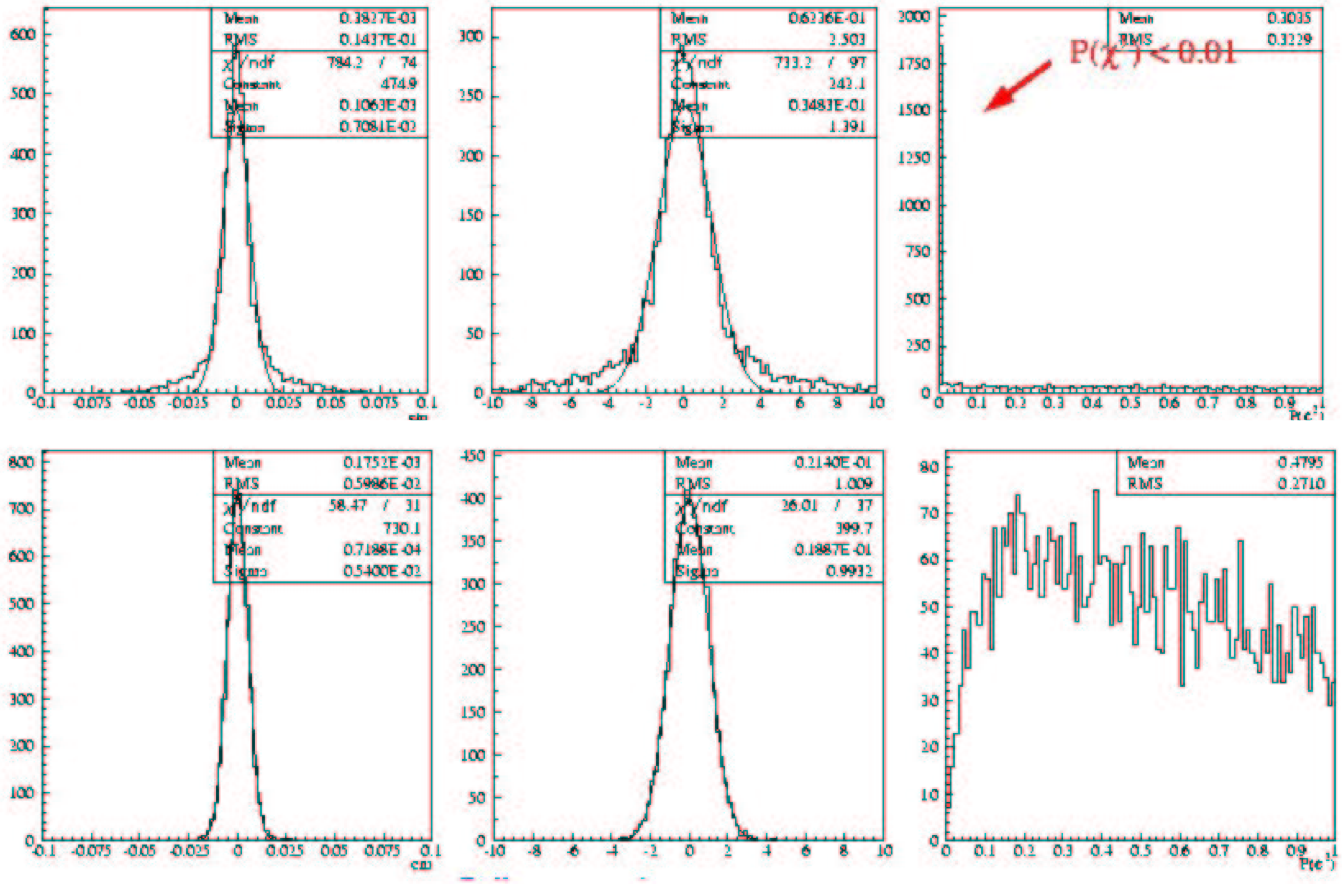
➔ Robust fitters show overall better performance on both resolution and errors.

➔ Most significant improvement in events with more outliers (c-jets), where the spatial resolution of the Linear Fitter is comparable with the distance between primary and secondary vertices in the jet.

Gaussian-Sum Vertex Fitter

- ➔ Track parameter error distribution modeled by a mixture of Gaussians
- ➔ Iterative: the estimate of the vertex is updated with one track at the time
- ➔ **Simplified simulation**: track parameters smeared with 2 Gaussian mixture

Residuals **Pulls** **P(χ^2)**



Kalman Filter:
Non-Gaussian tails
P(χ2) distribution peaks below 0.01

GS Filter:
Better resolution
Smaller tails

Gaussian-Sum Vertex Fitter

The Gaussian Sum Vertex Filter can be combined with the Adaptive Vertex Filter

=> Adaptive-GSF: adaptive vertex filter which uses the GSF as updator

Able to both down-weight outliers and use the full mixture of tracks

Robustness tests have been performed with a simplified simulation

Vertices without outliers

Filter	mean $P(\chi^2)$	Res (μm)	Pull
Kalman	0.32	71	1.39
GSF	0.48	54	0.99
Adaptive	0.3	59	1.08
A-GSF	0.44	54	0.93

Vertices with 1 outlier (mismeausured track)

Filter	mean $P(\chi^2)$	Res (μm)	Pull
Kalman	0.18	115	1.61
GSF	0.37	83	1.11
Adaptive	0.18	92	1.34
A-GSF	0.24	84	1

Conclusions

- CMS has a very robust and versatile tracker and track reconstruction algorithms, able to operate in a very challenging environment
- Combinatorial Kalman filter shown to give very good results even in difficult environments:
  - high efficiency, low fake rate
  - Good track parameter resolutions after using only the first five to six hits => usable at HLT stage
- More sophisticated methods available for specific applications (e.g. GSF): adaptive algorithms show improvements w.r.t. LSM in difficult situations
- Vertex reconstruction: several algorithms have been presented providing both a good efficiency on identifying vertex candidates and high precision on evaluating the best estimate of the position
- Adaptive algorithms are available and they are shown to be more stable w.r.t. Outliers and non-Gaussian tails of track errors (robustification of LSM, Gaussian Sum Vertex Fitter)