Track reconstruction with the CMS tracking detector

B. Mangano (University of California, San Diego)

&

O.Gutsche (Fermi National Accelerator Laboratory)

Overview

The challenges

The detector

Track reconstruction

- algorithms for general purpose tracking
- implementation of the tracking code in the "new" SW framework of CMS
- advanced algorithms and special applications

Conclusions & Outlook

The challenges

pp-collisions at design luminosity ($10^{34}cm^{-2}s^{-2}$, 14TeV)

- 40 MHz crossing rate
- O(20) superimposed pileup (PU) events / crossing
- O(2000) charged tracks / crossing

Charged track density

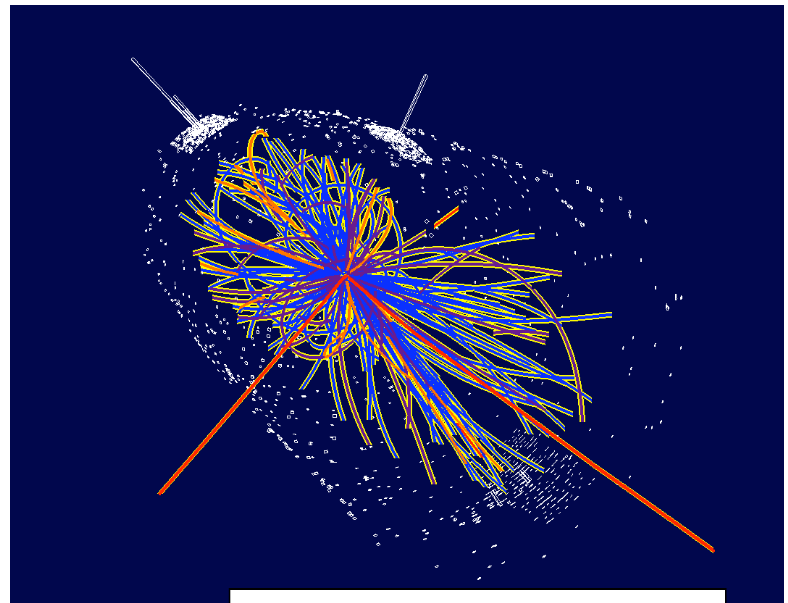
- 2.5 / cm^2 / 25ns at r = 4cm
- 0.15 / cm^2 / 25ns at r = 10cm

Material budget

- high granularity is obtained by means of a “heavy” tracker (0.2-1.4 X_0)

Trigger

- Level 1:
  Design rate 100kHz, no tracker
- Levels 2-3 (HLT):
  Reduction to ~200Hz
  Includes (partial) track reconstruction



Higgs -> ee μμ event with Low Luminosity PU

The challenges

Physics requirements

Highly efficient track reconstruction & low fake-track rate

Excellent momentum resolution

- Mass reconstruction
- Energy flow
- Charge separation

Excellent impact parameter resolution

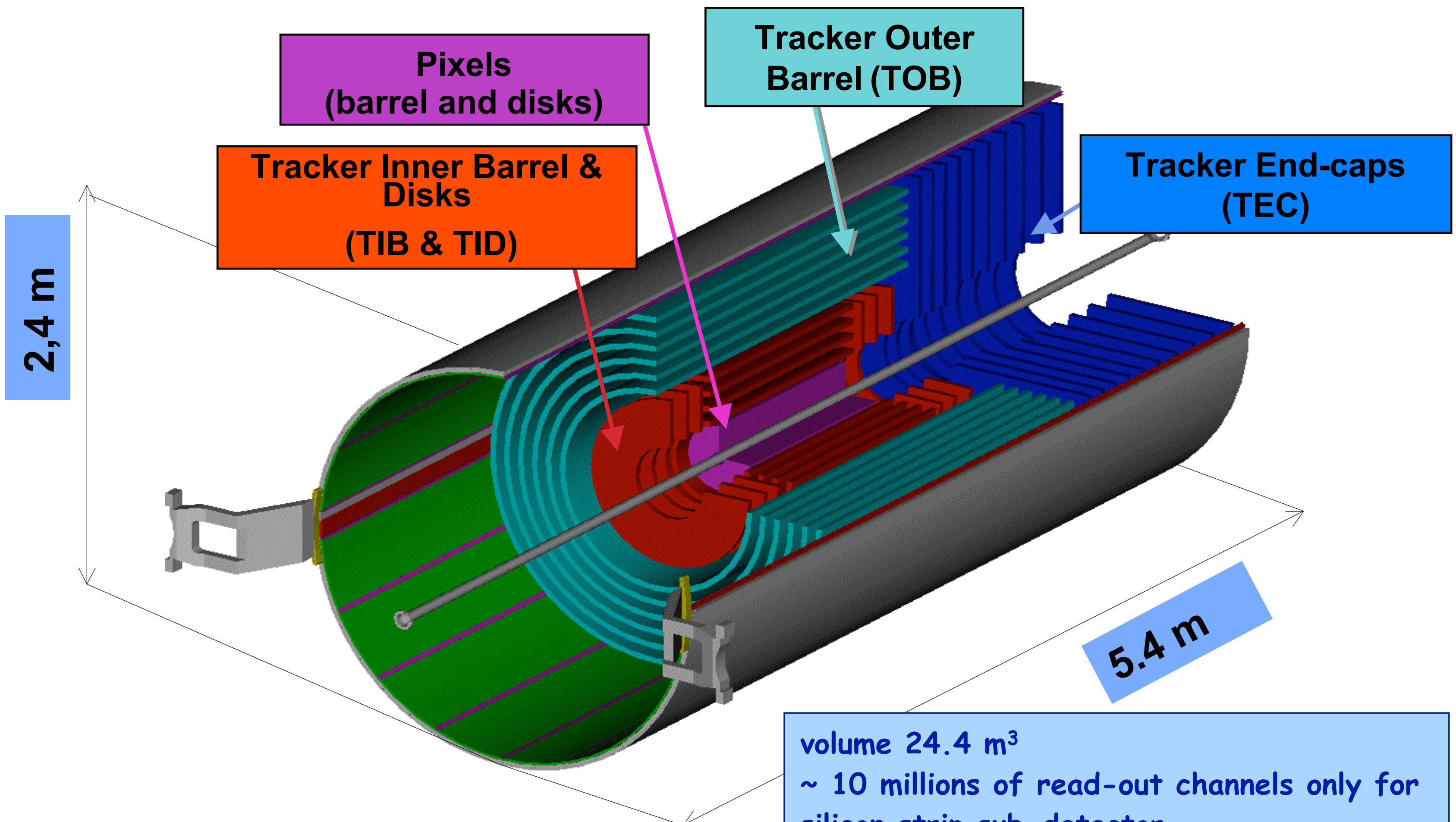
- Primary vertex reconstruction & separation of pileup vertices
- Secondary vertex reconstruction
- Heavy flavor tagging

Combined reconstruction

- Link to ECAL and Muon systems

The detector

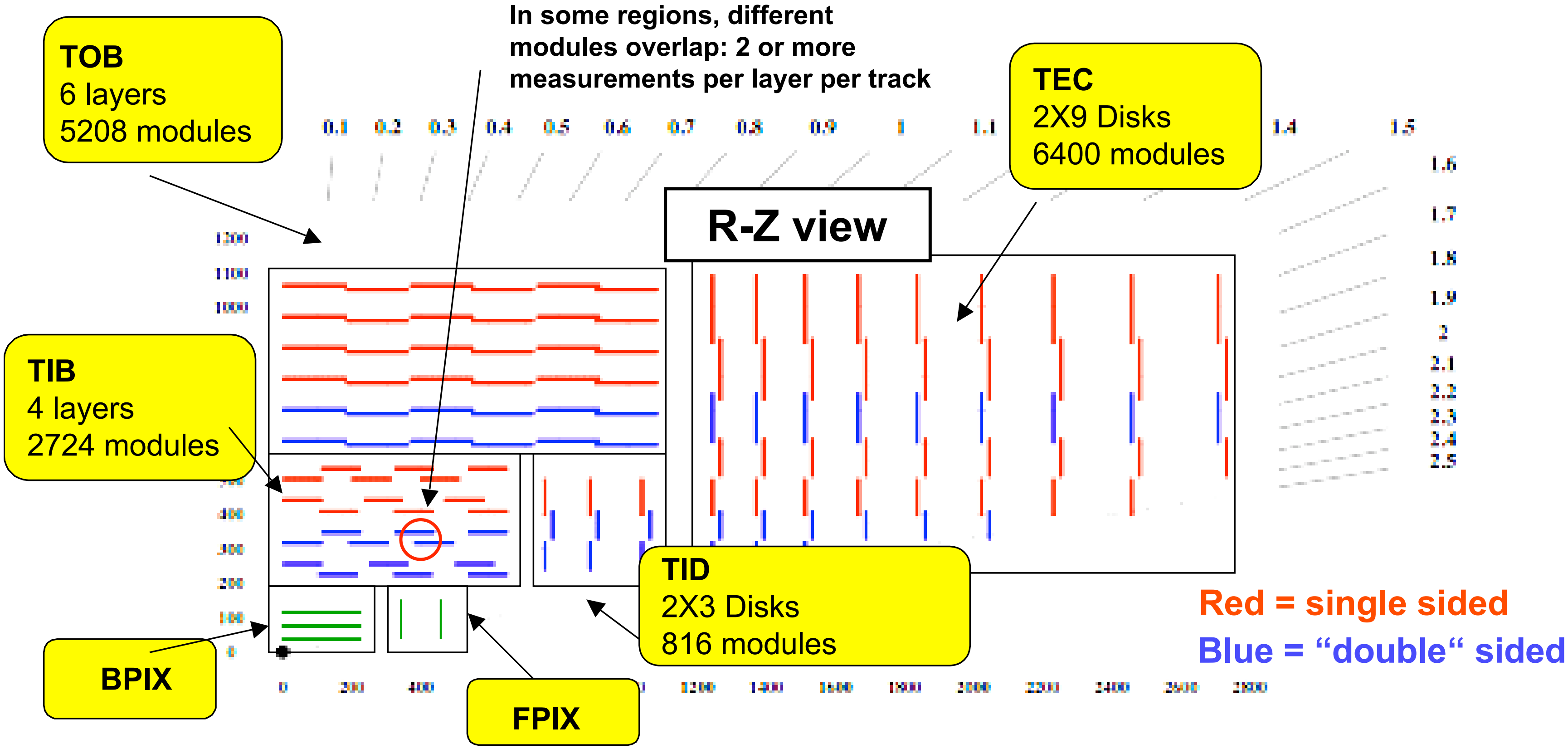
sub-structures of the full-silicon CMS tracker



volume 24.4 m^3
~ 10 millions of read-out channels only for silicon strip sub-detector

The detector

sub-structures of the full-silicon CMS tracker



Strip lengths range from ~10 cm in the inner layers to ~ 20 cm in the outer layers.

Strip pitches range from 80 μm in the inner layers to near 200 μm in the outer ones.

general purpose tracking

logical modularization

- **hit reconstruction:**

  strip and pixel signals are grouped (clustering) and finally hit positions and corresponding error matrices are evaluated.

local reconstruction

- **reconstruction of tracks:**

  1) seed finding: a fast and rough estimate of the track's parameters (+ errors) is obtained from a minimal amount of information.
  2) pattern recognition: an iterative process which, starting from the seed's parameters, collects all the hits in the tracker which are compatible with a unique track.
  3) final fit: the positions all the hits associated to the same charged particle are used to provide the best estimate of the track parameters and corresponding errors.

global reconstruction

general purpose tracking

reconstruction algorithms in CMS

Two different general purpose tracking algorithms are currently implemented:

1. **Combinatorial Track Finder (CTF) is the default one:**
   - the seeding uses innermost tracker's layers (mainly pixel in the standard configuration).
   - the pattern recognition uses a *track-following* approach: every time a new hit is associated to the track, the "partially reconstructed" trajectory's parameters are re-evaluated and the search window on the next tracker layer is narrowed. This is done according to the smaller uncertainty on the track parameters.
   - The final set of hits is fitted using a Kalman-Filter fitting/smoothing logic.
2. **Road Search (RS):**
   - the seeding is based on hits from modules on inner and outer layers of the tracker. .
   - the pattern recognition initially uses a set of pre-calculated trajectory's *roads* to collect *clouds* of hits along the seed's direction hypothesis. The final set of compatible hits is then obtained after a subsequent cleaning of the hit collection.
   - The final fit is identical to the one used by the CTF algorithm.

general purpose tracking

the implementation

During 2006, most of the efforts of the tracking group have been devoted to the porting of the tracking code from the previous software framework of CMS (COBRA/ORCA) to the current one (CMSSW):

- Each step of the reconstruction has been implemented as a distinct plug-in framework module.

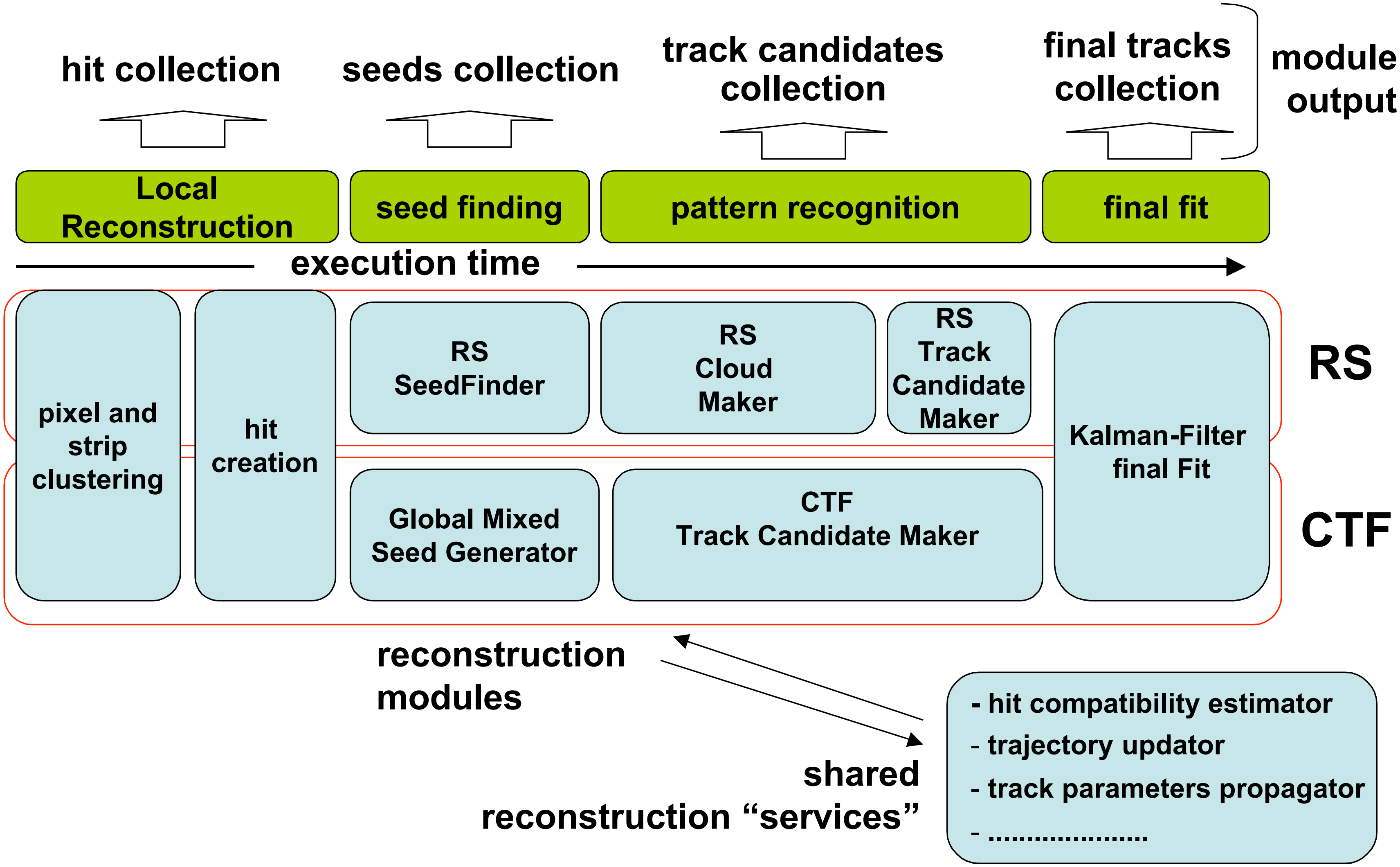
- Each module has access to currently processed data and several “services”.

- The main difference between a module and a service is that the former is allowed to produce new data: e.g. a new track-seeds collection starting from an existing hits collection. Services are *used by the modules* during the data processing.

Each of the 2 tracking algorithms is the result of a definite sequence of modules. The output of a module is the input of the subsequent one.

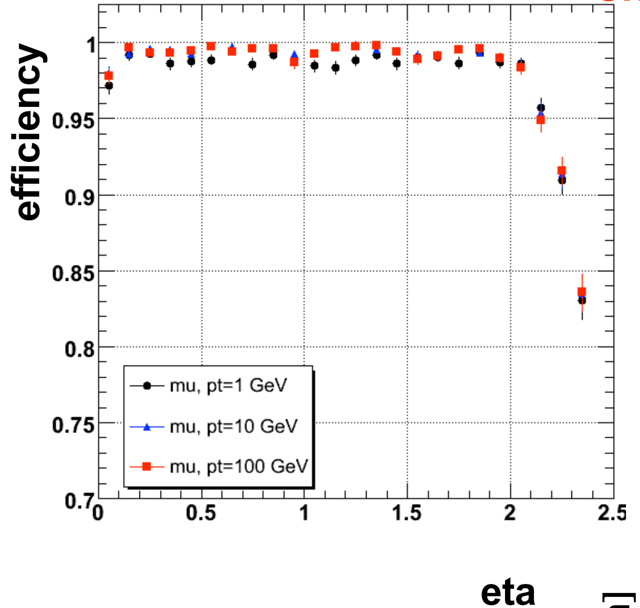
general purpose tracking

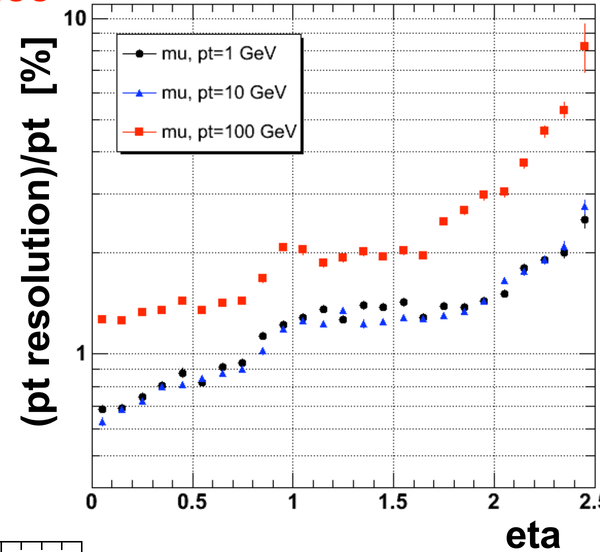
the implementation



general purpose tracking

snap-shot of performance

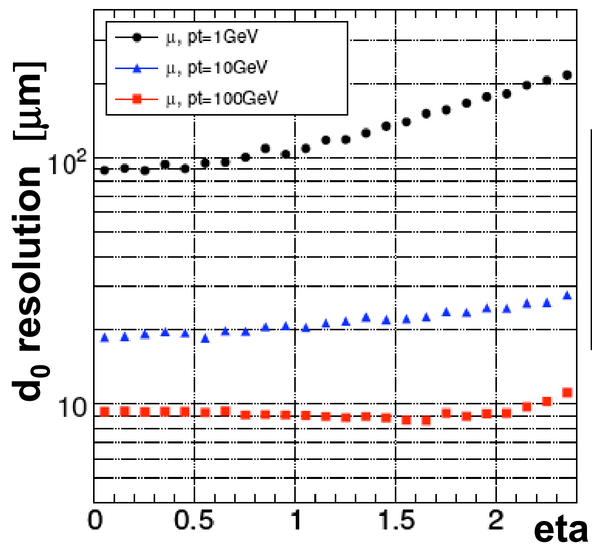




-efficiency close to 99% up to |eta| = 2.0

- pt resolution between 0.5% and 2%

- resolution on impact parameter around 10-100 microns



- ● muon, pt = 1 GeV
- △ muon, pt = 10 GeV
- ■ muon, pt = 100 Gev

Configurability and extension of the tracking code

The behavior and the performance of the track reconstruction can be adapted to different needs thanks to 3 levers:

1. implementing one or more completely **new modules** and plugging them into the reconstruction sequence.
2. **changing the services** which are used by one or more modules
3. **acting on the parameters** of each single module (or service) which define the track reconstruction sequence.

Thanks to the plug-in logic, the framework allows to play with different combination of 1), 2) and 3) changes at run-time without the necessity of recompile the code.

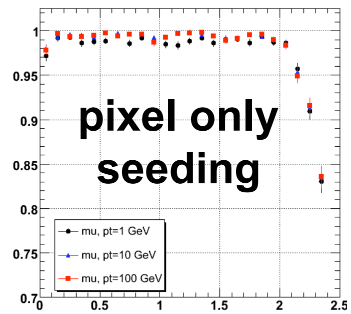
Some examples in the following slides.

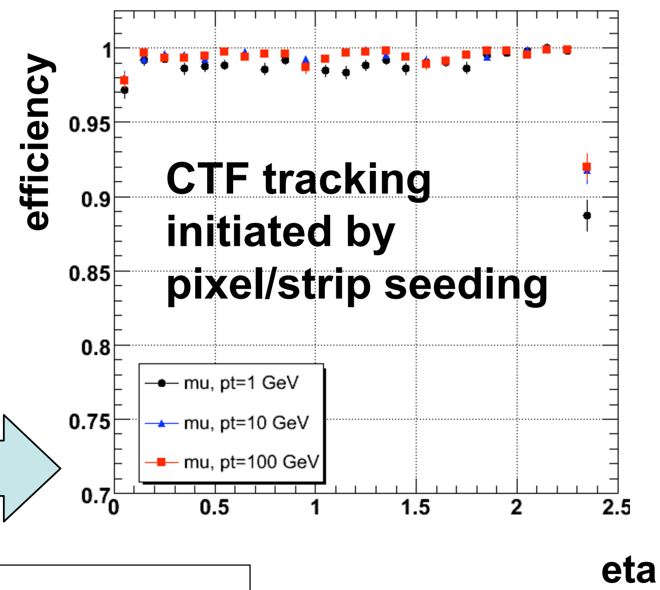
improvements plugged into the tracking code

new seed generator

Currently the default track reconstruction is initiated by a seeding which use both pixel and silicon strip hits.

The efficiency at high eta is maximized thanks to the bigger geometrical acceptance of the strip sub-detector.





snippet of the configuration files

```
..........
CtfTrackingSequence = {
  CtfPixelOnlySeeder,
  CtfTrackCandidateMaker,
  KfFitter
}
..........
```

```
CtfTrackingSequence = {
  CtfMixedSeeder,
  CtfTrackCandidateMaker,
  KfFitter
}
..........
```

A new *module* is plugged into in the track reconstruction sequence

change of type 1)

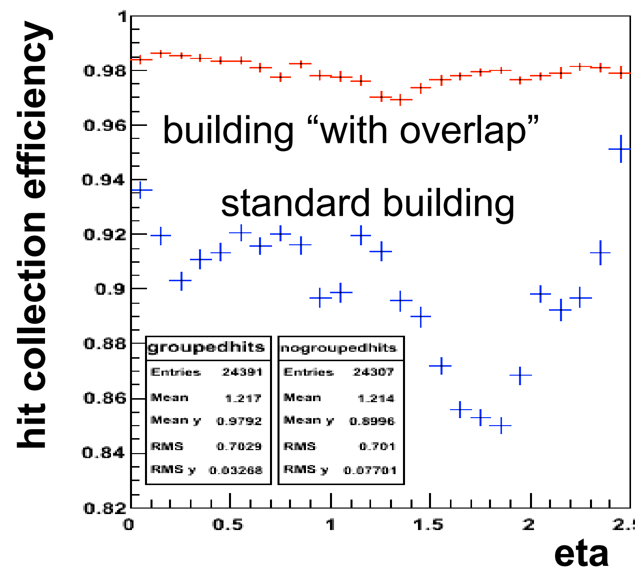
improvements plugged into the tracking code

new trajectory builder

During HLT the tracking has to be fast and **only one hit per tracker's layer** per track is usually collected.

Nevertheless, during offline reconstruction, the full available information has to be exploited. So, a special trajectory builder (*service)* has been developed to recover additional hits inside the regions of a layer where 2 or more tracker's sensors are overlapping.

The **final hit collection efficiency** is therefore **increased**.




```
..........
CtfTrackingSequence = {...,
CtfTrackCandidateMaker, ....
}
..........
```

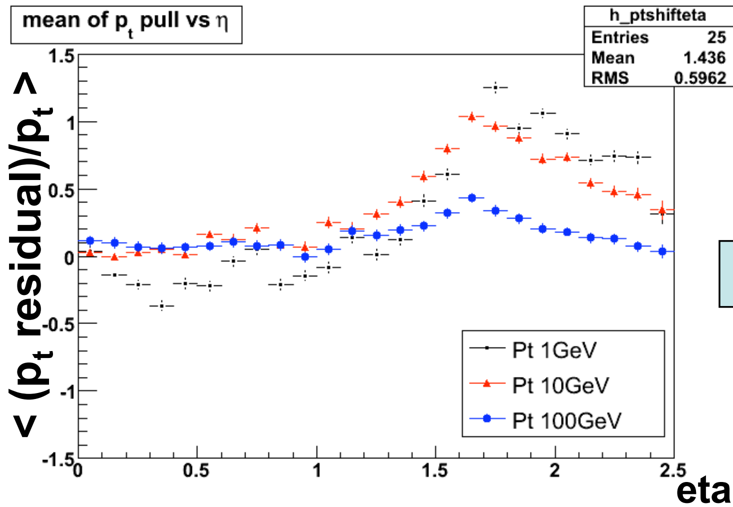
```
replace CtfTrackCandidateMaker.builder =
"OverlapModulesBuilder"

CtfTrackingSequence = {... ,
  CtfTrackCandidateMaker, ...
}
..........
```

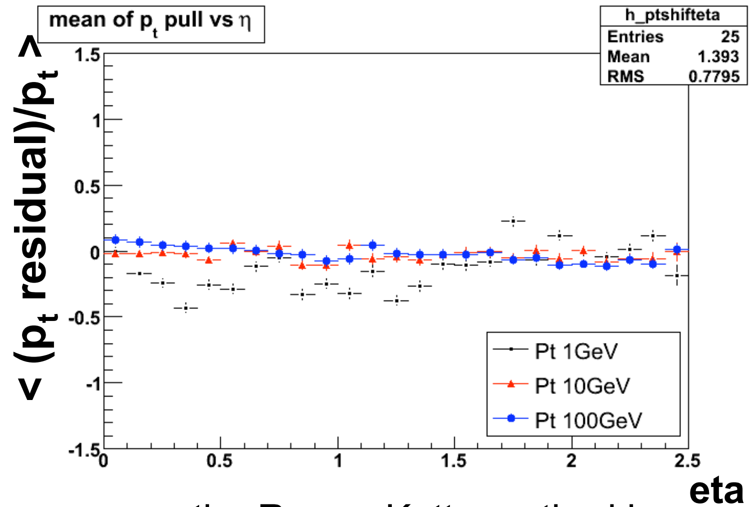
not-standard service is used by the TrackCandidate maker module

change of type 2)

improvements plugged into the tracking code

more accurate track fitter





A simple analytical propagator can be used to propagate track parameters from one tracker layer to the other ones during the KF fitting/smoothing.

Because the analytical propagator takes into account only partially the in-homogeneities of the magnetic field, there is a **bias in the estimated momentum** of the tracks.

A slightly slower, but more **accurate propagation based on Runge-Kutta method**, can be activated to fix this problem at high eta.

the Runge-Kutta method is activated changing one boolean parameter of the PropagatorWithMaterial service

change type 3)

```
..
replace
PropagatorWithMaterial.useRK = true
..........
```

special tracking and interaction with other reconstruction modules

Specific seed generators

+

(almost) the same CTF and RS tracking sequence

- **tracking without pixels:** default solution for the run without pixel data and backup solution for the physics run

- **tracking for cosmic muon:** tested on real data during integration tests of the CMS tracker (no B-Field)

see specific poster:
http://indico.cern.ch/contributionDisplay.py?contribId=264&sessionId=21&confId=3580

- **tracking for alignment** of the tracker with cosmic and beam-halo muon tracks.

CTF regional seed generator module

+

rest of the CTF tracking sequence

- **regional tracking for Trigger reconstruction**

special tracking and interaction with other reconstruction modules

Service for track parameters propagation with electron mass hypothesis

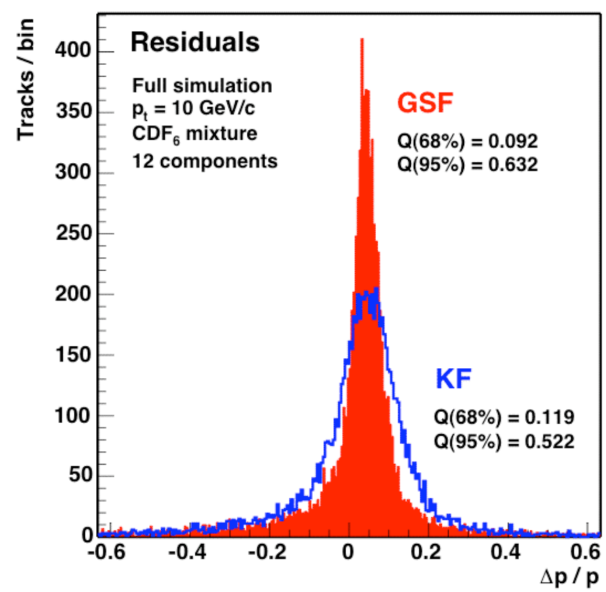
+

Final fit module based on a *Guassian Sum Filter*

(it takes properly into account bremsstrahlung and subsequent kinks in the electron's trajectory)

+ rest of the CTF tracking sequence

tracking for electron reconstruction



see specific poster:
http://indico.cern.ch/contributionDisplay.py?contribId=193&sessionId=21&confId=3580

Conclusions & Outlook

• **The original CMS tracking algorithm (CTF) and a new one have been successfully ported/implemented inside the new software framework of CMS**

• **The tracking code is highly modularized in order to:**

- facilitate the interaction with the rest of the reconstruction code
- share common resources
- minimize the duplication of code
- facilitate the debugging/maintaining of the code

• **Important improvements to the track reconstruction sequence have been plugged into it changing only specific key modules or services.**

• **The track reconstruction has been successfully tested on many (simulated) circumstances and, recently, during a (real) data-taking of cosmic muons.**

• **The tracking code appears to be sufficiently organized and flexible to cope with the challenges which CMS will have to face during the LHC data-taking**