ELSEVIER

Nuclear Instruments and Methods in Physics Research A 478 (2002) 460–464

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A

www.elsevier.com/locate/nima

Tracking in CMS: software framework and tracker performance

A. Khanov[a], M. Lenzi[b], T. Todorov[b], T. Speer[b], P. Vanlaer[c,*], M. Winkler[b]

[a] *Kansas State University, USA*
[b] *CERN, Rue de Meyrin, 1211 Geneva 23, Switzerland*
[c] *IIHE-ULB, Brussels, Belgium*

Abstract

Tracking in LHC experiments requires reconstruction software that is able to deal with high hit multiplicity and complex detector geometry. The software framework should also be flexible enough to allow online event reconstruction and selection at high trigger levels. We present an object-oriented framework that fulfills these constraints. We describe two track reconstruction algorithms currently implemented within this framework, with a comparison of their performance at high multiplicity of noise hits. We present the performance of the CMS tracker in terms of momentum and impact parameter resolutions, evaluated by means of simulation studies.

PACS: 29.40.Gx; 07.05.Kf

Keywords: LHC; CMS; Central tracker; Track reconstruction

1. Introduction

Track reconstruction is a complex task involving mathematical, geometrical and combinatorial problems. The latter issue will be more severe at the LHC because of the high luminosity of the collider. The CMS tracker will record of the order of 50,000 hits per bunch crossing at the nominal LHC luminosity of 10^{34} cm^{-2} s^{-1}. Several pattern recognition algorithms must thus be tried before finding the one(s) that are adequate.

In addition, event selection at high trigger levels relies on online track reconstruction in order to identify leptons, to apply sharp p_T cuts and isolation criteria, to select jets containing a displaced vertex etc. [1]. Online code of high performance and reliability is thus mandatory. We have thus developed an object-oriented framework allowing easy implementation and evaluation of track reconstruction algorithms.

The CMS tracker is described in Section 2. The elements of the reconstruction framework are explained in Section 3. Section 4 discusses two track reconstruction algorithms implemented within this framework, as well as their performance in dense hit environment. Section 5 describes the performance of the CMS tracker in terms of momentum and impact parameter resolutions, evaluated by means of simulation studies.

*Corresponding author.
E-mail address: pascal.vanlaer@cern.ch (P. Vanlaer).

0168-9002/02/$ - see front matter
PII: S0168-9002(01)01802-2

2. The CMS tracker

The CMS tracker is a 5.5 m long, 1.1 m radius detector embedded in a nearly homogeneous 4 T magnetic field. It is meant to provide 13 track points over a pseudorapidity range $\eta \leqslant 2.5$. Silicon pixel counters will equip the innermost detection layers (4 cm $\leqslant R \leqslant$ 11 cm), while silicon strip counters will be used for the outer part. A detailed description can be found in [2].

2.1. The pixel vertex detector

The vertex detector consists of 3 detection layers in the barrel part, and 2 layers in each endcap. The sensors are 250 μm thick, with square, $150 \times 150\ \mu m^2$ pixels. The substrates are made of p-type silicon with n^+ pixel implants. The resolution of the counters in $R\phi$ is optimized by making use of charge sharing between neighbouring pixels. In the barrel part this is done by deliberately not tilting the counters and exploiting the 32° Lorentz deviation for electrons. In the endcaps, the modules will be mounted onto turbine blades rotated by 20° with respect to a radial axis. The expected $R\phi$ resolution is 10–15 μm.

2.2. The silicon strip tracker

The silicon strip tracker consists of an inner barrel with 4 layers, an outer barrel with 6 layers, mini-endcaps with 3 disks each and endcaps with 9 disks each. All sensors are single-sided, with p^+ strips on a n-type bulk. Strips are parallel to the beam direction in the barrel counters, and run radially in the endcaps. Four stereo measurements are provided per track. They are achieved by mounting 2 single-sided counters back to back at an angle between strips of 100 mrad.

The sensor thickness is foreseen to be 320 μm in the inner barrel and first four rings of the endcap disks, and 500 μm for the remaining outer counters. The readout pitch ranges from 81 μm in the innermost counters till 183 μm in the outermost ones. The $R\phi$ resolution varies significantly with the incident angle in the projection perpendicular to the strips and ranges from 10 μm to about 60 μm.

3. Object-oriented framework for track reconstruction

Track reconstruction in a layered detector consists in four steps:

(1) seed generation: finding initial track segments or "seeds";
(2) trajectory building: growing each seed, layer by layer, into one or several track candidates;
(3) trajectory cleaning: removing duplicate tracks and bad fits;
(4) trajectory smoothing: refitting the remaining tracks in order to get optimal parameters all along the trajectory.

The performance of each step depends on the effectiveness of more basic operations: access to hits compatible with a growing trajectory, extrapolation of the track parameters and their covariance matrix in the detector material and magnetic field, update of the track parameters using the hit coordinates measured in the tracker elements; etc. These operations are realized by a collaboration of elements—C^{++} classes—with well defined responsibilities. The main elements are described below.

3.1. Surfaces and reference frames

A Surface is described by its shape, position and orientation in the global reference frame of CMS. A BoundSurface has additional information about its boundaries. A Surface determines a local Cartesian coordinate system, with the Surface position as an origin and the x- and y-axes parallel to the Surface. The Surface knows how to transform coordinates from the local system to the global one and vice-versa.

3.2. The TrajectoryState

The TrajectoryState is a local measurement of a particle trajectory. In the presence of a magnetic field, it can be described by a vector of 5 independent parameters and their covariance matrix. In our framework, all useful parametriza-

tions are supported, as well as transformations of one into another.

A particularly useful parametrization is the one where position and momentum are given in the local frame of a detection surface: $(q/p, \mathrm{d}x_{\mathrm{loc}}/\mathrm{d}z_{\mathrm{loc}}, \mathrm{d}y_{\mathrm{loc}}/\mathrm{d}z_{\mathrm{loc}}, x_{\mathrm{loc}}, y_{\mathrm{loc}})$. This is the frame in which detectors measure hit coordinates, and in which the update of the TrajectoryState is performed.

3.3. The DetUnit

The DetUnit represents one detector module with its sensitive surface, its readout electrode topology and its readout electronics. The main responsibility of the DetUnit is to provide hits, i.e. measurements of track impact points. This requires access to digitized electronics signals (Digis) and grouping of the signals into clusters. Like in the real DAQ, the access to Digis is done through readout units. Clusterization is performed by an algorithm object, the Clusterizer, which can be different for every detector type.

3.4. The ReadoutUnit

The ReadoutUnit represents the RAM buffer of the DAQ in which event fragments, after reception of a Level-1 trigger accept, are pushed and await for online processing. One ReadoutUnit groups the Digis of 50-100 DetUnits. The exact grouping of DetUnits into ReadoutUnits is not yet fixed and will be the outcome of DAQ optimization studies.

When a DetUnit is asked for its hits, its Digis may first have to be fetched from the ReadoutUnit before clusterization can start. This triggers filling of all the other DetUnits connected to the same ReadoutUnit in a single master—many slaves logic. The Digis are then cached in the DetUnits for the rest of the event processing; no further access is needed for that ReadoutUnit.

3.5. The DetLayer

In the CMS tracker, tracking algorithms can take advantage of the fact that sensitive elements and dead material are arranged in layers. A DetLayer is a set of DetUnits providing hermetic coverage of the detection surface within its boundaries. Examples are forward silicon strip disks, barrel silicon strip cylinders etc. The DetLayer implements two essential functions:

- effective access to hits compatible with a TrajectoryState;
- navigation to the neighbouring layers that might contain the continuation of a trajectory.

The effectiveness of the access to hits is due to the fact that DetUnits in a layer, and hits in a DetUnit, are sorted according to their position. Finding a hit compatible with a given TrajectoryState is a matter of performing two linear searches, one in z in the barrel layers (respectively in r in the endcaps) and one in ϕ.

The navigation links are computed analytically making hypotheses on the origin of the tracks that must be reconstructed (the beam spot plus some tolerance) and on their minimum p_{T}.

3.6. The Propagator

The task of Propagators is to extrapolate TrajectoryStates in the magnetic field of CMS, accounting for dead material. In addition to a wrapper to the GEANE numerical algorithm [3], a very fast, reasonably precise implementation optimized for the tracker is provided. It is based on the approximations that the magnetic field is almost uniform and that dead material is concentrated in detection surfaces.

3.7. The Updator

This object constraints a TrajectoryState with a hit, and produces a new, more precise TrajectoryState. A Kalman filter implementation is provided. The use of constraints of various dimensionalities, like a 2D position in a tracker module and a 4D position + direction in a muon station, is allowed.

4. Trajectory building algorithms

Using the reconstruction framework, the logic of trajectory building becomes simply a matter of combining navigation, access to compatible hits,

and hermeticity. Starting from the last layer of a growing trajectory candidate:

(1) ask for the next layers;
(2) for each of these layers, ask for the hits that are compatible with the current TrajectoryState extrapolated onto the layer;
(3) if more than one compatible hit is found, resolve ambiguities;
(4) if no hit is found count one missing hit;
(5) if too many successive hits are missing, stop building the trajectory candidate, store it if it is long enough, and proceed with the next candidate.

The two algorithms described here mostly differ in the way they deal with ambiguities. At each layer, the combinatorial algorithm creates one trajectory candidate per compatible hit found, plus one candidate which does not include any hit, to account for a possible layer inefficiency. The number of candidates grows exponentially as trajectory building proceeds. To limit this number, only the candidates with the smallest χ^2 are kept at each layer, a penalty being added to the χ^2 for each missing hit.[1] Duplicates are removed only after trajectory building, which leads to a waste of CPU.

In the Deterministic Annealing Filter (DAF) [4], instead, all compatible hits found in a layer contribute in the update of the TrajectoryState. The weight of each hit is given by an assignment probability depending on the reduced distance between the hit and the TrajectoryState. To avoid pulling the trajectory too much towards outliers, the hit resolutions are first blown up by a factor of ≈ 100, then "annealed" towards their true value in a few steps (8 steps for the results shown below). In contrast with the combinatorial algorithm, only one trajectory candidate is grown per seed.

Fig. 1 shows the computation time spent per track for the two algorithms, as a function of the density of noise hits.[2] A density of 100% corresponds to one additional noise hit for each hit from a particle. The DAF is 3 times faster than the combinatorial algorithm, with a CPU time of 0.3 s per track at the maximum occupancy studied as compared to 0.9 s per track. Checks of the statistical consistency of the DAF have also been performed, which have demonstrated the unbiasness and robustness of that algorithm up to the maximum occupancy studied.

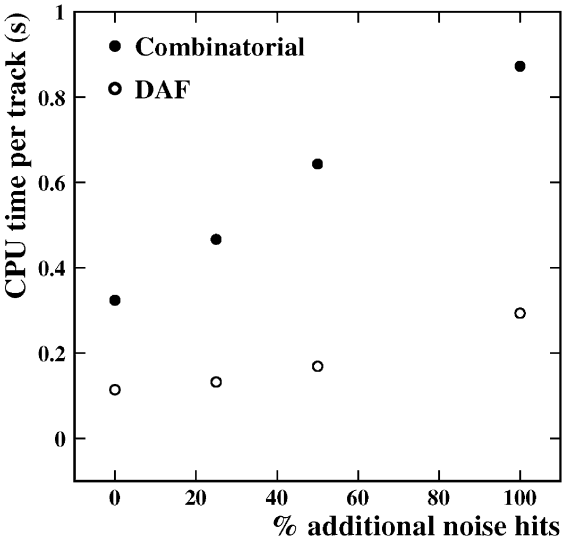

Fig. 1. CPU time per track for the combinatorial and DAF algorithms.

5. Performance of the CMS tracker

The performance of the tracker layout described in Section 2 was evaluated by means of simulation studies using CMSIM, the CMS detector simulation program, and ORCA, the Object-oriented Reconstruction program for CMS Analysis. This section describes preliminary results obtained with the combinatorial track reconstruction algorithm.

Fig. 2 shows the resolution in p_T as a function of η for three values of p_T: 1, 10 and 100 GeV/c. At $p_T > 100$ GeV/c the p_T resolution is determined by the spatial resolution of the tracker modules and scales roughly as $\sigma(p_T)/p_T \simeq 1.5 \times 10^{-4} p_T$, p_T expressed in GeV/c. The degradation at $\eta > 1.5$ is due to the fact that particles exit the tracker at $R < 1.1$ m, which leads to a worsening of the

[1] If no restrictions on the number of missing hits were applied, the number of candidates per seed would be $\sim 2^{13}$ to allow for an inefficiency at each layer.

[2] The processor used is an Intel P-II 400 MHz with 256 MB RAM.

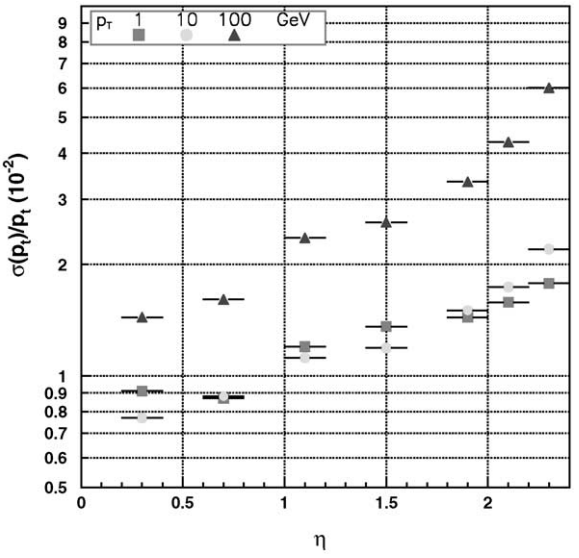

Fig. 2. Resolution in p_T as a function of η, for $p_T = 1, 10$ and 100 GeV/c.

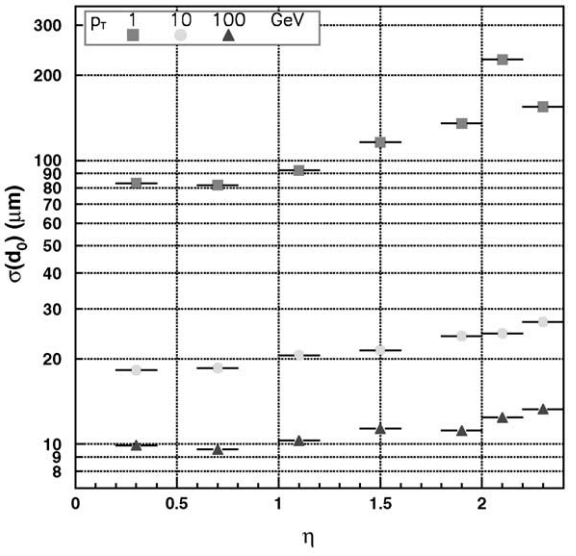

Fig. 3. Resolution in the transverse impact parameter d_0 as a function of η, for $p_T = 1, 10$ and 100 GeV/c.

sagitta measurement. For $p_T = 1$ and 10 GeV/c, the dominant contribution to the p_T resolution comes from multiple scattering. The resolution is around 1% in the barrel and its dependence in η reflects the amount of material traversed.

Reconstruction of B-hadron decay vertices and b and τ-jet tagging requires good resolutions in the transverse and longitudinal impact parameters. Fig. 3 shows the resolution in the transverse impact parameter d_0 as a function of η for the same three values of p_T. A d_0 resolution around 100 μm is expected for 1 GeV/c particles, and the asymptotic value at high p_T is 10 μm. Previous studies [2] have shown that such a resolution allows to tag 100 GeV b-jets with an efficiency around 50% while maintaining the mistagging rate from light quark (u, d, s) and gluon jets at the level of 1–2%.

6. Conclusions

We have developed a powerful and flexible environment for the implementation and evaluation of track reconstruction algorithms. We have successfully implemented a combinatorial algorithm using the Kalman filter approach. The performance of the CMS tracker was evaluated using this algorithm to reconstruct simulated events. Preliminary results indicate that the momentum resolution is around 1% for low p_T tracks and scales approximately as $1.5 \times 10^{-4} p_T$ for $p_T >$ 100 GeV/c. The transverse impact parameter resolution ranges between 10 μm for $p_T >$ 100 GeV/c and 100 μm at $p_T = 1$ GeV/c.

Algorithms that are robust to noise are being developed. The Deterministic Annealing Filter shows promising results: statistical efficiency and unbiasness, and CPU consumption per track about 3 times smaller than required for the combinatorial algorithm. The performance of these algorithms for online event selection is being studied in the framework of the CMS Physics Reconstruction and Selection groups.

References

[1] CMS Collaboration, The Level-1 Trigger TDR, CERN/LHCC 2000-038, 2000.
[2] CMS Collaboration, The Tracker Project TDR, CERN/LHCC 98-6, 1998. CMS Collaboration, Addendum to the Tracker TDR, CERN/LHCC 2000-016, 2000.
[3] V. Innocente, et al., GEANE—Average Tracking and Error Propagation Package, CERN writeup W5013, 1994.
[4] R. Früwirth, A. Strandlie, Comput. Phys. Commun. 120 (1999) 197.