CMS reconstruction improvements for the tracking in large pileup events

This content has been downloaded from IOPscience. Please scroll down to see the full text.
(http://iopscience.iop.org/1742-6596/664/7/072040)

View the table of contents for this issue, or go to the journal homepage for more

Download details:

IP Address: 137.138.93.202
This content was downloaded on 09/03/2016 at 07:58

Please note that terms and conditions apply.
CMS reconstruction improvements for the tracking in large pileup events

M Rovere\textsuperscript{1} on behalf of the CMS collaboration

\textsuperscript{1}CERN, Geneva, Switzerland

E-mail: marco.rovere@cern.ch

Abstract.

The CMS tracking code is organized in several levels, known as iterative steps, each optimized to reconstruct a class of particle trajectories, as the ones of particles originating from the primary vertex or displaced tracks from particles resulting from secondary vertices. Each iterative step consists of seeding, pattern recognition and fitting by a kalman filter, and a final filtering and cleaning. Each subsequent step works on hits not yet associated to a reconstructed particle trajectory.

The CMS tracking code is continuously evolving to make the reconstruction computing load compatible with the increasing instantaneous luminosity of LHC, resulting in a large number of primary vertices and tracks per bunch crossing.

The major upgrade put in place during the present LHC Long Shutdown will allow the tracking code to comply with the conditions expected during RunII and the much larger pileup. In particular, new algorithms that are intrinsically more robust in high occupancy conditions were developed, iterations were re-designed (including new ones, dedicated to specific physics objects), code optimizations were deployed and new software techniques were used. The speed improvement has been achieved without significant reduction in term of physics performance.

The methods and the results are presented and the prospects for future applications are discussed.

1. CMS Track Reconstruction

The central component of the CMS experiment \cite{1} is the world’s largest all-silicon detector, composed of an inner Pixel detector (arranged in three barrel layers and two forward disks, for a total of 66 millions of channels, with pixel size of 100×150 \(\mu\)m\(^2\)) and an outer Strip detector (arranged in a barrel-forward complex geometry, for a total of 9.6 millions of channels, with pitch in the range 80-180 \(\mu\)m and length in the range 10-20 cm). A key component of the Strip detector is the usage of double-sided modules, which are composed of two different strip detectors glued together with a stereo angle of 100 mrad, to provide three-dimensional position measurements in global coordinates.

The CMS tracking \cite{2} is based on a Kalman filter and can be logically divided into four steps: the \textit{seeding}, in which a proto-track is formed starting from two or three consecutive hits with either a beamspot or a vertex constrain; the \textit{pattern recognition}, during which the proto-track is propagated into the CMS tracker and compatible hits are associated to the proto-track; the \textit{fitting}, in which the best parameters’ estimate is computed for all hits along the trajectory; and the final \textit{selection}, in which quality criteria are applied to the tracks to reject the badly
reconstructed ones and to reduce the fake rate\(^1\). This procedure is run iteratively: at each iteration, the hits associated to high quality tracks are masked so that the next iterations face a much reduced combinatorics problem. The driving principle is, though, to reconstruct easy tracks first, allowing for more complex algorithm to run in the later iterations. During RunI, seven iterations were used, as illustrated in Table 1. The logic behind the definition of the iterations used during RunI could be evinced looking at Figure 1 and Figure 2. The very first iterations are aimed at reconstructing prompt tracks within the full \(p_T\)-spectrum, identified by the blue-ish colors in Figure 1; the later iterations, thanks to the much lower combinatorics they have to face, are tuned to reconstruct much more difficult tracks, namely displaced and very displaced ones, identified by the green-ish color in Figure 2.

<table>
<thead>
<tr>
<th>Order</th>
<th>Name</th>
<th>Seed</th>
<th>Target Track</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Initial</td>
<td>pixel triplets</td>
<td>prompt, high (p_T)</td>
</tr>
<tr>
<td>1</td>
<td>LowPtTriplet</td>
<td>pixel triplets</td>
<td>prompt, low (p_T)</td>
</tr>
<tr>
<td>2</td>
<td>PixelPair</td>
<td>pixel pairs</td>
<td>recover high (p_T)</td>
</tr>
<tr>
<td>3</td>
<td>DetachedTriplet</td>
<td>pixel triplets</td>
<td>displaced</td>
</tr>
<tr>
<td>4</td>
<td>MixedTriplet</td>
<td>pixel+strip triplets</td>
<td>displaced</td>
</tr>
<tr>
<td>5</td>
<td>PixelLess</td>
<td>strip pairs</td>
<td>more displaced</td>
</tr>
<tr>
<td>6</td>
<td>TobTec</td>
<td>strip pairs</td>
<td>very displaced</td>
</tr>
</tbody>
</table>

Figure 1. Performance of iterative tracking in RunI: efficiency vs \(\eta\)

Figure 2. Performance of the iterative tracking in RunI: efficiency vs the production radius.

\(^1\) A fake track is a reconstructed track that has no correspondence in reality. In Monte Carlo simulation, fake tracks are characterized as tracks not matching the trajectory of any generated charged particle. The fake rate is the ratio between fake tracks and all tracks.
2. Tracking Developments

2.1. Motivations

Tracking in a high energy physics experiment is fundamentally a combinatorial problem: hence, one of the main components that will affect its output, both in terms of accuracy and timing, is the presence of multiple pileup (PU) collisions. This will become even more important during the imminent LHC Run II: the run is expected to start with the LHC operating at 50 ns bunch crossing (BX), collecting ∼1 fb⁻¹ of data with an average PU(⟨PU⟩) ∼30, followed by an extended period of data taking at 25 ns BX, with ⟨PU⟩ increasing from an initial value of ∼25 up to ∼45, collecting a total of ∼10 fb⁻¹. The effects that these new scenarios would have on the Run I tracking configuration are illustrated in Figure 3 and 4. The overall effect of an increase in the PU is a higher occupancy in the CMS tracker. While this translates into a +5% more occupancy in the pixel detector, the effect is much more severe on the Strip tracker, where the increase reaches 45%. In particular the increase in the occupancy is more evident on double-sided strip modules, where ambiguities may produce ghost hits. From Monte Carlo simulations, the rate of ghost hits was always found to increase with PU and, in the innermost layers of the Strip Tracker, becomes larger than the real hits at ⟨PU⟩ ∼ 40. This has dramatic consequences on the timing of the iterations that are seeded using double-sided hits, like PixelLess and TobTec. Figure 3 shows that the iterations seeded by pixel hits scale roughly linearly with the increase in the PU, while the ones seeded by Strip hits, like iterations 5 and 6, show an almost exponential dependency on the PU. Despite the lower increase in occupancy, the Pixel Detectors show the effects of an increase in PU through the so called pixel dynamic inefficiency, which is mainly caused by the saturation of the readout chip buffer. The net effect, measured on 2012 data, is illustrated in Figure 4.

![Figure 3: Timing of RunI iterations, normalized to iteration 020PU.](image1)

![Figure 4: Pixel Dynamic Inefficiency.](image2)

In order to overcome the problems that the increase in ⟨PU⟩ causes to tracking reconstruction, CMS has followed a twofold strategy [3]: on one hand, reduce the timing in order to fit the processing budget to perform Tier-0 reconstruction; on the other hand, focus on developments targeting improvements on the physics side. The two strategies will be explained in the following sections.
2.2. Timing-Oriented developments

Figure 3 clearly shows that the first priority for tracking in CMS was to reduce as much as possible the time needed for the strip-seeded iterations, trying to keep the same physics accuracy, or even improving it. A good seeding can improve tracking performances dramatically. On one hand, constructing high-quality seeds is a guarantee to spend time in the next steps reconstructing good tracks and reducing, at the same time, the fake rate. On the other hand, a precise estimate of the track parameters and errors computed at seeding stage can better drive the process of pattern recognition, reducing the combinatorics while associating directly the best hits to the trajectory. Both results have been achieved designing a new seeding algorithm for the PixelLess and TobTec iterations, by extending the strip-pair pattern to include an additional third hit. The main feature of the new algorithm is a straight line fit of the three hits in the R-z plane, including the dynamic estimation of their position taking into account the seed direction hypothesis for an even higher precision. The fit $\chi^2$ is used as a criterion to accept or reject the triplet. The effect of the new algorithm is illustrated in Figure 5 and Figure 6: the number of produced seeds is reduced by a factor of two, while essentially the same number of tracks are produced as using the old seeding code.

![Figure 5.](image1.png) **Figure 5.** Number of seeds produced, as a function of $\eta$, by the PixelLess and the TobTec iterations using the old seeding (black curves) and the new seeding (red curves).

![Figure 6.](image2.png) **Figure 6.** Number of tracks produced, as a function of $\eta$, by the old seed algorithm (black curves) and by the new seeding (red curves), for iterations PixelLess and TobTec.

Another factor, mainly relevant for the strip-seeded iterations, is the increase in the occupancy of the Strip tracker that is visible while running at 25 ns BX and that is due to the out of time particles. The out of time pileup is responsible of an increase by a factor of two both in the reconstruction time and in the fake rate. A peculiar feature of this kind of pileup is its charge distribution: since the particles coming from the interactions produced by different bunches with respect to the in-time one arrive at random time, the corresponding clusters are characterized by a low collected charge. A selection on the cluster charge (CCC) has been introduced: this can be applied with increasing strength at different stages of track reconstruction: upfront, directly during the cluster reconstruction; during seeding and during pattern recognition, where the sensor thickness and the trajectory crossing angle are taken into account to achieve a higher rejection power. During RunII, a stable selection efficiency will be ensured by the inclusion
of the strip gain calibration in the so called **Prompt Calibration Loop** [4]. The effects of the developments so far described is illustrated in Figure 7 and 8: both the new seeding algorithm and the introduction of CCC guarantee each a gain of a factor of two in the timing of the strip-seeded iterations, while the CCC alone reduce the fake-rate by \(\sim 50\%\).

![Figure 7](image1.png)  ![Figure 8](image2.png)

**Figure 7.** Time spent in each iteration for RunI tracking (black curve), after the introduction of the new seeding algorithm for strip-seeded iterations (red curve) and adding the Cluster Charge Cut on top of it (blue curve).

**Figure 8.** Fake-rate before (red curve) and after (blue curve) the introduction of the Cluster Charge Cut during track reconstruction.

In addition to the major developments described above, many other improvements have been introduced in order to further reduce timing, while keeping the same physics performances. In particular, the order of the iterations has been changed so that faster iterations run first. The seeding and the selection criteria have been modified to minimize the number of tracks sequentially reconstructed by different iterations. The code re-factoring, the introduction of modern C++11 features and the adoption of more advanced compilers gained another significant speedup both in terms of timing and in terms of better memory usage.

Several \(t\bar{t}\) Monte-Carlo samples with realistic alignment and calibration conditions have been produced and reconstructed with different releases under various scenarios (\(\langle PU\rangle = \{25, 40, 70, 140\}\) at BX=25 ns and \(\langle PU\rangle = 25\) at BX=50 ns), in order to properly benchmark the timing improvements and check the final physics performances.

The reconstruction time has been evaluated both with the RunI software release and with the current development release for RunII. Note that it was not possible to process the \(\langle PU\rangle = 140\) sample with the RunI release.

The results are summarized in Figure 9 and in Figure 10: the achieved gain in terms of reconstruction time is equal to a factor of 2, 3 and 4 for \(\langle PU\rangle = 25\), \(\langle PU\rangle = 40\) and \(\langle PU\rangle = 70\), respectively. Moreover, Figure 10 seems to suggest that the timing dependence as a function of the \(\langle PU\rangle\) is not exponential, as it was using the RunI software release, but is somewhat closer to a linear dependency (in the \(\langle PU\rangle\) range explored).
2.3. Physics-Oriented developments

The improvements detailed in the previous sections are such that the reconstruction time fits the timing budget driven by prompt reconstruction at Tier-0. More care has been then devoted to also improve the physics performances of tracking in CMS, since now a small trade-off between timing and physics performances could be achieved.

The main targets for this further tuning have been the high-level physics objects that are crucial for many CMS analyses, namely muons and (tracking in) high-p_T jets. In 2012 data, a PU-dependent loss of muon reconstruction efficiency in the tracker was noticed (Figure 11). The addition of an outside-in iteration seeded using information coming from the outermost muon detectors fully recovers the lost efficiency. Moreover, another iteration has been added to re-reconstruct muon-candidate tracks with somewhat looser requirements, to recover hit collection efficiency for this particular kind of tracks to enhance the momentum resolution and the particle identification. The results of this iteration are illustrate in Figure 12.

Especially for LHC RunII conditions, TeV-scale jets are a very dense environment, characterized by a small two-track separation that could possibly cause nearby clusters to be merged, producing only one hit with badly estimated position and uncertainties. Tracking in such a harsh environment is anyway crucial in order to have efficient b and τ tagging and for resolving jet substructures. Hence, a dedicated, regional iteration has been developed in order to improve the track reconstruction efficiency. High-p_T calorimetric jets are first identified, the merged pixel clusters that lie in a narrow region around them are split and finally track reconstruction is performed, allowing for a large number of parallel track-candidates. The threshold on the jet p_T is a careful balance between the smooth improvement in physics performance and the increase in overall computing time. The effect of the new iteration is illustrated in Figure 13.
Figure 11. Muon reconstruction efficiency versus PU for RunI reconstruction (red band) and for RunII candidate release (black dots).

Figure 12. Muon (with at least hits in seven layers in the tracker) reconstruction efficiency as a function of \( \eta \) for RunI reconstruction (red curve) and for RunII candidate release (black curve).

Figure 13. Track reconstruction efficiency as a function of track-jet separation for tracks inside high-p\( _{T} \) jets.

3. Overall Performances
The impact of the developments presented in this paper is evaluated in terms of track reconstruction performance at different \( \langle PU \rangle \) by comparing the results obtained with the software release used for RunI reconstruction to those obtained with the current RunII candidate release. Figure 14 shows plots of efficiency for high-p\( _{T} \) prompt tracks vs \( \eta \), fake-plus-duplicate rate vs
\( \eta \) and efficiency for soft tracks vs their production radius for nominal conditions (RunI-like for RunI release, RunII-like for current release). The efficiency for prompt tracks is similar, while the efficiency for displaced tracks is slightly reduced in the current release. As a matter of fact, performance under nominal conditions are very similar and one can conclude that in RunII CMS tracking performance will be close to those in RunI, but in a much more challenging environment.

Figure 14. Performances of RunI and RunII candidate release at their respective nominal conditions.

4. Conclusion
The high PU is the main challenge for tracking due to an increased occupancy and a degradation of detector performances. In order to improve tracking in face of the coming LHC RunII, CMS developed a twofold strategy: reduce combinatorics first and only after improve the performance of specific physics object. The obtained results are excellent, with timing under control and RunII performances comparable or better than RunI.

New challenges are coming for tracking at CMS after RunII that will likely require the modification of the reconstruction algorithms to fully exploit the improved detector geometries and performances, like the ones designed for the CMS Upgrade programs. A first implementation with upgrade geometries is already fully functional.

At the same time, the current trend of computing technologies goes in the direction of many-cores processors with large vector units; in order to efficiently exploit this kind of hardware, new algorithms need to be developed or old algorithms need to be reworked so that they are easily parallelizable and compliant with the SIMD (single-instruction-multiple-data) paradigm. As a first step in this direction the CMS tracking software is now thread safe and can be executed in multi-threaded processes.

References
[1] Chatrchyan S et al. (CMS) 2008 JINST 3 S08004