TrickTrack: An experiment-independent, cellular-automaton based track seeding library

Valentin Völkl[1,2,*], *Felice* Pantaleo[1,**], and *Benedikt* Hegner[1,***]

[1]CERN, CH-1211 Geneva 23, Switzerland
[2]University Innsbruck, Innrain 52, 6020 Innsbruck, Austria

Abstract.
The design of next-generation particle accelerators evolves to higher and higher luminosities, as seen in the HL-LHC upgrade and the plans for the Future Circular Collider (FCC). Writing track reconstruction software that can cope in these high-pileup scenarios is challenging due to the inherent complexity of current algorithmic approaches. In this contribution we present TrickTrack, a track reconstruction toolkit based on the cellular automaton-based algorithm used for track seeding in the CMS experiment. It is a concurrency-friendly implementation of an algorithm for pattern recognition problems, and tries to remain general enough to be of use in most tracking detectors. The performance of TrickTrack in the FCC-hh design study, which features pileup rates of 1000 interactions per bunch crossing and a high-occupancy environment for tracking, is presented as the first use case beyond CMS.

1 Introduction

Particle physics experiments using tracking detectors to measure particle properties are faced with the combinatorial task of associating detector hits to particle tracks. Increasing the luminosity in particle colliders commonly introduces additional collisions at each bunch crossing (pileup) and increases detector occupancy. More efficient solutions to this computational problem thus allow for higher data acquisition rates and increased sensitivity for rare processes. In future colliders such as the proposed hadron-hadron collider FCC-hh, the increase in pileup rate requires dedicated studies of the feasibility of track reconstruction with current algorithms and software to support these studies.

In this paper we report on TrickTrack [1], a standalone library created from the cellular automaton-based seeding code used in the CMS experiment, in order to establish the performance of current algorithms in the context of the FCC and to enable developments to extend them. The implementation of the code and its usage in CMS are described in detail in Ref. [2].

The basic concepts of the HitChainMaker algorithm that is at the core of the TrickTrack library are illustrated in Fig. 1. Information on the exact detector geometry only enters via the hit position and connections between tracking layers, eliminating the need for a detailed

*e-mail: valentin.volkl@cern.ch
**e-mail: felice.pantaleo@cern.ch
***e-mail: benedikt.hegner@cern.ch

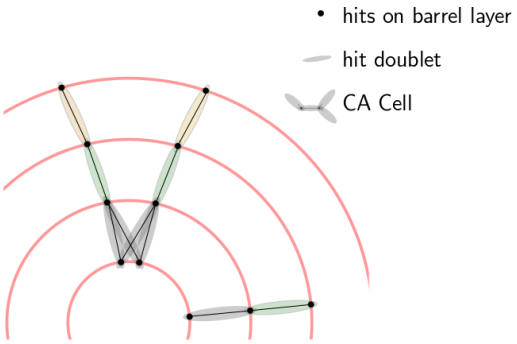

Figure 1. Schema of the basic concepts of the HitChainMaker algorithm. Hits on neighboring layers are connected to *hit doublets*, which form the cells of the cellular automaton. Track seeds are created from doublets that share a hit. To grow the track seeds, the cellular automaton is evolved using an update rule that increments an internal integer state variable of any cell if it has inner neighbors with the same state. The states of the cells after two iterations are represented by different colors.

geometry description for any experiment that measures three dimensional space points. Storing hit positions by reference to the module and channel that was hit is usually slighly more efficient. However, this representation allows the library to be effectively decoupled from details of the experimental geometry and thus be very general.

The cells acted on by the algorithm are formed by doublets of hits on neighboring layers and connections between doublets sharing one hit. The number of possible track candidates can be reduced both in the doublet creation step and while connecting the cells using default or user-implemented filters. The default geometric filters included in the libraries will be useful for most detector geometries in which the magnetic field in the seeding region is approximately constant. They filter track seeds whose curvature and longitudinal alignment exceeds a given parameter and is not compatible with a track originating from a luminous region, which is also defined by the user. This filtering step is easily extensible and adaptable as any classifier for two and three hits can be used in conjunction with TrickTrack. To extract tracklets of length n from the connected cells, the HitChainMaker performs $n - 2$ evolutions of a cellular automaton, incrementing the state of each cell with same-state inner neighbors. A depth-first search starting from cells whose state equals n finally yields all possible track candidates.

As it relies exclusively on local operations, the HitChainMaker algorithm is well adapted to parallel computing infrastructures. However, the computing performance is still dominated by the filters used to suppress fake tracks. Fig. 2 shows the pure computing performance for varying fake rates at constant input size (or equivalently varying input size at constant fake rate).

2 Track parameter estimation using the Riemann Fit

Many approaches to track reconstruction including the Kalman Filter require an estimate of the parameters of the initial track comprised of the seeding hits. The Riemann fit [3] is well-suited for obtaining a first estimate of the track parameters, requiring as input only a subset of the hits of the track. Fig. 3 shows that the computing performance is sufficient, allowing the calculation of track parameters even for a large number of track seeds. This

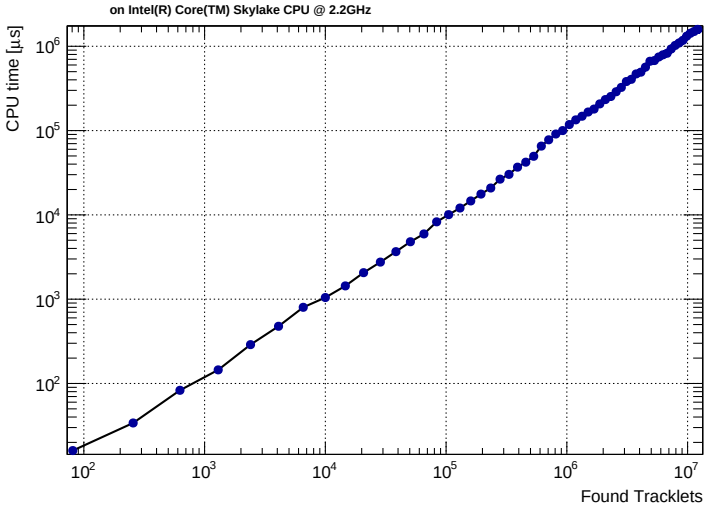

Figure 2. Computing performance of the TrickTrack implementation of the HitChainMaker.

method is limited to homogeneous solenoidal magnetic fields, but this condition is typically fulfilled in the detector regions where tracks are reconstructed. The Riemann fit is not an iterative method, but operates on matrices that scale with the number of hits to be fitted. For performance reasons it is preferrible to operate on fixed-size matrices, which require the user to declare the maximum number of hits in one fit during compilation. Fig. 3 also shows the cost of expanding the fit to more than the typical number of hits in a track seed.

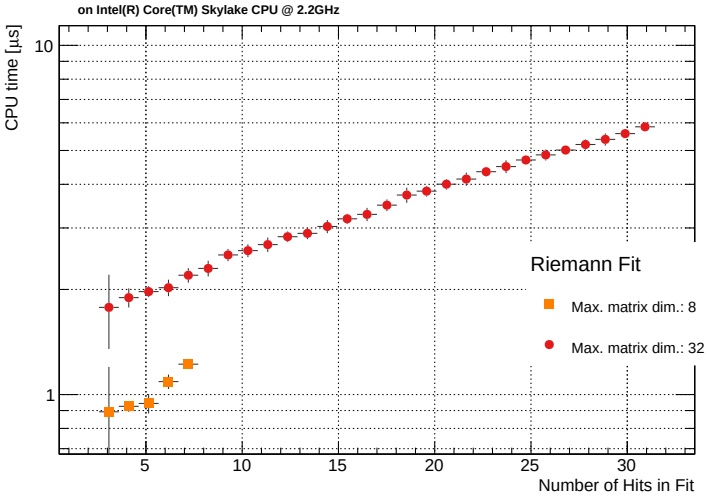

Figure 3. Computing performance of the TrickTrack implementation of the Riemann Fit. The maximum number of hits to be fitted is a compile-time constant that determines matrix size and thus the performance of the fit.

3 Software infrastructure

In order to fit into a modular software ecosystem and enhance usability, TrickTrack follows the HEP Software Foundation best practices [4]. Dependencies are kept minimal. Apart from the use of the Eigen library [5] for linear algebra, there are no required external libraries. Special consideration is given to the issue of integrating the library into software frameworks currently in use by experiments. The code is fully templated on the data structure describing the track hit in order to avoid unnecessary data conversions.

Capturing the library logging output in external software frameworks is a requirement for thorough bookkeeping of compute jobs. TrickTrack fulfills this requirement by its use of the spdlog library [6] and customizable macros.

A high level interface interface with python bindings decreases the work needed to setup and run the reconstruction code for new users.

4 Application to the Future Circular Collider

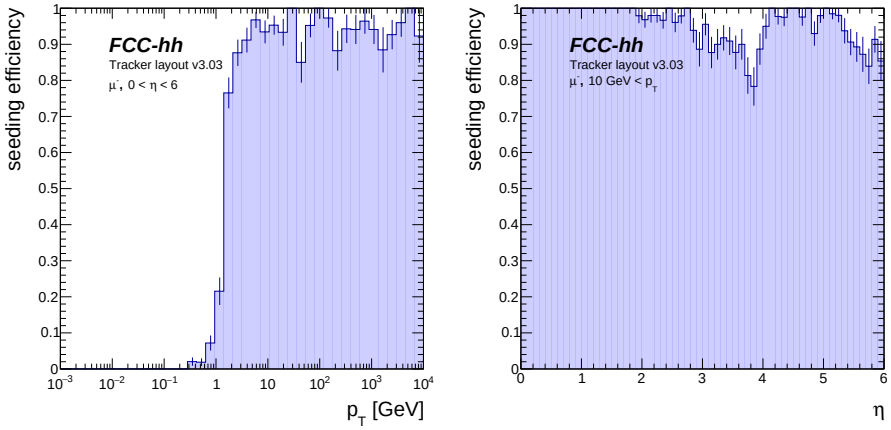

Figure 4. Seeding efficiency for single muons in the FCC-hh tracker. The left plot shows the efficiency versus transverse momenta, the right plot shows efficiency versus the particle trajectory inclination towards the beamline. The cut for the right plot is $p_T > 10$ GeV, so as not to show inefficiencies of low-p_T particles. The TrickTrack-based track seeding covers the full design range of the FCC-hh tracker for very forward and low-momentum particles.

Beyond the original use in CMSSW, a software collection for data processing in the CMS experiment, the decoupled code has been applied to track reconstruction for the Future Circular Collider (FCC) Design Study [7]. Experiments at the hadronic Future Circular Collider (FCC-hh), a proposed high-energy frontier particle accelerator using a 100 km tunnel infrastructure to reach center of mass energies of up to $\sqrt{s} = 100$ TeV, will have many similarities with current LHC experiments. However the increased center-of-mass collision energy imposes some new requirements on the detectors, such as a larger acceptance in the forward region, and track reconstruction in particular is greatly complicated by the large rate of simultaneous proton-proton collisions per bunch crossing. Current designs foresee more then 1000 of these pileup collisions in addition to the hard scatter event that triggered data taking.

To study the efficiency of the reconstruction, the detector response to single muons was simulated using Geant4 full detector simulation integrated in the FCC software framework

[8]. The tracker layout used is shown in Fig. 5 and is the baseline layout for the FCC-hh detector studies. TrickTrack , integrated as part of the reconstruction chain in the FCC software framework, reconstructs the simulated data forming track seeds comprised of hit quadruplets. The efficiency for single muons is shown in Fig. 4.

4.1 Detector geometry and layer graph

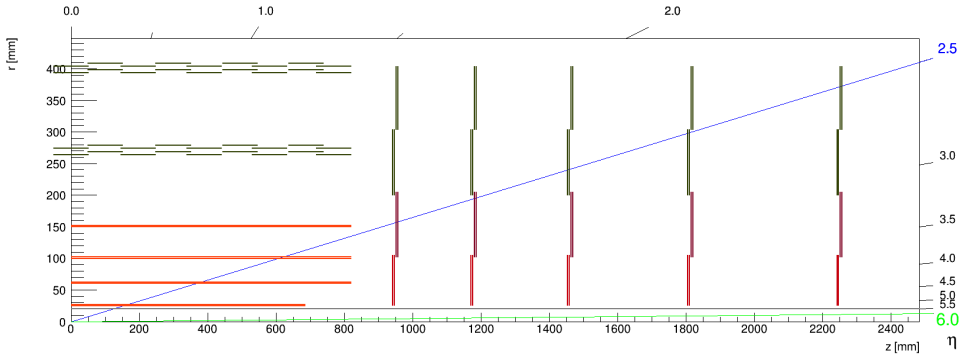

Figure 5. FCChh tracker geometry v3.03 [9], showing the longitudinal layout of modules comprising the inner layers used for seeding.

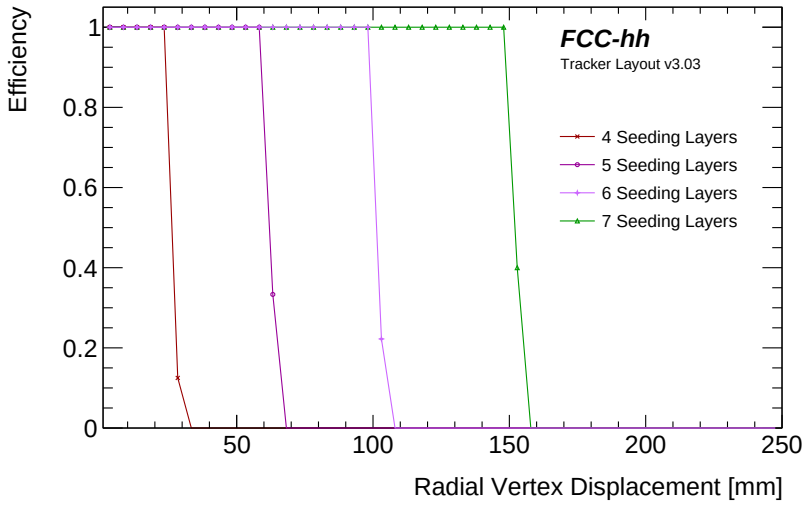

Figure 6. Seeding acceptance for tracks with start vertices displaced radially from the luminous region, for different numbers of seeding layers. Increasing the number of seeding layers allows for reconstruction of tracks originating even outside the innermost barrel layers.

In the case of FCC-hh, full coverage up to $\eta = 6$ can be achieved by using the inner barrel and the inner and forward endcaps in the seeding graph. The acceptance for displaced tracks can be extended by including the outer barrel layers. The acceptance for all tracks whose production vertices are still within four seeding layers is almost perfect, as shown in Fig. 6.

5 Conclusion

The application of the CMS track seeding code, decoupled from the CMS software framework in a standalone library to the FCC design study demonstrates the potential for synergies in track reconstruction software. TrickTrack is integrated into the common FCC software framework and FCC-hh in particular profits from the efficient implementation as a baseline for current track reconstruction solutions and the flexibility to extend the code for further developments to cope with the high pileup rates.

References

[1] V. Volkl, *HSF/TrickTrack: v1.0.6: Graph utils, python bindings, functional filters* (2018), https://doi.org/10.5281/zenodo.1305816

[2] F. Pantaleo, A. Schmidt, V. Innocente, B. Hegner, A. Pfeiffer, A. Meyer, *New Track Seeding Techniques for the CMS Experiment* (2017), https://cds.cern.ch/record/2293435

[3] R. Frühwirth, A. Strandlie, Journal of Physics: Conference Series **762**, 012032 (2016)

[4] B. Hegner, *HSF Project Best Practices (HEP Software Foundation)*, HSF-TN-2016-PROJ

[5] G. Guennebaud, B. Jacob et al., *Eigen v3* (2010), http://eigen.tuxfamily.org

[6] G. Melman, *spdlog v1.3.0* (2016), https://github.com/gabime/spdlog

[7] Z. Drasal, *Status & challenges of tracker design for fcc-hh*, Presentation at the 26th International Workshop on Vertex Detectors (2017), https://indico.cern.ch/event/627245/contributions/2675784/

[8] A. Zaborowska, Journal of Physics: Conference Series **898**, 042053 (2017)

[9] Z. Drasal et al., *Official FCC-hh tracker layout repository: Tracker in flat configuration - v3.03* (2017), http://fcc-tklayout.web.cern.ch/fcc-tklayout/FCChh_v3.03/index.html