Fast Track Seed Selection for Track Following in the Inner Detector Trigger Track Reconstruction

Andrius Vaitkus on behalf of the ATLAS Collaboration
University College London, Gower St, London, WC1E 6BT, UK
E-mail: andrius.vaitkus@cern.ch

Abstract. During ATLAS Run 2, in the online track reconstruction algorithm of the Inner Detector, a large proportion of the CPU time was dedicated to the track finding. With the proposed HL-LHC upgrade, where the event pile-up is predicted to reach $\langle\mu\rangle = 200$, track finding will see a further large increase in CPU usage. Moreover, only a small subset of track candidate seeds is accepted after the track finding procedure, spending the CPU time on seeds that are discarded. Therefore, a computationally cheap track candidate seed pre-selection procedure based on approximate track following was designed, which is described in this report. The algorithm uses a simplified track extrapolation and a combinatorial Kalman filter simplified by a reference-related coordinate system to find the best track candidates. For such candidates, a set of numerical features were created to classify seeds using a Support Vector Classifier, tuned with a high True Positive rate to ensure no significant loss of track finding efficiency. The algorithm was implemented into the Athena framework for online seed pre-selection, and could be potentially adapted for the ITk geometry for the Run 4 of the HL-LHC.

ATL-DAQ-PROC-2023-002
02 March 2023

1. Introduction

The ATLAS experiment [1] is a multipurpose particle detector of the Large Hadron Collider (LHC). One of its core components is the Inner Detector (ID), which is designed for fast track and vertex finding in the pseudorapidity range of $|\eta| < 2.5$, defined in terms of the polar angle θ as $\eta = -\ln\tan(\theta/2)$. It comprises an inner silicon Pixel detector, semiconductor tracker (SCT) and an outer transition radiation tracker (TRT). Events are first processed at a 40 MHz bunch crossing rate by the the first level hardware trigger system (L1) and selected at a rate of about 100 kHz. They are then further filtered by software algorithms in the High Level Trigger (HLT), which records events to permanent storage at $\approx$ 1.2 kHz. The Fast Track Finder (FTF) [2] is an algorithm developed to create track candidates early in the HLT system. It includes the generation of track seeds from triplets of spacepoints (SPs), track candidate extrapolation and fitlering using a combinatorial Kalman Filter. Its design philosophy is to prioritise efficiency over purity. The track candidates can be further refined with a precision tracking stage.

In the FTF, the track following is the most computationally expensive task, taking up a large proportion of the CPU time [2]. Moreover, only a small subset of track candidates coming from Pixel-only seeds are accepted after the track following procedure, meaning the CPU time used for track following is wasted for discarded candidates. The work described in this report is aimed to design a computationally cheap pre-selection procedure based on an approximate track following to identify and reject bad Pixel seeds early, while ensuring no significant loss of track finding efficiency.

2. Seed Preselection Algorithm

For this study, Monte Carlo (MC) generated $t\bar{t}$ events with the centre-of-mass energy of $\sqrt{s} = 13$ TeV and the mean pile-up interaction multiplicity (number of simultaneous pp collisions) of $\langle\mu\rangle = 80$ were used. For each event, a list of all associated SPs and triplet seeds constructed at the combinatorial stage of ATLAS track seeding from the Run 2 geometry was given. SP information included its position in global cylindrical polar coordinates (r, ϕ, z), a detector layer index which the SP corresponds to, and a particle identifying barcode. The barcode information is only available for simulated data and was only used for the algorithm performance evaluation. Triplet seeds include indices of the SPs and a label indicating whether the seed was accepted ("Good Seed") or rejected ("Bad Seed") in the baseline algorithm.

2.1. Approximate Track Extrapolation

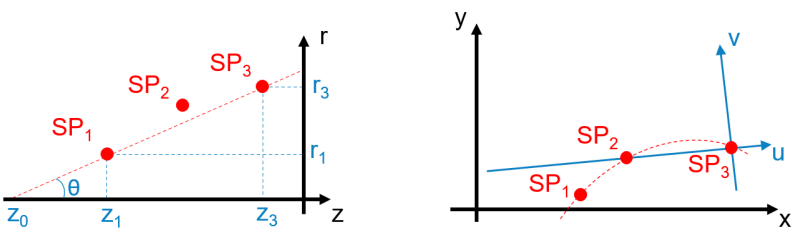
The first part of the algorithm involved extrapolating each triplet seed to find the approximate track trajectory and intersection points (IPs) with the detector layers. The simplified extrapolation was designed to be computationally cheap, while still being sufficiently accurate. The track path was defined in two different coordinate systems. The trajectory along the beamline was approximated in the zr coordinate system, where z-direction is parallel to the beamline and r is the perpendicular displacement from it. The trajectory in the plane perpendicular to the beamline was defined in the cartesian coordinates xy. The full trajectory can therefore be found with the $r^2 = x^2 + y^2$ relationship between the coordinate systems.

In the zr-plane, a straight line was fitted through the first and last SPs of the triplet seed (SP_1 and SP_3). It was defined by parameters z_0, the intersection of the track with the beamline, and $\tau = \cot\theta$, where θ is the track inclination angle in zr-plane. The trajectory in the xy-plane was approximated as a parabola. To reduce the number of parameters, an extra set of coordinates was defined: uv with the origin at the coordinates of SP_3 and u-axis passing through SP_2. The conversion between the two coordinate systems was then done by shifting the coordinate of SP_3 and rotating by an angle α between x- and u-axes. The visualised track fit in the zr- and xy-planes can be found in figure 1. Overall, the track was defined by the following equations:

$$r(z) = \frac{z - z_0}{\tau} \tag{1}$$

$$v(u) = au^2 + bu. \tag{2}$$

With the above track trajectories, the positions of the IPs with the detector layers were found, checked against the layer bounds and stored. The sequence of IPs was sorted by increasing distance from SP_3 and defined the trajectory for each triplet seed. For each IP in the track candidate, SPs in the vicinity were collected and stored for track filtering using their ϕ coordinate and a search window of ± 15 mm from IP in the detector layer's non-reference coordinate.



(a) (b)

Figure 1: Approximate track extrapolations in zr (a) and xy (b) coordinate systems. The dashed red line shows the fitted track. In (b), the rotated uv coordinate plane is shown in blue.

2.2. Track Seed Filtering and Classification

To reduce the computational complexity of the track filtering, a set of reference-related coordinates was defined. For each seed, the sequence of IPs was used as a reference trajectory, such that the new coordinate system was defined by $(s, \Delta) = (s, \Delta_x, \Delta_y)$, where s is the track path length and Δ_x, Δ_y are perpendicular deviations from the reference (IP). It is important to note that Δ_x, Δ_y do not define the same plane as the global xy. This way, the track was defined as an almost straight line and the need for modelling of the magnetic field in the later stages of the algorithm was eliminated. Schematic representation of the conversion to reference-related coordinates can be found in figure 2.

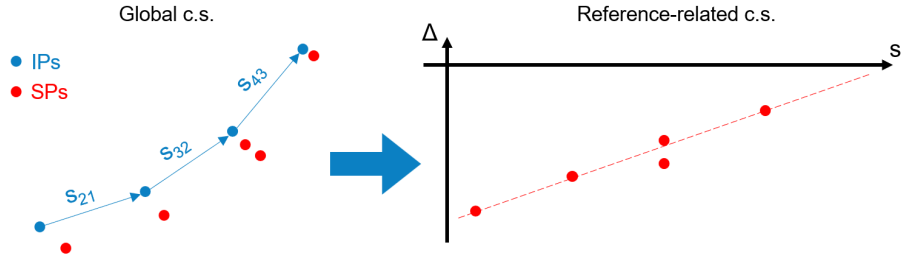


Figure 2: Schematic conversion of SP coordinates from global to reference-related coordinate system. Note that Δ is two-dimensional, but visualised as one-dimensional.

The track candidates and their collected SPs in the reference-related coordinate system were then passed through a combinatorial Kalman Filter (KF) [3]. The use of this coordinate system allowed for a linear propagation stage of the KF, further optimising the calculations. During the KF correction stage, SP positions were used as measurements, with each collected SP creating a new separate track candidate. To limit the number of candidates in the KF, only the 3 best tracks were kept after each iteration, based on the Track Quality feature that is described below. Measurement covariance was generated from the SP measurement uncertainties, while the noise covariance was created from the multiple scattering uncertainty using Moliére's formula [4].

Two features were generated for the classification during the KF. First, the Track Quality was calculated by classifying hits as successful or missed based on a cut on the χ^2 value (χ^2_{cut}). The track candidates were penalised for missed hits by subracting $\alpha_p \chi^2_{cut}$ and rewarded for successful ones by adding $\alpha_r \chi^2_{cut} - \chi^2$, where $\alpha_p = \alpha_r = 2.0$ were the penalty and reward coefficients. The second feature designed was Hole Value, which was generated by selecting the largest number of consecutive missed hits in a row of a given track, caclulating the total path length between these missed hits as a ratio to the total track path, and integrating it over the importance function $f(x) = x^2$. This placed higher importance on a large number of missed hits further into the track.

Based on the above features, a Support Vector Classifier with a polynomial kernel of order 2 was trained to predict whether a seed is rejected or accepted. It was tuned for a high acceptance of good seeds to limit the loss of track finding efficiency. The tuned model classification on the test $t\bar{t}$ dataset resulted in a Good Seed acceptance (True Positive) rate of 0.96 ± 0.02 and a Bad Seed rejection (True Negative) rate of 0.614 ± 0.010.

3. Performance Evaluation

3.1. Efficiency Comparison

Approximate track extrapolation and triplet seed preselections based on the classifier prediction were implemented as an independent step in the ATLAS HLT Athena framework [5], in between the Triplet Making and the Combinatorial Tracking stages of the FTF. To reduce

the computational overhead, the seed classification was stored as a lookup table (LUT) for fast inference during the online preselection.

Figure 3 shows the efficiencies with and without the added preselection for FTF full detector tracking in $t\bar{t}$ events with pile-up $\langle\mu\rangle = 80$ as a function of truth track η (left) and p_T (right). The average track finding efficiency loss due to the preselection algorithm is 0.7%, from 93.2% in the original FTF to 92.5% with the added preselection algorithm. The majority of the efficiency loss is found in the large $|\eta|$ region, which can be explained to be a result of material effects from the forward detector region, which are not accounted for in the algorithm.

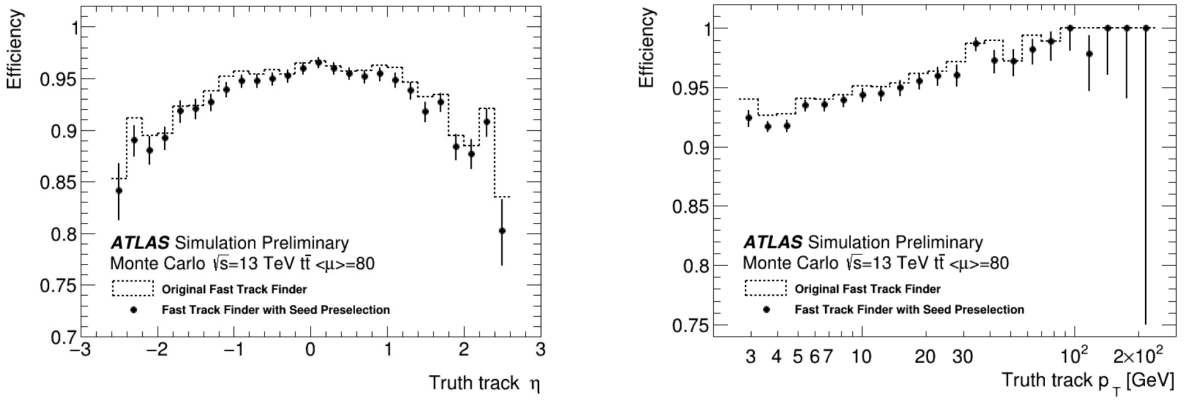


Figure 3: Track finding efficiency as a function of truth track η (left) and p_T (right) with and without the preselection of the track seeds algorithm added to the FTF. Error bars represent statistical uncertainties. ATLAS full detector tracking with $t\bar{t}$ $\langle\mu\rangle = 80$ [6].

3.2. CPU Time Comparison

Figures 4-6 show CPU time comparisons of different stages of FTF with and without the added preselection stage. The preselection algorithm timing is included in the Triplet Making stage, increasing the CPU time by a factor of 1.17 for $t\bar{t}$ full detector tracking with pile-up $\langle\mu\rangle = 80$. However, the added preselection reduces the number of triplet seeds processed in the Combinatorial Tracking stage, reducing the CPU time to 0.77 of that of the original FTF, indicating a speed up factor of 1.29. The Triplet Making and the Combinatorial Tracking stages make up 29% and 67% of the mean total time of the FTF, respectively, resulting in the preselection stage reducing the total mean time to a factor of 0.89 of the baseline algorithm, indicating a mean speed up of 1.12. This reduction of CPU time is expected to be greater for higher levels of pile-up.

4. Summary

The addition of the track seed preselection to the Fast Track Following provided a reduction in the CPU time to some of the stages of the algorithm. The applied preselection LUT resulted in a speed up factor of 1.12 at a cost of a negligible efficiency loss of 0.7% compared with the original FTF, at $\langle\mu\rangle = 80$ pile-up. This timing optimisation could be improved further by providing a different LUT, without the need to redesign the algorithm itself. With higher levels of pile-up, the effect of the optimisation is expected to be greater. This algorithm could be also adapted for the ITk geometry [7] to be used in the HL-LHC [8], where the levels of pile-up are expected to reach $\langle\mu\rangle = 200$.

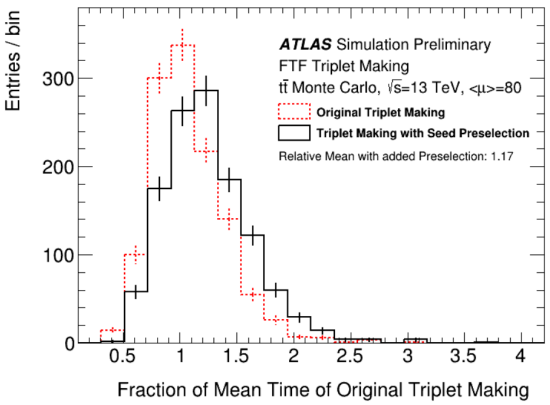


Figure 4: Mean CPU time of the triplet making stage of the FTF, with (black) and without (red) the preselection, scaled by a normalising factor such that the original FTF distribution's mean is 1. ATLAS full detector tracking with $t\bar{t}$ $\langle\mu\rangle = 80$ [6].

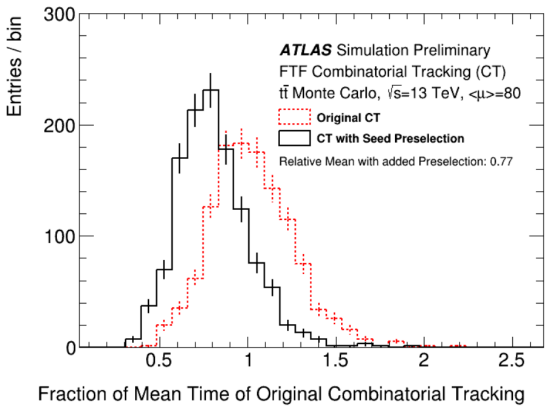


Figure 5: Mean CPU time of the combinatorial tracking stage of the FTF, with (black) and without (red) the preselection, scaled by a normalising factor such that the original FTF distribution's mean is 1. ATLAS full detector tracking with $t\bar{t}$ $\langle\mu\rangle = 80$ [6].

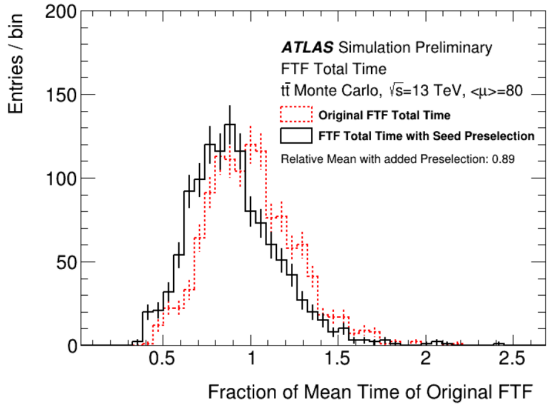


Figure 6: Mean CPU total time of the FTF, with (black) and without (red) the preselection, scaled by a normalising factor such that the original FTF distribution's mean is 1. ATLAS full detector tracking with $t\bar{t}$ $\langle\mu\rangle = 80$ [6].

References

[1] ATLAS Collaboration, *"The ATLAS experiment at the CERN Large Hadron Collider"*, 2008 *JINST* **3** S08003
[2] ATLAS Collaboration, *"The ATLAS inner detector trigger performance in pp collisions at 13 TeV during LHC Run 2"*, 2022 *Eur. Phys. J. C* **82** 206
[3] Frühwirth R, "Application of kalman filtering to track and vertex fitting", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 262, no. 2, pp. 444–450, 1987
[4] Zyla P et al., "Passage of Particles Through Matter" in *Review of Particle Physics*, **8**, vol. 2020, 2020, ch. 34, 083C01
[5] ATLAS Collaboration 2021 Athena
[6] *HLT Tracking Public Results Atlas Public TWiki*, en, `https://twiki.cern.ch/twiki/bin/view/AtlasPublic/HLTTrackingPublicResults`, Accessed: 2023-02-20.
[7] ATLAS Collaboration, "Technical Design Report for the ATLAS Inner Tracker Pixel Detector", 2017 *CERN*
[8] ATLAS Collaboration, "High-Luminosity Large Hadron Collider (HL-LHC): Technical design report", 2020 *CERN*