Abstract. The Large Hadron Collider (LHC) will be upgraded to High-luminosity LHC, increasing the number of simultaneous proton-proton collisions (pileup, PU) by several-folds. The harsher PU conditions lead to exponentially increasing combinatorics in charged particle tracking, placing a large demand on the computing resources. The projection on required computing resources exceeds the computing budget with the current algorithms running on single-thread CPUs. Motivated by the rise of heterogeneous computing in high-performance computing centers, we present Line Segment Tracking (LST), a highly parallelizeable algorithm that can run efficiently on GPUs and is being integrated to the CMS experiment central software. The usage of Alpaka framework for the algorithm implementation allows better portability of the code to run on different types of commercial parallel processors allowing flexibility on which processors to purchase for the experiment in the future. To verify a similar computational performance with a native solution, the Alpaka implementation is compared with a CUDA one on a NVIDIA Tesla V100 GPU. The algorithm creates short track segments in parallel, and progressively form higher level objects by linking segments that are consistent with genuine physics track hypothesis. The computing and physics performance are on par with the latest, multi-CPU versions of existing CMS tracking algorithms.

1 Introduction

The Large Hadron Collider at CERN will soon upgrade to the High-Luminosity LHC (HL-LHC), increasing the concurrent $pp$ collision in each bunch crossing, called pileup (PU), from an average of 70 to around 200. Many aspects of $pp$ collision event reconstruction will be complicated by the increased PU conditions.

Among the various event reconstruction algorithms, the charged particle tracking is one of the most significantly affected reconstruction algorithms. This is due to the current state-of-the-art charged particle tracking algorithms’ execution time growing exponentially with increased combinatorics of correlating hits in the tracker. This ultimately impacts the amount of computing resources needed for the CMS experiment to be successful. The current projection estimates that by 2029, the computing resources needed will exceed the available resources [1], with track reconstruction putting significant demands on CPU resources.
It is in this context that Line Segment Tracking (LST) was developed. The LST algorithm is a charged particle tracking algorithm that can easily be parallelized in order to mitigate the effect on the execution time due to the increased combinatorics. In addition, the improvements in CPU computing power has been stagnating for years while GPU computing power has steadily been going up. The LST algorithm aims to utilize the GPU resources for the future and was designed and implemented with the usage of GPU in mind from its conception.

2 Line Segment Tracking

2.1 Linking Hits in Outer Tracker and Building Tracklet Objects

The algorithm runs on the hits from the Outer Tracker (OT) of the Phase-2 CMS tracker [2]. Detector modules in Outer Tracker are \( p_T \) modules [2], which consist of two silicon layers closely placed in parallel with a gap of \( \approx 2-4 \) mm. This allows two hits from the same particle to be correlated locally, providing an opportunity to reduce the occupancy of the detector with local hit information. This first step of the LST creates mini-doublets (MDs) and is performed in parallel on GPU, with each thread performing the MD creation in each module.

The second step is to connect two MDs in different layers of the tracker to create line segments (LSs). To create LSs, a module map was created to compile a list of modules in the next outer layer of the tracker, compatible with a given module in the previous inner layer of the tracker. Then each GPU thread takes two modules in two different neighboring layers and performs LS building with MDs in each module.

After the LS building, the hits in the tracker that are to be connected are fully connected. No new connection between hits in the OT will be created. Therefore, the steps beyond LS will be pruning the connections and identifying a longer sequence of hits that represents a track from a single charged particle. The connection from OT to the Inner Tracker (IT) hits will be discussed in the later section.

To identify a longer sequence of hits, pairs of two LSs that share a common MD are considered. If the two LSs (i.e. total of 6 hits) sharing a common MD satisfy various geometric conditions that they are consistent with a hypothesis that the hits are originating from a single charged particle track, the two LSs are linked up and are called triplet (T3). Similarly, two T3s shared by a common MD, can be linked up to create a quintuplet (T5), and the T5s will contain 10 hits total.

2.2 Linking Track Objects in Inner Tracker to Outer Tracker

The objects described so far are all residing in the OT. However, there are also IT pixel seeds that can be connected to create longer tracks. Using a subset of IT pixel iteration seeds (i.e. initial iteration seeds, and highPtTriplet iteration seeds [3]), LST algorithm creates longer track objects through linking of OT objects with IT seeds. When a pixel seed and a T5 are linked, it is called a pixel quintuplet (pT5). When a pixel seed and a T3 are linked, it is called a pixel triplet (pT3). LST algorithm first creates pT5s, then uses the T3s not part of any T5s to create pT3s. This is to give priority to creating longer tracks. The full details of the algorithm can be found in Ref. [4].

2.3 Creating Track Candidate List

From the created objects, a list of Track Candidates (TC) is created by collecting a list of pT5, pT3, T5, and unused initial iteration seeds as track candidates. Priority is given to pT5,
pT3, T5, and unused initial iteration seeds, when creating the final TC set; i.e. in this priority order, subsequent objects are checked not to contribute to the previous.

It is important to note that LST objects are required to be consistent with the hypothesis that the $p_T$ of the object is greater than 0.8 GeV at various stages. For MD objects, the $p_T$ estimate will have a broader resolution, and for the higher-level objects with more hits, the $p_T$ resolution gets better. In the final performance, therefore, the efficiency curve is expected to show a turn-on effect around 0.8 GeV. Also, the OT does not extend beyond $|\eta|$ of 2.5, and the LST objects are not created or do not exist beyond $|\eta|$ roughly 2.4. Therefore, beyond $|\eta|$ of 2.4, the objects are mostly from IT only.

The algorithm is currently implemented using CUDA and has been tested on NVIDIA GPUs. Each of the steps only requires local neighboring information to link lower-level ingredients, and therefore the linking algorithm executions are massively parallelized on GPUs to accelerate the processing. We have currently implemented the workflow using CUDA and NVIDIA GPUs. We will use the Alpaka portability platform [3] in the future to make the algorithm device-agnostic.

3 Performance

3.1 Tracking efficiencies, fake rates, and duplicate rates

The performance of the charged particle tracking has been measured in two different setups. The first is a standalone setup where the hits and Inner Tracker seeds are used to create track candidate objects. The second is a setup where the algorithm is integrated into CMS Software (CMSSW), and the first two iteration seeds are consumed by LST to create track candidates. The first setup will measure the performance of the track pattern recognition step only and is useful in understanding the algorithm itself. The second setup is closer to a realistic scenario and is useful in understanding what the final physics performance at the HL-LHC may look like. In both scenarios, efficiencies, fake rates, and duplicate rates are measured. The performance results may differ between the setups due to different selections are applied to the final track candidates.

The efficiency is defined as the fraction of TC-matched simulated tracks from the hard scattering vertex with the following selections:

- $p_T > 0.9$ GeV
- $|\eta| < 4.5$
- production vertex radial distance (i.e. $r_{\text{vertex}}$) $< 2.5$ cm
- absolute value of production vertex z position (i.e. $|z_{\text{vertex}}|$) $< 30$ cm

The fake rate is defined as the fraction of TCs not matched to any simulated tracks. The duplicate rate is defined as the fraction of TCs matched to simulated tracks that are matched to multiple TCs. For fake rate and duplicate rate, the track parameter for a given TC is estimated with the pixel seeds’ track parameter if pixel seeds are available. If not available, a simple circle fit is done to obtain $p_T$, using the first hit position for $\phi$ estimate, and the last hit position to estimate $\eta$. For fake rate and duplicate rate definition, all simulated tracks are used for matching.

The standalone performance for all LST track candidates noted as “All LST objects” in Figure 1 shows good efficiency starting from around 0.9 GeV. The fake rate plot shows a fake rate around 10%. The duplicate rate is high in the high $|\eta|$ region, which is coming from IT seeds. In the near future, the duplicate rate is expected to be reduced when Patatrack seeds, which is another pixel track reconstruction algorithm on GPUs, are implemented [5].
Figure 1. Comparison of tracking performance measured in standalone setup: Efficiency as a function of tracking particle transverse momentum (left), fake rate (middle), and duplicate rate (right) as a function of tracking particle $\eta$ are shown. The performances of final track candidate lists created by the LST algorithm are shown in black. The breakdown of the performances by the sub-types of LST track candidates is shown in different colors.

Of particular interest is the tracking performance measured as a function of the production vertex of the particles shown in Figure 2. The different types of TCs that rely on IT seeds have a sharp cutoff as the production vertex is displaced from the beampipe. However, it can be clearly seen that efficiencies reach $\approx 60\%$ for the displaced tracks nearly exclusive from T5 contributions.

Figure 2. Efficiency as a function of tracking particle transverse production vertex position is shown. The performances of final track candidate lists created by the LST algorithm are shown in black. The breakdown of the performances by the sub-types of LST track candidates is shown in different colors.
So far, the discussion has been on the performance of the LST algorithm in a standalone setup. However, the TCs provided by the LST will be consumed by the track fitter in the later stage when integrated into the CMSSW workflow, and fake rates and duplicate rates are expected to change while the efficiencies are less affected. Therefore, the next performance studied is the performance from the workflow where the LST algorithm is integrated into the CMSSW reconstruction. The integrated the LST algorithm in the CMSSW reconstruction workflow finds the TCs and are passed onto the track fitting stage, and loose quality track selections (details in [3]) are applied to the fitted tracks to produce the final list of tracks from which the performances are measured. Three configurations were compared:

- **Baseline (all iter.):** Baseline tracking with combinatorial Kalman Filter (CKF) with all iterations.
- **Baseline (2 iter.):** Baseline with IT seeds from only the initial and highPtTriplet iterations.
- **LST (2 iter.):** LST with IT seeds from only the initial and highPtTriplet iterations.

The first configuration uses the conventional CKF-based tracking with all iterations of seeds and therefore is naively expected to have the best performance. The second configuration is similar to the first but uses only the first two iterations of seeds; this closely mimics the behavior of tracking in the High-level Trigger (HLT). The third configuration uses the LST algorithm presented in this note with the same first two iterations of seeds as the second configuration; therefore comparing the second and the third is more comparable. Figure 3 shows the efficiency as a function of $p_T$, fake rate, and duplicate rate as a function of $\eta$. The efficiency plots demonstrate that the LST (2 iter.) configuration shown in black performs similarly to the Baseline (2 iter.) configuration shown in red. The two configurations are comparable beyond $p_T$ of 1 GeV, which is largely due to implicit cuts applied in the LST configuration that require the objects to be consistent with $p_T > 0.8$ GeV. As expected, the Baseline configuration (all iter.) in blue shows the best possible performance. The fake rate has been found to be substantially lower in the LST configuration. The duplicate rate has been found to be higher. Future studies will investigate the tuning of configuration to reduce duplicate rates. Figure 4 shows the efficiency as a function of $r_{\text{vertex}}$. The result shows a very promising improvement over the other Baseline configurations at high $r_{\text{vertex}}$ phase-space. The LST configuration (2 iter.) is shown to perform better than the Baseline (all iter.) configuration showcasing a significant improvement in physics reach for long-lived particle research.

### 3.2 Timing

The timing was measured using A100 NVIDIA GPUs at the University of Florida. When one event is being processed at a time using the GPU, the timing measurement was measured to be around 18 ms. However, this can be improved by concurrently processing multiple events. Figure 5 (left) shows the average time it takes to process each event. The figure shows that the average time per event improves with more number of events being concurrently processed as the GPU utilization increases as more GPU kernel calls are being scheduled. The timing improves significantly up to 4 events, and the improvement plateaus beyond.

Figure 5 (right) shows the timing breakdown of the LST algorithm on GPU (NVIDIA A100). Due to limitations in the measurement method, the breakdown was measured only in the configuration where one event is processed concurrently. The percentage breakdown is ordered by the order of execution in the GPUs, and it can be seen that each step is generally equally distributed.
### Figure 3
Comparison of tracking performance measured in CMS Software (CMSSW) environment: Baseline with all iterations (blue), Baseline with 2 iterations (red), LST with 2 iterations (black) are shown. Efficiency as a function of tracking particle transverse momentum (left), fake rate (middle), and duplicate rate (right) as a function of tracking particle $\eta$ are shown.

### Figure 4
Efficiency as a function of tracking particle transverse production vertex position is shown. The LST algorithm result shown in black exhibits a significantly higher efficiency in tracking particles produced from displaced vertices, even compared to the full iteration result.

### 4 Summary
In order to address the computing challenges due to the increase in pileup at the HL-LHC, we presented the Line Segment Tracking algorithm. The performance is shown to be on par with the baseline approach while having a much lower fake rate and slightly higher duplicate rate which will be addressed in the following studies. The timing shows promising results of 18 ms per event and can be improved to less than 10 ms when running multiple events in flight.
The algorithm is currently on track to be integrated into CMSSW for main production in the near future. To run with GPUs other than the NVIDIA ones, the LST algorithm is going to be incorporated into CMSSW using Alpaka.

5 Acknowledgements

This work was supported by the U.S. National Science Foundation under Cooperative Agreements OAC-1836650 and PHY-2121686 and grant PHY-1912813.

References