GPU-based Online Tracking for the $\overline{\text{P}}$ANDA Experiment

Andreas Herten[1] *on behalf of the* $\overline{P}$*ANDA collaboration*

[1]Forschungszentrum Jülich, Wilhelm-Johnen-Straße, 52428 Jülich

DOI: http://dx.doi.org/10.3204/DESY-PROC-2014-05/11

The $\overline{\text{P}}$ANDA experiment is a new hadron physics experiment currently being built at FAIR, Darmstadt (Germany). $\overline{\text{P}}$ANDA will study fixed-target collisions of antiprotons of 1.5 GeV/c to 15 GeV/c momentum with protons and nuclei at a rate of 20 million events per second. To distinguish between background and signal events, $\overline{\text{P}}$ANDA will utilize a novel data acquisition technique. The experiment uses a sophisticated software-based event filtering scheme involving the reconstruction of the whole incoming data stream in real-time to trigger its data taking. Algorithms for online track reconstruction are essential for this task. We investigate algorithms running on GPUs to solve $\overline{\text{P}}$ANDA's real-time computing challenges.

1 The $\overline{\text{P}}$ANDA experiment

$\overline{\text{P}}$ANDA, short for Anti**p**roton **An**nihilation at **Da**rmstadt, is a new hadron physics experiment currently being built for FAIR, the Facility for Antiproton and Ion Research in Darmstadt, Germany. $\overline{\text{P}}$ANDA will deliver unprecedented insight into current topics of hadron physics. The experimental program includes meson spectroscopy in the charm energy region (charmonium; open charm), baryon production, nucleon structure, charm in nuclei, and searches for exotic states.

$\overline{\text{P}}$ANDA is an internal experiment at the High Energy Storage Ring (HESR) of FAIR. Antiprotons of 1.5 GeV/c to 15 GeV/c momentum collide with protons inside of $\overline{\text{P}}$ANDA in a fixed-target fashion. The resulting physical events are detected by $\overline{\text{P}}$ANDA's target spectrometer, located around the interaction region, and the forward spectrometer, dedicated to precision measurements of the forward-boosted event structure. [1]

Tracking detectors The innermost sub-detector of $\overline{\text{P}}$ANDA is the Micro Vertex Detector (MVD), a silicon-based detector directly around the interaction point. 10 million pixel and 200 000 strip channels enable vertex reconstruction with a resolution of $< 100\,\mu$m. The Straw Tube Tracker (STT), $\overline{\text{P}}$ANDA's central tracking detector, surrounds the MVD. The detector consists of 4636 small drift tubes of 1 cm diameter, aligned into 26 layers (18 parallel to the beam axis, 8 tilted by $\pm 1.5°$ with respect to the beam axis). Both the STT and MVD are located within the field of $\overline{\text{P}}$ANDA's 2 T solenoid. A third tracking detector covers forward angles around the beam line with gas electron multiplying (GEM) foils. Three GEM stations with two tracking planes each will be installed.

Online trigger In the investigated energy region, background and signal events have similar signatures and can not be distinguished easily. A conventional hardware-based triggering mechanism would not be efficient for $\overline{\text{P}}$ANDA. Instead, the experiment will use a sophisticated software-based online event trigger for event selection in realtime [1]. This process needs to reduce the raw data rate of 200 GB/s ($\approx 2 \times 10^7$ event/second) by three orders of magnitude to match the imposed limits by the storage facility (~ 3 PB/year) for further in-depth offline analysis.

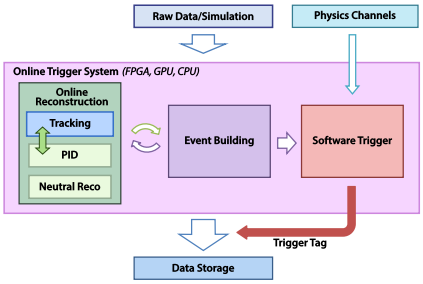

Figure 1: $\overline{\text{P}}$ANDA's online triggering system.

The online trigger of $\overline{\text{P}}$ANDA is structured as in Figure 1. In particular, online track reconstruction is a challenging part in terms of performance and efficiency, as a large number of computations are performed on a combinatorial complex datastream.

In this work, different tracking algorithms are investigated for their suitability to match the performance needed for $\overline{\text{P}}$ANDA's online reconstruction. The algorithms are implemented on GPUs, exploiting the low-cost, high-performance computing infrastructure offered by these devices. In addition to GPU-based tracking, event reconstruction using FPGAs is also investigated for $\overline{\text{P}}$ANDA [2], but not part of this report.

2 Online tracking algorithms

We are currently investigating three different algorithms for GPU-based track reconstruction. They are presented in the following.

2.1 Hough Transform

Hough Transforming is a method used in image processing to detect edges in images. It was first used for digitalization of images from bubble chamber experiments at CERN in the 1960s [3]. Adapting it for track finding and fitting means, to stay in computer vision terminology, to find edges (*tracks*) in images with a small number of pixels (*hit points*).

Method The method centers around transforming a point into a set of values in a parameter space (Hough space) and then extracting the most frequently generated parameters.

For every hit point (x_i, y_i), the equation

$$r_{ij} = x_i \cos \alpha_j + y_i \sin \alpha_j \tag{1}$$

is solved for a number of $\alpha_j \in [0°, 180°)$. r_{ij} is an equation of a line going through (x_i, y_i) avoiding possible poles. Every parameter pair (r_{ij}, α_j) is filled into a histogram. As more and more hit points of the same track are evaluated, a bin with the highest multiplicity emerges. This is the parameter pair equivalent to the line best connecting all hit points – the desired track parameter. See Figure 2.

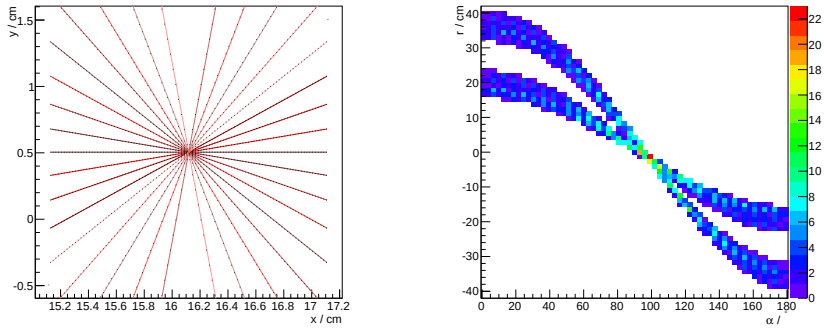

Figure 2: Schematic visualizations of Hough Transforms with coarse step sizes α for illustration. *Left*: Different lines going through a hit point (center) are sampled in 10° steps with Equation 1. *Right*: Hough space for a set of STT hit points, the α granularity is 2°.

Conformal map pre-step Since $\overline{\text{P}}$ANDA's central tracking detectors are enclosed in a solenoid magnet, the trajectories of particles are bent. A conformal mapping as a Hough Transform pre-step is used to convert curved tracks with hit points (x_i, y_i) into straight lines with hit points (x'_i, y'_i):

$$\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \frac{1}{x_i^2 + y_i^2} \begin{pmatrix} x_i \\ y_i \end{pmatrix}. \tag{2}$$

While conformal mapping involves only two computations per hit point, the actual Hough Transform calculates a large set of transformed points per hit – all independent from each other and hence very suitable to compute on a GPU. For example, an average event with 80 STT hits, evaluated every 0.2 °, leads to 72 000 computations, all independent from each other. The α granularity is here the main source for parallelism and subject to tuning. Its value is limited by detector measurement uncertainties on the one side and by available computing resources on the other.

Implementations Currently, two implementations are available for the Hough Transform. The first uses Thrust [6], the template library for CUDA, offering a big set of standard computing functions. No explicit CUDA kernel has been written for this implementation, only specialized operators were used. The benefit of this method is the ability to make use of the pre-programed sophisticated algorithms, optimized for high GPU performance. The drawback is an overhead of converting the data into Thrust-compatible data types, and the inflexibility when customizing the algorithms. With 3 ms/event, this implementation is also not at its performance maximum.

The second implementation is a plain CUDA implementation, built completely for this task. Its performance is six times better (0.5 ms/event) than the Thrust version. It is fitted for $\overline{\text{P}}$ANDA's exact task and is completely customizable, since it uses plain kernels throughout. Since this version is both faster and more flexible, we consider the plain CUDA implementation the better approach for Hough Transforms at $\overline{\text{P}}$ANDA.

The Hough Transform implementations give the expected results and fill the Hough space reliably with the data of possible tracks. (Figure 3, left). A further challenge, though, is peak

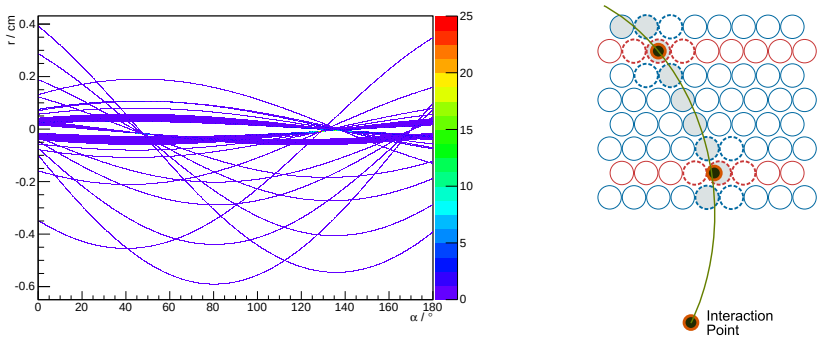

Figure 3: *Left*: Hough Transform of 68 hit points (STT, MVD). α is evaluated every 0.1°. Thrust is used for computation, ROOT [4] for displaying, and CUSP [5] for providing fast, templated methods to interface between both using the GPU. *Right*: Sketch of Triplet generation of the Triplet Finder.

finding in the Hough space: Large sets of hits create overlapping bands of parameter values in the Hough space. A complicated picture emerges to find peaks in – a simple threshold cut is not feasible. This part is currently under study.

2.2 Riemann Track Finder

The Riemann Track Finder is a track reconstruction algorithm already in use in PandaRoot, $\overline{\text{P}}$ANDA's software framework, since several years [7]. The basic principle of the method is a projection of two-dimensional hit points onto a Riemann surface (paraboloid) in a three dimensional space. There, a plane going through the mapped hit points can easily be fitted. The parameters of the plane are re-mapped into the two-dimensional and, with this, converted into track parameters. The method is based on [8].

GPU optimization As three points can always parameterize a circle, this is the minimum number of hits required for the Riemann Track Finder. A set of three hits (*seed*) is grown to a proper track candidate by subsequently testing additional hits against it and checking for passing certain quality cuts. The algorithm works on MVD hits. The seeding three-hit combinations are generated in the CPU version by three cascaded `for` loops, each one for a different layer of the MVD, implicitly assuming a track can not have two hits in the same layer. For running seed generation in parallel on the GPU, the serial loops are flattened out. Using explicit indexes, a 1 : 1 relation between global kernel variables involving CUDA's `threadIdx.x` and a layer hit combination is formulated.

As this produces a considerable amount of track seeds, the rest of the core Riemann Track Finder code is directly ported to the GPU. No further optimization has initially been done – the parallelism based on seeds suffices for a 100× performance increase when compared to the CPU version. Tracks of an event are found, on average, in 0.6 ms using a NVIDIA Tesla K20X GPU. The overhead needed for copying data to and from the device comprises 7 % of this time, making the Riemann Track Finder a computing-intensive algorithm and a good candidate for running on GPUs.

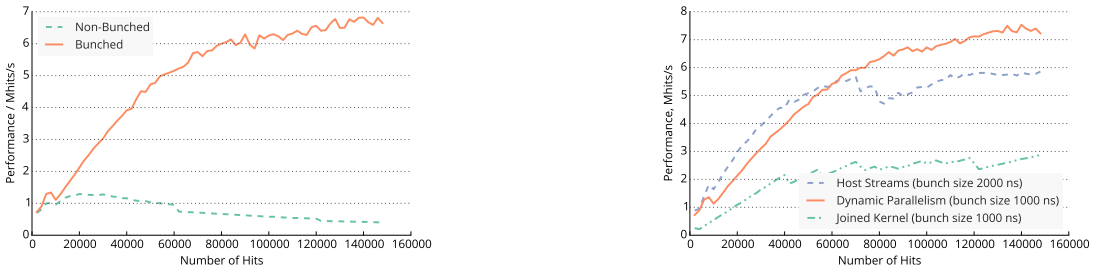

Figure 4: Performance for different optimization aspects of the Triplet Finder. Shown: Time needed to process a given number of hits, in million hits processed per seconds. *Left*: Comparison of Triplet Finder performance with and without bunching wrapper (optimal bunch size (1000 ns), dynamic parallelism approach). *Right*: Comparison of performance of different kernel launch strategies, invoked with respective ideal bunch sizes in bunching wrapper.

2.3 Triplet Finder

The Triplet Finder is an algorithm specifically designed for the $\overline{\text{P}}$ANDA STT [9]. It is ported to the GPU in collaboration with the NVIDIA Application Lab of the Jülich Supercomputing Centre.

Method Instead of evaluating data of the whole STT all the time, the Triplet Finder initially takes into account only a subset. Certain rows of neighboring STT drift tubes (*straws*) are selected for initiating the algorithm. As soon as a hit is detected in such a so-called *pivot layer*, the algorithm searches for additional hits in the directly neighboring straws. A center-of-gravity point is formed from all the hits, called a *Triplet*. Combining a Triplet from a pivot layer on the inner side of the STT with a Triplet from a pivot layer from the outer side of the STT with the interaction point at $(0, 0)$, a circle as a track candidate is calculated. In the next step, hits close to the track candidate are associated to it to eventually form a proper track, as shown in Figure 3, right. [10]

The Triplet Finder is a robust algorithm with many algorithmic tuning possibilities. Track candidates are calculated without relying on the event start time t_0, a value usually needed by algorithms to generate an isochronous hit circle around a straw's anode wire. t_0 is not known a-priori, as $\overline{\text{P}}$ANDA is running without any trigger information and needs to be provided by dedicated algorithms.

GPU optimizations A number of different GPU-specific optimizations are performed on the algorithm. Two of them are highlighted in the following, for a full list see [11].

Bunching wrapper The Triplet Finder looks at a set of hits at once and computes all possible track candidates. For a certain amount of hits, the algorithm reaches its peak performance. On the tested graphics card (NVIDIA Tesla K20X), this point is roughly equivalent to 25 000 hits (or 1000 ns STT data). To always invoke the algorithm with the number of hits at which it is performing best, a *bunching wrapper* is introduced. This wrapper cuts the continuous hit stream into sets (*bunches*) of sizes that maximize the occupancy of the GPU.

The algorithmic complexity is reduced and a performance maximum of 7 Mhit/s is reached (Figure 4, left).

Kernel launch strategies Different methods of launching the Triplet Finder processes are evaluated.

- **Dynamic Parallelism**: This method was used during the GPU development of the algorithm. Per bunch, one GPU-side process (*thread*) is called, launching itself the different Triplet Finder stages subsequently as individual, small kernels directly from the GPU.
- **Joined kernel**: One CUDA block per bunch is called on the GPU. Instead of individual kernels for the stages of the Triplet Finder (as in the previous approach), one fused kernel takes care of all stages.
- **Host streams**: Similarly to the first approach, the individual stages of the algorithm exist as individual kernels. But they are not launched by a GPU-side kernel, but by *stream* running CPU-sided. One stream is initiated per bunch.

Results in Figure 4, right, show the Dynamic Parallelism approach to be the fastest, as GPU occupancy is high and GPU-CPU communication reduced.

Optimization results The best performance in terms of computing time needed to process an event, which was reached with the aforementioned and further optimizations, is 0.02 ms/event (see also [11]). Employing only the Triplet Finder as a tracking algorithm, a multi-GPU system consisting of $\mathcal{O}(100)$ GPUs seems sufficient for $\overline{\text{P}}$ANDA.

3 Conclusions

Different algorithms have been presented for online track reconstruction on GPUs in the $\overline{\text{P}}$ANDA experiment. Each algorithm is in a different stage of development and has distinct feature sets specializing on different objectives. The most optimized algorithm is the Triplet Finder, with performance results making GPU-based online tracking a promising technique for $\overline{\text{P}}$ANDA's online event filter.
We are continuing tailoring the algorithms for $\overline{\text{P}}$ANDA's needs – optimizing performance and modifying specific aspects. All presented algorithms still need to be validated with physics cases of $\overline{\text{P}}$ANDA and benchmarked in terms of reconstruction quality. Also, further research is needed beyond the STT when including more sub-detectors into the different algorithms. Recently, high-throughput data transfer to the GPU is also subject of our research to complement the promising algorithmic developments towards a integrated GPU online track reconstruction system for $\overline{\text{P}}$ANDA.

References

[1] PANDA Collaboration. Physics Performance Report for PANDA: Strong Interaction Studies with Antiprotons. *arXiv:0903.3905*, 2009.

[2] H. Xu, Z.-A. Liu, Q. Wang, J. Zhao, D. Jin, W. Kühn, S. Lang, and M. Liu. An atca-based high performance compute node for trigger and data acquisition in large experiments. *Physics Procedia*, 37(0):1849 – 1854, 2012. Proceedings of the 2nd International Conference on Technology and Instrumentation in Particle Physics (TIPP 2011).

[3] Paul VC Hough. Machine analysis of bubble chamber pictures. In *International Conference on High Energy Accelerators and Instrumentation*, volume 73, 1959.

[4] Rene Brun and Fons Rademakers. Root—an object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1):81–86, 1997.

[5] Webpage: `http://cusplibrary.github.io/`.

[6] Nathan Bell and Jared Hoberock. Thrust: A productivity-oriented library for cuda. *GPU Computing Gems*, 7, 2011.

[7] Stefano Spataro and the PANDA Collaboration. The PandaRoot framework for simulation, reconstruction and analysis. *Journal of Physics: Conference Series*, 331(3):032031, 2011.

[8] R Frühwirth, A Strandlie, and W Waltenberger. Helix fitting by an extended riemann fit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 490(1):366–378, 2002.

[9] PANDA Collaboration. Technical design report for the panda (antiproton annihilations at darmstadt) straw tube tracker. *The European Physical Journal A*, 49(2):1–104, 2013.

[10] M. C. Mertens for the PANDA collaboration. Triplet based online track finding in the PANDA-STT. *Journal of Physics: Conference Series*, 503(1):012036, 2014.

[11] Andrew Adinetz, Andreas Herten, Jiri Kraus, Marius C Mertens, Dirk Pleiter, Tobias Stockmanns, and Peter Wintz. Triplet finder: On the way to triggerless online reconstruction with gpus for the panda experiment. In *Procedia Computer Science*, volume 29, pages 113–123. Elsevier, 2014.