Parallel Algorithms for Online Track Finding for the $\overline{\text{P}}$ANDA Experiment at FAIR

L Bianchi[1], A Herten[2], J Ritman[1], T Stockmanns[1] for the $\overline{\text{P}}$ANDA Collaboration

[1] IKP, Forschungszentrum Jülich, Germany
[2] JSC, Forschungszentrum Jülich, Germany

E-mail: `l.bianchi@fz-juelich.de`

Abstract. $\overline{\text{P}}$ANDA is a future hadron and nuclear physics experiment at the FAIR facility in construction in Darmstadt, Germany. Unlike the majority of current experiments, $\overline{\text{P}}$ANDA's strategy for data acquisition is based on online event reconstruction from free-streaming data, performed in real time entirely by software algorithms using global detector information. This paper reports on the status of the development of algorithms for the reconstruction of charged particle tracks, targeted towards online data processing applications, designed for execution on data-parallel processors such as GPUs (Graphic Processing Units). Two parallel algorithms for track finding, derived from the Circle Hough algorithm, are being developed to extend the parallelism to all stages of the algorithm. The concepts of the algorithms are described, along with preliminary results and considerations about their implementations and performance.

1. Introduction

1.1. The $\overline{P}$ANDA Experiment

$\overline{\text{P}}$ANDA (Anti-**P**roton **An**nihilation at **Da**rmstadt) is one of the key experiments at FAIR (**F**acility for **A**ntiproton and **I**on **R**esearch in Europe), currently in construction in Darmstadt, Germany. $\overline{\text{P}}$ANDA will detect collisions between a cooled antiproton beam ($\Delta p/p \sim 10^{-5}$) and a fixed target, in the momentum range 1–15 GeV/c. The $\bar{p}p$ initial-state configuration gives $\overline{\text{P}}$ANDA unique opportunities to study a wide range of topics in the area of Quantum Chromodynamics (QCD), including hadron spectroscopy, hypernuclei, and nuclear structure [1].
$\overline{\text{P}}$ANDA shares with many other experiments the challenge of studying rare physics processes in a collision environment dominated by background events. Due to the high luminosity necessary to accumulate sufficient statistics, an overwhelming amount of uninteresting background events, whose cross-section is many order of magnitude greater than the signal, is produced at the same time. The overall data rate coming from the detector greatly exceeds the offline storage capacity, so some form of real-time signal/background rejection needs to be applied.
The conventional strategy to reduce the incoming data is to perform a fast discrimination between signal and background events, based on partial information from a subset of the detector, implemented with custom hardware. If the event is judged to be signal-like, data from the whole detector are collected and stored offline for further analysis.
For $\overline{\text{P}}$ANDA, this approach is not feasible: due to the nature of hadronic interactions in the energy range of interest, background and signal events have very similar signatures, and

information from the full reconstructed event is needed to perform the selection. $\overline{\text{P}}$ANDA's strategy for data acquisition is instead based on an online event filtering scheme. Data from the whole detector is read continuously. An initial stage of event building is performed by FPGA-based compute nodes. Then, software algorithms perform full reconstruction of each event and background/signal identification based on multiple software trigger lines. Events passing the trigger selection are saved for offline storage. To match the available offline storage capabilities for data of approximately 1 PB/year to the incoming data rate from the detector of up to 200 GB/s, an online background rejection factor of up to about 1000 is required.

1.2. Online Track Finding Algorithms in the $\overline{\text{P}}$ANDA Reconstruction Chain

Online tracking is an essential step in the $\overline{\text{P}}$ANDA online event reconstruction chain. Tracking algorithms take discrete hit data from the tracking detectors, and process them to produce continuous particle trajectories. The tracking information is then used as input for all subsequent online event building and event selection phases. Online tracking is thus of crucial importance, since a) the tracking phase takes place at the earliest stages of the reconstruction and discrimination stage, so it directly affects all following steps; and b) it is among the most computationally intensive tasks, operating on a high volume of minimally processed data. For this reason, an optimal balance must be ensured between the physics requirement imposed by the reconstruction algorithms, and the computing resources available online. In the $\overline{\text{P}}$ANDA scheme, online track finding algorithms are primarily targeted for quick event building and selection. The primary performance parameter is the tracking efficiency, since, at this stage, tracks not reconstructed by the algorithms are lost. Parameters such as momentum resolution and track coverage, defined as the percentage of hits correctly associated with each tracks, are also important: track finding algorithms should produce tracks of sufficient quality to match the requirements of the downstream phases without any further refitting of the track. Once the event selection decision has taken place, more accurate (and thus more computationally demanding) algorithms can be applied on the events saved for offline analysis, given the greatly reduced data size and the comparatively less stringent limitations to the available processing time.

1.3. Parallel algorithm paradigms

In recent years, it is possible to observe a continuous shift towards data parallelism in most areas of computing, including high-energy physics (HEP). Due to, among other factors, the saturation of the increase in CPU clock frequencies, and radical improvements in the available technology, data-parallel processors such as multicore CPUs, Many-Integrated Cores (MICs) and general-purpose GPUs (GPGPU) are often regarded as the leading solution to increase performance and efficiency, while at the same time maintaining the versatility of general-purpose programming paradigms. The fundamental nature of HEP data, which can be divided into separate units to be processed independently, is naturally suited to the data-parallel model.

However, in addition to this coarse-grained parallelism between events, to fully exploit the potential increase in performance, the target of the algorithms should be to expose fine-grained parallelism *within* events: the unit of parallelism should be as finely grained as possible. This is especially true for GPUs, whose execution model requires thousands of threads running in parallel on hundreds of relatively low-powered computing cores. In addition, any operation that requires synchronization or shared information between parallelism units will most likely introduce performance penalties.

In the following, the emphasis is placed on parallelism at the algorithmic level, regardless of the actual implementation, and how operations that limit parallelism can be avoided or limited.

1.4. The $\overline{\text{P}}$ANDA Central Tracking Sub-detectors

The $\overline{\text{P}}$ANDA detector is composed of two main sections, the Target Spectrometer (TS) and the Forward Spectrometer (FS). The TS has a 2 T solenoidal magnetic field, whose axis coincides with the beam direction. The tracking sub-detector closest to the beam is the Micro Vertex Detector (MVD) [2]. The MVD is a solid-state detector composed of a combination of pixel detectors (10.3×10^6 channels) and double-sided silicon strips (2×10^5 channels), capable of a vertex resolution of < 100 μm. Surrounding the MVD is the Straw Tube Tracker (STT) [3]. The STT is made up of 4636 drift tubes, or *straws*, with a length of approx. 130 cm and a diameter of 10 mm. It has a spatial resolution of $\sigma_{xy} \sim 150$ μm, $\sigma_z \sim$ 2–3 mm. To reach this resolution, each hit is associated with an isochrone radius, calculated from an externally provided starting time t_0 from the drift time of electrons in the straw. Three Gas Electron Multiplier (GEM) [1] disks are located in the forward direction, downstream of the target, with about 35000 readout channels and a spatial resolution of < 100 μm.

2. Circle Hough Algorithm

General concepts The Circle Hough algorithm is based on the principle of the Hough transform, a technique for feature extraction used e.g. for detecting straight edges in a picture. In this method, an input dataset $\mathcal{D}$ is checked against a model $\mathcal{M}$, described as a function of parameters $(p_0, \ldots, p_N)$. The goal is to find the set of parameters $(p_0^*, \ldots, p_N^*)$ corresponding to the model with the best fit to $\mathcal{D}$. A number of instances of the model, the Hough elements, are generated for each component in the input dataset; their parameters are collected and the resulting parameter space, the Hough space, is analyzed. Points in the Hough space with the highest density, corresponding to the parameters occurring with the highest frequency in the input dataset, coincide with the model that fits best to the data. The phases of any Hough-like algorithm can thus be summarized as follows:

(i) Generation of elements
(ii) Accumulation of elements in the Hough space
(iii) Identification of most-likely values
(iv) Extraction of parameters

Application to track finding in $\overline{\text{P}}$ANDA In the case of track finding, the input dataset consists of detector hits within a Hough frame: a representation of all hits generated by tracks which are physically related, which can be event- or time-based. The feature to be extracted are the properties of the tracks that generated the hits: number, momentum vector, and hits associated to it. The model $\mathcal{M}$ are tracks that satisfy one of the possible track-hit compatibility condition. The elements are all possible tracks compatible with the hits in the frame.

Circle Hough original variant A version of the Hough transformation applied to tracks in $\overline{\text{P}}$ANDA is described in [4] and more in detail in [5]. Extensive tests show that the algorithm offers a good performance in terms of track reconstruction efficiency, and other track quality parameters. Furthermore, it is versatile, and can easily be applied to hits coming from different detectors. Nonetheless, in the version described in Ref. [5], only the generation of elements phase is performed in parallel. For this reason, the possibility of increasing the level of parallelism of the algorithm by extending it to all processing steps, while at the same time keeping or improving the track reconstruction performance, is being investigated.

Reducing the Hough space Additional assumptions on the models can be applied, reducing the number of both parameters and elements needed to describe the model, and thus both the dimensionality and the occupancy of the Hough space. First, only the transverse plane is considered: instead of its full helical trajectory, only its projection in the xy coordinate plane is considered. Additionally, the assumption of *primary track* is made: all tracks are considered originating from the interaction point (IP). Under these hypotheses, the track representation becomes a circle in the transverse plane, passing through the origin $(0, 0)$, and only two parameters are necessary to uniquely identify the a track: equivalently, the coordinates of its center in the transverse plane (the radius R is then fixed by the conditions of passing through the origin), or the radius R and the angular coordinate φ.

Hough conditions for different tracking detectors The nature of a hit point, and as a consequence its hit-track compatibility condition, depends on the detector which recorded the hit. For hits coming from the MVD or the GEM detectors, the hits are *point-like.* The track-hit compatibility condition is then reduced to whether the track passes through the hit point. In case of hits coming from the STT, each hit is associated with an isochrone radius. The track-hit compatibility condition is then satisfied if the track circle is tangent to the circle centered in $(x_h,\ y_h)$, and radius $r = r_h^{\text{iso}}$. Hits of this type are referred to as *extended.* Once the elements are generated, they can be accumulated together within the same hit frame even if they originated from hits of different detectors. The multiplicity $M(h)$ of tracks compatible with each hit also depends on whether the hit is point-like or extended. In case of extended hits, for each hit and radius value, there are two tracks (externally and internally tangent to the hit isochrone circle) for each of the two possible curvatures (corresponding to positive and negative charge), for a total of four tracks. For point-like hits, the external and internal branches are coincident, and the number of tracks is reduced to two.

Sampling the Hough space For each hit point, N elements are created, such as to be comparible with the hit according to the track-hit conditions. One possible choice is to treat one of the parameters as the sampling (independent) variable, define a set of N_{S} values, and calculate the corresponding elements from the track-hit compatibility condition. The radius R of the candidate track is chosen as the sampling parameter, while the azimuthal angle $\varphi = \varphi(R)$ is the calculated parameter. This choice of variables naturally takes into account the intrinsic φ symmetry of the problem, and offers an intuitive physical interpretation of the sampling parameter since for a primary track, $R \propto p_{\text{t}}$ (Fig. 1). The range of the sampling values is chosen so that smallest R_{min} is the minimum radius for a track compatible with the hit, while the largest R_{max} correspond to the maximum p_{t} physically possible for tracks. The operation is repeated for each hit in the input frame, for a total numer of elements of $N = M(h) \cdot N_h \cdot N_R$.

Collecting the Hough elements Once they are produced, all Hough elements in a frame must be collected and compared together within one instance of a Hough space. The simplest choice is to analyze a discretized representation in the form of an accumulator array, implemented as a D-dimensional histogram. [1] Although not strictly necessary, it is useful to assume that the accumulator array is filled evenly, i.e. every hit contributes exactly once to each cell of the accumulator array. Even if the bin size is optimally matched to the number of generated Hough elements, this is not enough to guarantee it, and explicit checks have to be included, reducing parallelism in this phase of the algorithm. Then, a peakfinding operation is performed. Once the peak values are detected, a post-processing phase takes care of removing or merging spurious peaks, and producing the candidate tracks from the sets of parameters corresponding to the peaks.

[1] Depending on the performance requirements and the target implementation, other choices for the accumulator array are also possible, e.g. space-partitioning data structures such as k-d trees.

3. Circle Hough Algorithm: Locus/Single Hit Variant

In this variant, like in the original version, the prototype element is a primary track in the transverse plane, compatible with one hit. The most relevant change is in the Hough element generation phase, but results in significant differences also in the subsequent phases of filling and scanning of the accumulator array. Instead of generating the elements directly for the track-hit condition, its properties are exploited to a deeper degree. Unlike the most general case of the Hough transform, an analytical expression describing the family of elements generated for each hit can be calculated. In particular, given a hit h, whose polar coordinates in the transverse plane [2] are $(\rho_h, \varphi_h, k_h = r_h^{\text{iso}}/\rho_h)$, the locus of track centers compatible with it can be written as:

$$\varphi(R, s, z) = \varphi_h + z \arccos\left(\frac{\rho_h}{2R}(1 - k_h^2) + s k_h\right) \tag{1}$$

where $s, z = \pm 1$ indicate the possible choices due to the track multiplicity. This corresponds to the equation of a hyperbola whose center lies on the midpoint between $(0, 0)$ and $(\rho_h,\ \varphi_h)$, aperture $\theta = k_h$, rotated of φ_h w.r.t. the center. In case of a point-like hit, $k_h = 0$. The hyperbola coincides with its conjugate axis and the expression becomes:

$$\varphi(R, z) = \varphi_h + z \arccos\left(\frac{\rho_h}{2R}\right), \tag{2}$$

corresponding to the equation of a straight line segment. A pair of rational Bezier curves[3] of degree 2 [6] are used to parametrize the hyperbola (Fig. 1).
The properties of Bezier curves can be leveraged to build an intrinsically regular accumulator array of arbitrary density. Rational Bezier curves can represent conic sections (such as hyperbolas) exactly, and they can be easily linearized into a polyline of straight line segments. Furthermore, Bezier curves are widely used in computer graphics, and several efficient algorithms operating on Bezier curves exist. A 2D accumulator array is equivalent to an image raster, so by knowing the analytical expression of the Bezier curves associated with each hit it is possible to perform the sampling and the collecting phase of the algorithm at the same time by rasterizing the curve over the accumulator array. In this case, it is guaranteed that each cell of the accumulator array is filled exactly once by each hit, without over- or under-filling, without having to match the element generation to the grid size, or any additional check on already filled values.

Two-dimensional hit/R grid In this variant, the same set of N_R sampling values $(R_0, \ldots, R_k)$ is used for all hits within a Hough frame. If polar coordinates $(\rho = R,\ \varphi)$ are used for the accumulator array, and the cell boundaries on the ρ axis are chosen so as to coincide with the sampling values, then the ρ cell of all elements is known by design. The accumulator array can then be treated as a series of N_R independent 1D arrays, each with a known constant occupancy of $M(h) \cdot N_h$. This offers the possibility of performing the fillling and peakfinding for each 1D array in parallel, and also facilitates the implementation of recursive, variable-width schemes in which smaller sampling/binning intervals are used until peaks are detected with sufficient precision. In addition, it is possible to choose array filling strategies so that it is guaranteed that only one cell of the accumulator array is being filled at the same time. This in principle ensures the thread-safety of the accumulator array filling at the algorithmic level, without resorting to implementation-dependent techniques (atomics, locks, ...) that might result in performance penalties. Finally, because of the more regular structure of the accumulator array, peakfinding is

[2] $\rho = \sqrt{x^2 + y^2}, \quad \varphi = \text{atan2}(y, x)$

[3] Bezier curves are parametric polynomial curves, identified by a degree N and $N + 1$ control points $P_0 \ldots P_N$. For Rational Bezier curves, an additional weight w_i is defined for each of the control points.

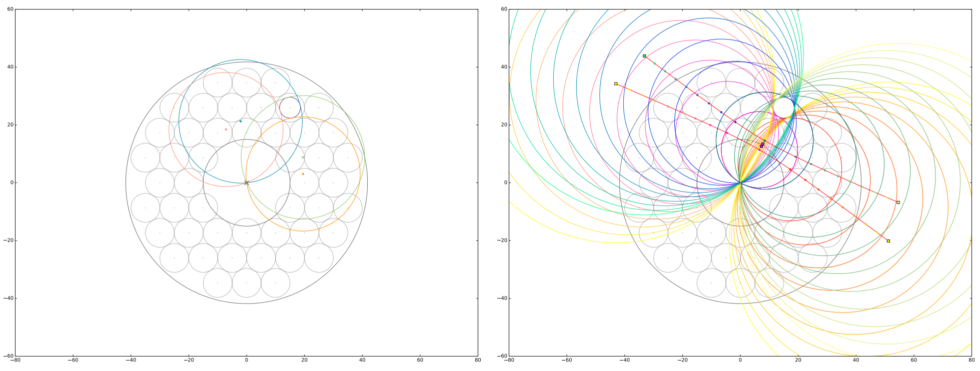

Figure 1: Circle Hough algorithm. (Left) All 4 possible tracks for a given hit and value of R. (Right) The same hit, showing Hough elements for a range of R values, and the Bezier curve representation of the locus of the element centers (red). The size of the straws has been exaggerated for clarity.

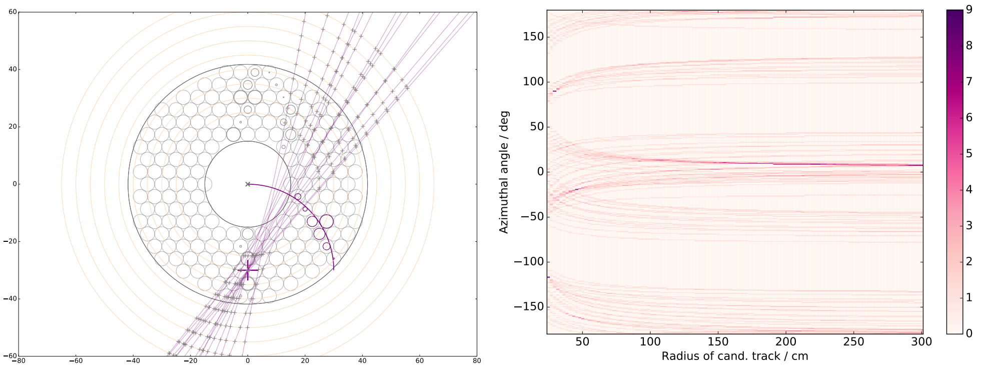

Figure 2: Circle Hough algorithm, grid version. (Left) Example Hough frame. One track and the hits originating from it are highlighted in purple. The radii corresponding to the common R sampling values are depicted in yellow. The loci for each hit, along with the elements calculated directly, are shown in light purple and grey respectively. The center of the true track, outlined with a cross, corresponds to the point where the majority of the loci intersect. (Right) Accumulator array for the depicted frame. The size of the straws has been exaggerated for clarity.

also simplified, with criteria based on a fixed threshold plus a requirement of relative maximum within nearest neighbors.

4. Circle Hough Algorithm: Hit Pair Variant

In this version of the algorithm, the prototype is a primary track compatible with a pair of hits. As in the single hit case, the total number of potential tracks depends on the hit type. For a pair of two point-like hits, only one track is possible. For a pair of two extended hits, there are four possible tracks, corresponding to all possible combinations of external or internal tangency. For the intermediate case of point-like plus extended hit pair, the number of possible tracks is two. The problem of finding such tracks is a special case of the problem of Apollonius [7], where the requirement is, given three non-intersecting circles, to find all possible circles tangent to them. In this case, one of the circles (C) is reduced to a point (P) in the origin, so the problem is reduced to finding the solutions for the cases PCC, PPC or PPP. The up to four solutions can

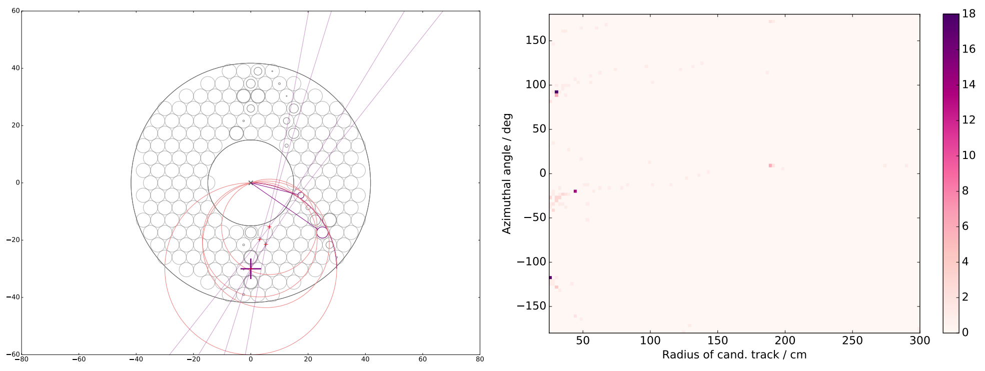

Figure 3: Circle Hough algorithm, hit pair variant. (Left) All possible tracks originated from an example hit pair. The big purple mark corresponds to the true track. In light purple, the track center loci for the two hits in the pair are shown. The center of the tracks lie on their intersections. (Right) Accumulator array for the depicted frame. The size of the straws has been exaggerated for clarity.

be found analytically, using e.g. the method of determinants. In this case, the track's radius or p_t is not an input parameter, but rather a result of the algorithm. An additional condition is placed upon the tracks found in this way to exclude those with unphysical parameters (Fig. 3).

Reduction of combinatorics For an input frame of h hits, there are $h(h-1)/2$ possible pairs. However, only a fraction of these will correspond to pairs of hits originated from the same track. The remaining pairs generated from spurious combinations of hits increase the computational cost, and at the same time pollute the Hough space by shadowing real pairs. To reduce the number of spurious combinations, a selection is performed, after the hits in the Hough frame are combined into pairs, and before the Hough elements for each pair are calculated. The criterion used for selection is the angular distance δ between hits in a pair: only pairs with $\delta < \delta_\mathrm{thr}$ are considered. For a fixed value of δ_thr, the list of possible pairs is static and only depends on the detector layout. It is possible to pre-compute this information using e.g. a lookup table within a pre-processing stage of the algorithm.

Accumulation As in the single hit variants, Hough elements for each pair are generated and processed in parallel, and then collected to a common accumulator array. The features of Hough elements generated from hit pairs result in a different structure of the accumulator array. Elements are spread out more uniformely outside of high-density regions, so simpler threshold-based peakfinding methods can be used (Fig. 3).

4.1. Implementation

The core development of the Circle Hough variants described in this paper is carried out outside PandaRoot [8], the complex software framework used for simulations within the $\overline{\mathrm{P}}$ANDA collaboration. A minimal framework was developed, containing only components which are essential for the operation of the algorithms. This was done in an effort to keep the dependencies separate, and increase the flexibility of the algorithm so that it can be tested and evaluated outside of $\overline{\mathrm{P}}$ANDA. Python was chosen for the development of the prototyping framework, due to its intuitive syntax and extensive availability of both scientific and general-purpose libraries. To improve the runtime performance, while simultaneously keeping the implementation consistent with a vectorized approach, most of the internals of the prototyping framework use the pandas [9] and Numpy [10] libraries. The algorithms are tested using data produced by PandaRoot

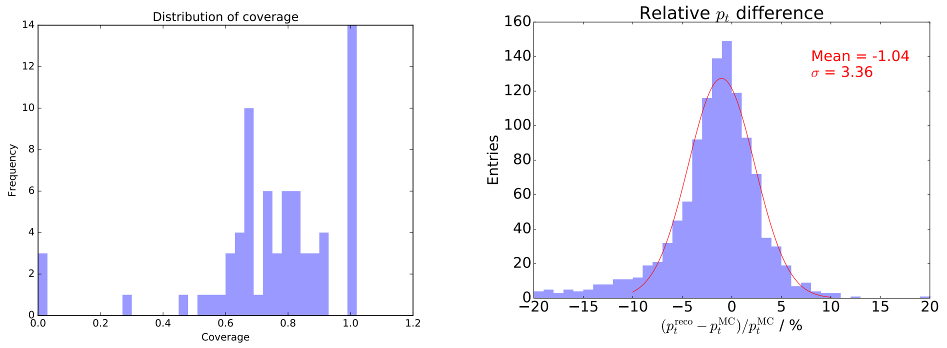

Figure 4: Preliminary results obtained by testing the algorithm with a series of tracks of various particle species, in a broad trasverse momentum range. (Left) Track coverage (Right) relative transverse momentum resolution.

simulations (Fig. 4). Interface modules are used to perform the conversion from and to the PandaRoot-compatible objects.
At the time of writing, an implementation of the algorithm in C++ is being completed and tested. This will be used as a baseline in measurements of computing performance, as well as being included in the PandaRoot codebase.

5. Summary and Outlook

In a continued effort for the online track finding at $\overline{\mathrm{P}}$ANDA, two novel variants of the Circle Hough algorithm were developed, building on the good performance of the existing algorithm in terms of efficiency, robustness, and versatility. For both variants, the instrinsic parallelism of the element generation phase of the original algorithm is improved and extended to subsequent phases.
Further development will focus on performance-optimized implementations targeting GPUs. At the same time, systematic testing will be performed, with the final goal of obtaining a full characterization of the parameter space of the algorithms in terms of both physics and computing performance.

References

[1] Lutz M *et al.* (PANDA) 2009 (*Preprint* `0903.3905`)
[2] Erni W *et al.* (PANDA) 2012 (*Preprint* `1207.6581`)
[3] Erni W *et al.* (PANDA) 2013 *Eur.Phys.J.* **A49** 25 (*Preprint* `1205.5441`)
[4] Bianchi L, Herten A, Ritman J, Stockmanns T, Adinetz A, Kraus J and Pleiter D 2015 *J. Phys. Conf. Ser.* **664** 082006
[5] Herten A 2015 *GPU-based Online Track Reconstruction for $\bar{P}ANDA$ and Application to the Analysis of $D \to K\pi\pi$* Ph.D. thesis Ruhr U., Bochum URL `https://juser.fz-juelich.de/record/255625/files/Thesis.pdf?subformat=pdfa`
[6] Weisstein E W Bezier curve From MathWorld–A Wolfram Web Resource URL `http://mathworld.wolfram.com/BezierCurve.html`
[7] Weisstein E W Apollonius' problem From MathWorld–A Wolfram Web Resource URL `http://mathworld.wolfram.com/ApolloniusProblem.html`
[8] Spataro S (PANDA) 2011 *J. Phys. Conf. Ser.* **331** 032031
[9] McKinney W 2010 Data structures for statistical computing in python *Proceedings of the 9th Python in Science Conference* ed van der Walt S and Millman J pp 51 – 56
[10] van der Walt S, Colbert S C and Varoquaux G 2011 *Computing in Science & Engineering* **13** 22–30