Proceedings of the CTD/WIT 2019
PROC-CTD19-105
October 25, 2019

Real-time Timepix3 data clustering, visualization and classification with a new Clusterer framework

Lukáš Meduna[1], Benedikt Bergmann[1], Petr Burian[1, 2],
Petr Mánek[1], Stanislav Pospíšil[1], Michal Suk[1]

On behalf of
[1] Institute of Experimental and Applied Physics
Czech Technical University in Prague, Czech Republic
and
[2] Faculty of Electrical Engineering
University of West Bohemia.

ABSTRACT

With the next-generation Timepix3 hybrid pixel detector, new possibilities and challenges have arisen. The Timepix3 segments active sensor area of 1.98 cm^2 into a square matrix of 256 x 256 pixels. In each pixel, the Time of Arrival (ToA, with a time binning of 1.56 ns) and Time over Threshold (ToT, energy) are measured simultaneously in a data-driven, i.e. self-triggered, read-out scheme. This contribution presents a framework for data acquisition, real-time clustering, visualization, classification and data saving. All of these tasks can be performed online, directly from multiple readouts through UDP protocol. Clusters are reconstructed on a pixel-by-pixel decision from the stream of not-necessarily chronologically sorted pixel data. To achieve quick spatial pixel-to-cluster matching, non-trivial data structures (quadtree) are utilized.
Furthermore, parallelism (i.e multi-threaded architecture) is used to further improve the performance of the framework. Such real-time clustering offers the advantages of online filtering and classification of events. Versatility of the software is ensured by supporting all major operating systems (macOS, Windows and Linux) with both graphical and command-line interfaces.
The performance of the real-time clustering and applied filtration methods are demonstrated using data from the Timepix3 network installed in the ATLAS and MoEDAL experiments at CERN.

PRESENTED AT

Connecting the Dots and Workshop on Intelligent Trackers (CTD/WIT 2019)
Instituto de Física Corpuscular (IFIC), Valencia, Spain
April 2-5, 2019

1 Introduction

Timepix3 [4] is the latest addition to the family of hybrid pixel detectors developed in the Medipix collaboration[*]. Timepix3 features an active are of $1.98\,\mathrm{cm}^2$ segmented into a square matrix of 256 x 256 pixels at a pixel pitch of 55 μm. Each pixel provides information about the energy deposition and the time of arrival (time binning 1.5625 ns). Its high temporal granularity allows measurements in high flux environments without risking track overlap and was successfully employed to perform 3D reconstruction of particle tracks [6, 7]. Recently, Timepix3 detectors have been installed in the ATLAS [1] and MoEDAL [9] experiments at CERN, where they provide valuable information about the radiation levels and composition of the radiation fields.

Whereas previous chips of Timepix technology [5] purely rely on a frame-based readout scheme, Timepix3 comes with a data-driven operation, allowing a (quasi-)continuous measurement (per pixel dead-time: 475 ns). In data driven mode, the output of the detector is a stream of individual pixel which can be processed almost in real-time. The aim of the presented work is to develop algorithms for fast track building to allow online visualization (for radiation level monitoring) and data filtering for pre-selection of relevant data to directly reject unwanted events and reduce disk space requirements. We have used the Katherine readout [3] to test the developed methodology and algorithms. However, the implemented methods are fully compatible with other Timepix3 readouts [8, 2].

2 Clusterer framework

2.1 Architecture

The presented framework is written in the modern C++ 14 language, which is suitable for high performance computations. The core of the framework is independent of the GUI[†] that is written in the Qt5 library[‡]. Aside from the quadtree [10] structure which bolsters the performance considerably, parallelism is used in the framework. Individual steps (i.e. reading input, parsing, clustering and visualizing) are separated into threads. Filtering of clusters is done using plug-ins compiled as dynamic libraries.

The framework was compiled, validated and tested on the major operating systems (Windows, macOS and Linux[§]) thus ensuring the versatility of the software. Both CLI[¶] and GUI versions were created.

2.2 Event building

Definition 1 (Cluster of pixels).
Let Δt be a positive real number called size of the time window, X be a set of pixels and let for each pixel $p \in X$, ToA_p denotes its time of arrival. A subset C of X is called cluster, if for each pair of pixels $p, p' \in C$ the following two conditions are satisfied:

1. *$|ToA_p - ToA_{p'}| < \Delta t$, i.e. the times of arrival of the pixels p and p' differ less than Δt* [‖]
2. *There exists a path of pixels in X from p to p', i.e. a sequence of 8-connected pixels in X which starts in p and ends in p'.*

Individual cluster events are recreated based on spatial and time configuration using the *ToA* and position in the sensor matrix (*x,y*) of pixel hits. Due to the technical implementation in the Timepix3 chip, the last 200 μs of pixels are not in chronological order.

[*] https://medipix.web.cern.ch/

[†] GUI - graphical user interface.

[‡] `https://doc.qt.io/qt-5/qt5-intro.html`

[§] Tested configurations – Windows 10 64-bit edition, macOS 10.14 (Mojave) and Linux Mint 19.0 (Debian based distribution).

[¶] CLI - command line interface

[‖] The value depends on the expected drift times of charge carriers and responses of pixel electronics. Conservatively, a value of 2000 *ns* was chosen.

2.2.1 Pixel processing

The goal is to transform partially unsorted hits from 2D space into the ToA sorted stream of separated events. The idea for processing incoming hits is given in the following.

Keep a set of partial clusters (denoted as *openClusters*). Try adding the new pixel to an existing cluster. If there is no existing cluster to which the new pixel can be added, create a new cluster consisting just of the current pixel. If the new pixel can be added to more than one cluster, join these clusters. This is achieved by Algorithm 1. If some of the partial clusters contains only pixels with ToA less than $ToA - 200\ \mu$s (the unordered time window) of the new pixel, the cluster is outputted and removed from the set of *openClusters*. Within these 200 μs, hundreds of clusters can be under construction.

Algorithm 1 Process one pixel

```
Require: $openClusters \leftarrow []$
 1: procedure ProcessPixel($timepixel$)          ▷ Pixel containing coordinates and ToA
 2:     $added \leftarrow False$
 3:     for each $cluster$ in $openClusters$ do
 4:         if CanBeAdded($cluster, timepixel$) then
 5:             if $added$ then
 6:                 $lastCluster \leftarrow$ JoinClusters($cluster, lastCluster$)
 7:             else
 8:                 AddPixel($cluster, timepixel$)
 9:                 $added \leftarrow True$
10:                 $lastCluster \leftarrow cluster$
11:     if not $added$ then
12:         $lastCluster \leftarrow$ CreateNewCluster($openClusters$)
13:         AddPixel($lastCluster, timepixel$)
14:     CloseAndDispatchOldClusters($openClusters$)
```

The structure *timepixel* is used to describe the tuple (x, y, ToA). The method *CanBeAdded* uses internal quadtree structure to determine if the pixel is adjacent (i.e. *8-connected*) to the cluster. Joining clusters in method *JoinClusters* is done by transfer of leaf nodes from the smaller tree to the bigger one.

For quick validation, if a pixel is adjacent to an open cluster, quadtree [10] was selected and implemented.

2.2.2 Quadtree

For logarithmic access time and low memory consumption per cluster, the quadtree data structure was selected to represent the neighboring pixels during the construction of separated events. The quadtree root node is representing a two-dimensional area. Each of four child nodes corresponds to one of the quadrants. At the end of branches, the leaf nodes are the minimal area of given size (called *leaf size*). The complexity is $\mathcal{O}(\log_4 n)$, where n is number of pixels in the root area.

In the framework's implementation, an advanced approach was used. To minimize the memory allocations, the tree is built from bottom up. Thus, for a single pixel the tree size is equal to the *leaf size* and it is increased when pixels outside the start area are added. To determine the optimal *leaf size*, the performance was measured on real data from a mixed radiation field in ATLAS. The results are shown in Figure 1. The size 16×16 was selected as a trade-off between performance and memory impact.

3 Results

With the real-time event reconstruction, multiple metrics of the radiation field can be observed in real-time. As an example, the cluster rates (radiation levels) and cluster energy spectrum as seen during a test beam measurement are given (Figure 3). The power (data reduction) of the real time data filtering is illustrated in Figure 2(a), where all events within a 2 s integration time are shown on the left and selected tracks of

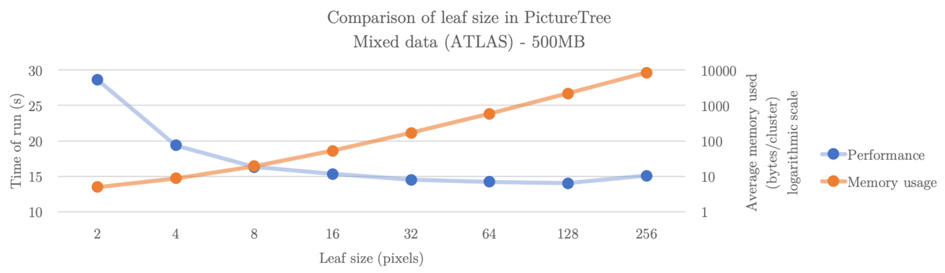
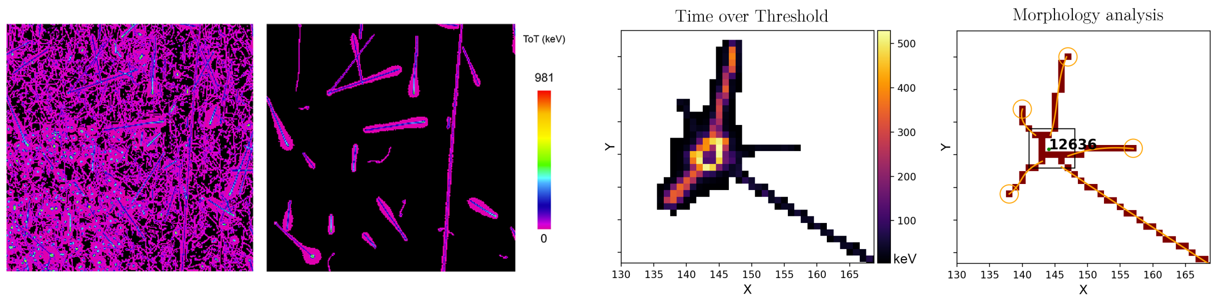


Figure 1: Performance test of different leaf sizes in real-world data. The memory usage is compared with the time to process the dataset.



(a) (b)

Figure 2: (a) Integration window of 2 s without (left) and with (right) filtering of events of interest; (b) Single cluster representing an inelastic interaction with the silicon nucleus of the detection medium (fragmentation reaction), which was selected and further evaluated (skeletonized). Five outgoing tracks (prongs) could be identified. Data were acquired by Timepix3 installed in ATLAS.

interest on the right. Proper filtering on track properties reduces the demand of disk space significantly without losing experiment relevant data. Pre-seleted Events of interest can then be further analyzed as shown in Figure 2(b), where the tracks of fragmentation products are identified by skeletonization.

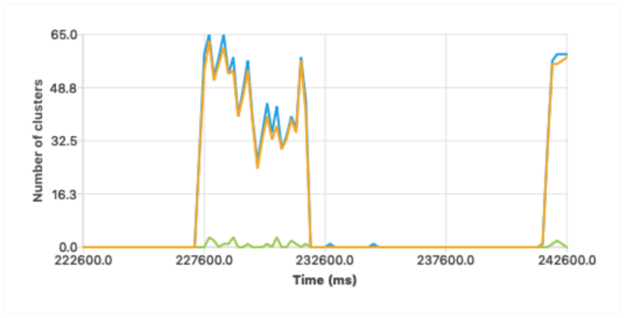
4 Conclusion

In this contribution, a comprehensive framework for data processing, event building and filtering on track features in the data-driven mode of Timepix3 was presented. Methods of fast parallel and reliable clustering of individual hits into separated events in real-time were developed. The software can be used both offline (the source is a file) and online (the source is a Timepix3 device). The clusterer offers a variety of online measurement statistics (e.g. energy spectrum, event intensity and integration live display), with support of user-defined filtering.

The software contains support for energy calibration and time-walk correction techniques [2, 6]. It can use the Katherine readout stream as direct data input. Modularity of the software allows interfacing different readouts or input sources. Online filtering on event properties was introduced as a powerful tool to reduce the amount of stored data for further analysis by regarding unwanted (background) events.

ACKNOWLEDGEMENTS

The authors would like to express their sincere gratitude to the Medipix2 and Medipix3 Collaboration for their permanent support. We would like to thank to RNDr. František Mráz, CSc. of the Faculty of Mathematics and Physics, Charles University, for his advices during the development of the framework.



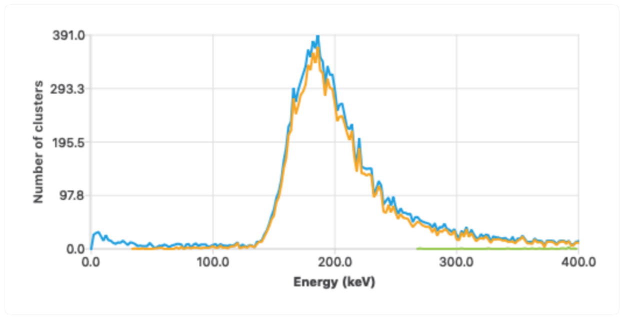


Figure 3: Real-time statistics of the measured radiation field. Cluster rate as a function of measurement time (left) and energy spectrum of the clusters during the measurement (right). The data were acquired during a test beam campaign in a relativistic particle beam at the Super-Proton-Synchrotron at CERN. The bottom image shows detail of the measurement while the filtering is applied. The blue line represents the raw input and the orange and green lines the filters based on class and size.

This work was supported by European Regional Development Funds: *"Van de Graaf Accelerator and Tunable Source of Monoenergetic Neutrons and Light Ions" (No. CZ.02.1.02 / 0.0 / 0.0 / 16 013 / 0001785)* and *"Research Infrastructure for Experiments at CERN" (LM 2015058).*

References

[1] P. Burian et al., Timepix3 detector network at ATLAS experiment, Journal of Instrumentation vol. **13** no. 11, C11024–C11024 (2018)

[2] D. Turecek et al., USB 3.0 readout and time-walk correction method for Timepix3 detector, Journal of Instrumentation vol. **11** no. 12, C12065–C12065 (2016)

[3] P. Burian et al., Katherine: Ethernet Embedded Readout Interface for Timepix3, Journal of Instrumentation vol. **12** no. 11, C11001–C11001 (2017)

[4] T. Poikela et al., Timepix3: a 65K channel hybrid pixel readout chip with simultaneous ToA/ToT and sparse readout, Journal of Instrumentation vol. **13** no. 11, C11024–C11024 (2014)

[5] X. Llopart et al., Timepix, a 65k programmable pixel readout chip for arrival time, energy and/or photon counting measurements, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment vol. **581** no. 1, 485 - 494 (2007)

[6] B. Bergmann et al., 3D track reconstruction capability of a silicon hybrid active pixel detector, The European Physical Journal C, vol. **77** no. 6, 421 (2017)

[7] B. Bergmann et al., 3D reconstruction of particle tracks in a 2 mm thick CdTe hybrid pixel detector, The European Physical Journal C, vol. **79** no. 2, 165 (2019)

[8] J. Visser et al., SPIDR: a read-out system for Medipix3 & Timepix3, Journal of Instrumentation, vol. **10** no. 12, C12028–C12028 (2015)

[9] B. Acharya et al., The physics programme of the MoEDAL experiment at the LHC, International Journal of Modern Physics A, vol. **29** no. 23, 1430050 (2014)

[10] Hanan Samet, The Quadtree and Related Hierarchical Data Structures, ACM Comput. Surv., vol. **16** no. 2, 187–260 (1984)