Free-Streaming Online Tracking in CBM

Sergey Gorbunov[1,*], *Sergei* Zharko[1], and *Valentina* Akishina[2,3]

[1]GSI Helmholtzzentrum fur Schwerionenforschung GmbH, Darmstadt, Germany
[2]FIAS Frankfurt Institute for Advanced Studies, Frankfurt am Main, Germany
[3]Johann Wolfgang Goethe-Universität Frankfurt, Frankfurt am Main, Germany

Abstract.
A free-streaming track reconstruction method has been developed for the CBM experiment at GSI. As a reconstruction core, it uses the existing single-event tracking based on the Cellular Automaton, but the approach to the free-streaming tracking is generic and can be can be used with various core algorithms without modification.
The method processes free-streaming data by manipulating the data at the input and output of the tracking core, leaving the reconstruction core unchanged. Data is fed into the tracking algorithm in small portions in a special way that avoids any track merging between the portions. A novel algorithm enables reading data portions without explicit time sorting, ensuring efficient processing.
The approach has been validated on simulated data at collision rates up to 10 MHz and was successfully applied online for the mCBM test setup during the March 2024 and February 2025 data-taking campaigns.

1 Introduction

The event reconstruction in the CBM experiment is a highly challenging task [1] [2]. There will be no simple hardware trigger due to the novel concepts of free-streaming data and self-triggered front-end electronics. Thus, raw data has no prior association with physical events. CBM is designed to operate at interaction rates of 10 MHz – an unprecedented case for heavy-ion collisions. At this rate, collisions overlap in time and are to be resolved in software by reconstruction algorithms. These challenges make the high speed and quality of data reconstruction crucial.

The core of the track reconstruction is the Cellular Automaton-based algorithm [3] used for the Silicon Tracking System. It digests free-streaming data both online and offline, taking large time slices of the hit measurements as input with no a-priori-defined physical collisions. Each time slice is reconstructed in small portions by applying a non-merging sliding window algorithm, which achieves a nearly constant event processing time, regardless of time slice duration and collision rate.

2 No-merging sliding window algorithm

The tracking component receives the data in large time slices that contain up to a few seconds of continuously recorded data. As no hit-to-event association yet exists, the tracking should process the entire time slice.

*e-mail: se.gorbunov@gsi.de

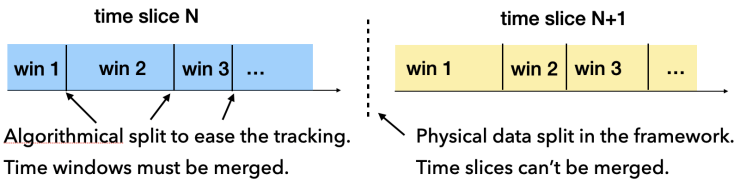

Figure 1. Data split in time: time slices in the framework and time windows within the time slice

Our approach enables free-streaming data reconstruction without modifying the core single-event tracking algorithm. Instead, we manipulate the data at a higher level.

CBM utilizes a Cellular Automaton-based tracking algorithm [3] for single-event reconstruction. However, since our method is independent of the core tracking algorithm, we do not discuss its specifics here.

To process a time slice, we split it into small intervals called time windows (Fig.1) and run the single-event tracking within each window. The time windows are processed sequentially.

Some tracks on the border between the time windows may be split and reconstructed only partially in the adjacent windows or missed entirely. To avoid merging these parts, we simply discard partially reconstructed tracks in the earlier time window and transfer their associated hits to the next time window. This ensures that these tracks are fully reconstructed in the subsequent window.

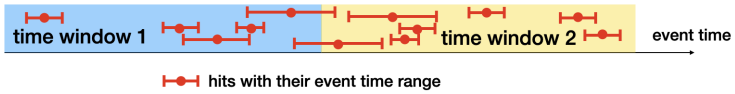

Figure 2. Hit event time range overlapping with two time windows

Let us consider the approach in detail:

- First, for each hit, the possible time interval of the corresponding collision event is determined. The interval is determined by the distance to the target, the velocity range of the particles of interest, and the detector?s time resolution.
- Then, the input time slice is divided into short, non-overlapping time windows (Fig.1). The window lengths are subject to optimization. Currently, all windows have the same predefined length; in future implementations, we plan to adjust window lengths dynamically based on data density, ensuring each window contains a comparable number of hits.
- Next, we assign hits to their respective time windows. If a hit?s time interval overlaps two adjacent windows, it is assigned to both (Fig. 2). In the extreme case, a hit can belong to many windows when its time resolution is very poor, and the windows are relatively short. This situation is undesirable, but technically, the algorithm can still handle this scenario.

- Next, we perform tracking window by window. After finding tracks in the time window, we keep only those tracks that contain at least one hit that does not belong to the next window. Tracks that are fully contained within the next window are discarded, as they will be reconstructed again in the following window with potentially more hits.

The above algorithm enables continuous data stream track reconstruction while avoiding the need for track merging at time window boundaries. The only minor drawback is the redundant processing of data near window boundaries, introducing a slight computational overhead. However, since there are only a few such tracks, this overhead is negligible.

The single-event tracking is used as it is, with only one modification. While the time window data is processed as a single large event, the hits that do not match in time are not allowed to be linked into the same track.

An additional advantage that comes for free is the natural usage of multithreading. We simply process time windows in parallel with independent threads handling different sections of the input time slice.

We call the presented method the "sliding window" approach, as we reconstruct the data in a short time window that moves in time along with the data stream.

3 Efficient data access

As our method processes time windows sequentially, we must read the input data in time portions. To achieve this, we developed a special algorithm that efficiently retrieves all hits within a given time window without accessing irrelevant data or requiring prior sorting. It is based on the assumption that the input data is already ordered in time on a large scale with a slight local disorder due to the readout features. If this assumption is violated, the algorithm will still function correctly but with reduced speed.

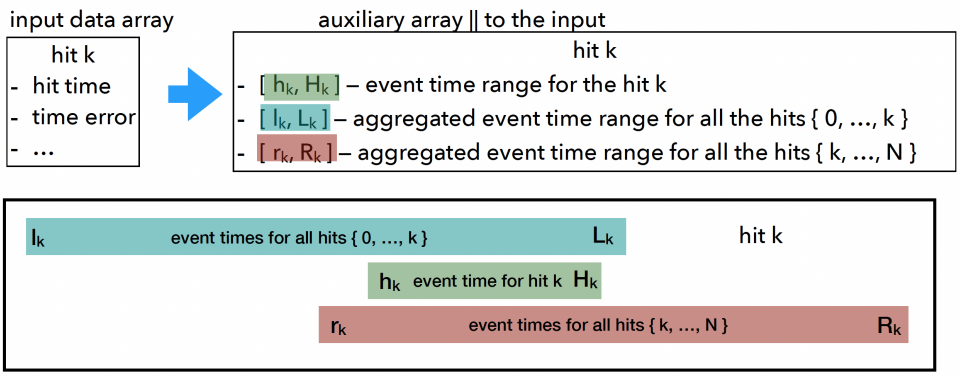

Figure 3. Auxiliary data to arrange input hits in time

For each hit k in the input data array, we create an auxiliary record that contains six values: $[h_k, H_k]$ — the time range of the event corresponding to the k-th hit, $[l_k, L_k]$ — aggregated time range for all the hits to the left – from the beginning of the array to the hit k, $[r_k, R_k]$ — aggregated time range for all the hits to the right – from the hit k to the end of the array (Fig.3). The auxiliary records are constructed in two linear passes over the data. This procedure does not involve any combinatorics.

With the auxiliary data array in place, retrieving hits that overlap with a given time range $[t, T]$ becomes straightforward – we need to:

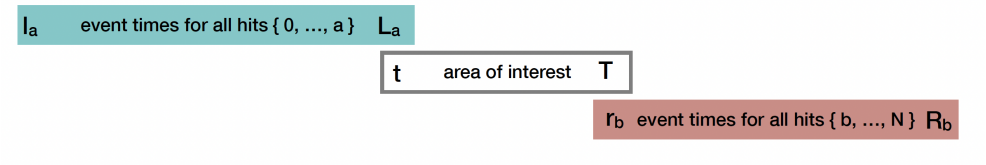

Figure 4. Finding hits that belong to the given time interval

- start at the first hit **a** satisfying the condition ($t <= L_{\mathbf{a}}$)
- stop at the last hit **b** satisfying the condition ($r_{\mathbf{b}} <= T$),

as illustrated in Figure 4.

At the start, we find the first and the last hit indices for the first time window and then move the two pointers when moving the time window. If the data is perfectly sorted in time, the algorithm will read it continuously without overhead. The more unsorted data is, the more time it will take to read it.

4 Performance on simulated data

Track type	E-by-E	1 MHz	10 MHz
Long high-p primary	100.0%	100.0%	100%
All tracks	94.9%	94.8%	94.8%
Primary high-p	98.2%	98.2%	98.1%
Primary low-p	95.4%	95.3%	95.3%
Secondary high-p	95.0%	94.7%	94.9%
Secondary low-p	75.8%	75.4%	75.7%
Clone ratio	0.5%	0.5%	0.6%
Fake ratio	0.4%	0.5%	0.5%
True hits assigned to reco track	96.1%	95.9%	95.3%
Reco time / event	26 ms	28 ms	30 ms

Table 1. Reconstruction efficiency with continuous readout depending on the collision rate. Reconstructable track: crosses at least four consecutive tracking layers, $p > 0.1$ GeV. A track has low momentum when $p < 1$ GeV; otherwise, it is classified as high-p. A long track has hits in all detector stations. Clone track: more than one track obtained for the same simulated particle. The track is reconstructed when the purity of its hits (the percentage of correctly assigned hits) is greater than 70%. Otherwise, it is classified as a fake track.

The presented algorithm demonstrates stable efficiency on the simulated data up to 10 MHz collision rate (Table 1). The reconstruction time per event also remains stable: at a 10 MHz collision rate, the free-streaming reconstruction is only 15% slower than the event-by-event reconstruction. It is important to note that the current tracking implementation is temporarily deoptimized due to ongoing intensive development; the optimized version is expected to be approximately four times faster.

The performance was evaluated using 1000 simulated AuAu UrQMD minimum bias events at 10 AGeV. Since we focused solely on measuring the scalability of the tracking algorithm, no channel dead time or other rate-dependent effects were included in this test.

5 Free-streaming tracking in the mini-CBM demonstrator

The "mini-CBM" (mCBM) demonstrator was assembled and placed on the SIS18 accelerator (GSI, Germany) in 2019 [4]. The mCBM setup consists of reduced detector subsystems, which will be used in the full CBM and serves as a good opportunity to test the hardware and software components in real data-taking conditions.

A key difference in the tracking configuration of mCBM compared to the full CBM is the absence of the magnetic field. Because of that, all the tracks become straight lines; their momentum can no longer be measured within the tracking procedure. Another feature is that the tracking setup is not uniform. Many detector subsystems with different technological aspects and acceptance coverage were used as tracking detectors.

The data-taking in mCBM also uncovered a problem of dead and noisy readout channels, which led to gaps in the acceptance of the tracking stations. This was solved by allowing one or multiple tracking stations to be skipped when constructing the track.

Since 2024, the Cellular Automaton tracking has participated in the online data processing stack [5], which performed successfully in the nickel (May 2024) and silver (February 2025) runs of mCBM. Here, the tracking is first performed on the entire time slice to define time seeds for event building. Then, it runs again on each event to form a physics event selector. In the first free-streaming mode, the tracking algorithm is parallelized over multiple CPU threads with the help of the STD thread library. Contiguous time windows are grouped so that the total number of hits in each group is approximately the same. Each group of time windows is then processed on the individual CPU thread. The multi-threading gives an almost linear boost of $0.75N_{\text{threads}}$ for the whole tracking component execution [6].

References

[1] V. Friese for the CBM collaboration, J. Phys. Conf. Series **898** 112003 (2017)

[2] V. Friese for the CBM collaboration, EPJ Web Conf. **226** 01004 (2020)

[3] V. Akishina and I. Kisel, Time-based Cellular Automaton track finder for the CBM experiment, J. Phys. Conf. Series **599** p.1 012024 (2015). https://doi.org/10.1088/1742-6596/599/1/012024

[4] 220072, mCBM@SIS18, CBM (CBM Collaboration, GSI, Darmstadt, 2017), https://repository.gsi.de/record/220072

[5] J. de Cuveland, D. Emschermann, V. Friese, I. Fröhlich, P. Gasik, D. Hutter, W. Müller, C. Sturm (CBM Collaboration), Technical Design Report for the CBM Online Systems – Part I, DAQ and FLES Entry Stage, CBM Technical Design Report (GSI Helmholtzzen- trum für Schwerionenforschung GmbH, Darmstadt, 2023), https://repository.gsi.de/record/340597

[6] S. Gorbunov, S. Zharko and V. Akishina, Cellular Automaton tracking for the mCBM experiment (CBM Progress Report 2023, Darmstadt, Germany), p.144-147, https://doi.org/10.15120/GSI-2024-00765