EPJ Web of Conferences **127**, 00003 (2016) DOI: 10.1051/epjconf/201612700003
Connecting The Dots 2016

4D Cellular Automaton Track Finder in the CBM Experiment

Valentina **Akishina**[1,2,a] and Ivan **Kisel**[1,2,b]

[1] *Goethe University, Frankfurt am Main, Germany*
[2] *Frankfurt Institute for Advanced Studies (FIAS), Frankfurt am Main, Germany*

Abstract. The CBM experiment (FAIR/GSI, Darmstadt, Germany) will focus on the measurement of rare probes at interaction rates up to 10 MHz with data flow of up to 1 TB/s. It requires a novel read-out and data-acquisition concept with self-triggered electronics and free-streaming data. In this case resolving different collisions is a non-trivial task and event building must be performed in software online. That requires full online event reconstruction and selection not only in space, but also in time, so-called 4D event building and selection. This is a task of the First-Level Event Selection (FLES).
The FLES reconstruction and selection package consists of several modules: track finding, track fitting, short-lived particles finding, event building and event selection. The Cellular Automaton (CA) track finder algorithm was adapted towards time-based reconstruction. In this article, we describe in detail the modification done to the algorithm, as well as the performance of the developed time-based CA approach.

1 Introduction

The CBM experiment [1] is targeted to explore the QCD phase diagram in the region of high baryon densities by investigating nuclear collisions from 2 to 45 AGeV (per nucleon) beam energy. The CBM detector is designed to measure rare diagnostic probes such as multi-strange hyperons, charmed particles and vector mesons decaying into lepton pairs with unprecedented precision and statistics. In order to perform measurements with the required precision, the experiment is being designed to operate under interaction rates of up to 10 MHz, which puts strong constraints on the data processing. Together with the high multiplicity of charged particles produced in heavy-ion collisions — up to 1000 particles in central collisions — this leads to huge data rates of up to 1 TB/s. The data acquisition is thus being designed in a free-running fashion, without a hardware trigger. Event reconstruction and selection will be performed online on a dedicated many-core CPU/GPU computer farm.

The traditional latency-limited trigger architectures, typical for conventional experiments with a hardware trigger, are not suited to the case of CBM. Instead, the experiment will ship and collect time-stamped data into a readout buffer in the form of a time-slice of a certain time length, not in the form of isolated collisions, and deliver it to a large computer farm, where online event reconstruction and selection will be performed. In this case a considerable fraction of collisions inside a time-slice may potentially overlap in time with each other. The reconstruction algorithms in this case should be

[a] e-mail: V.Akishina@gsi.de
[b] e-mail: I.Kisel@compeng.uni-frankfurt.de

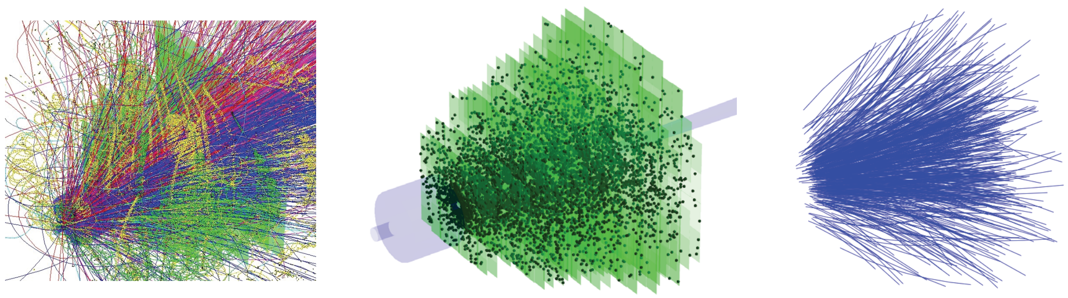


Figure 1. An illustration of the complexity of the track finding problem. From left to right: particle trajectories in a simulated central Au+Au collision at 25 *A*GeV energy in the CBM experiment; only hits in the main tracking detector as the input information for track finding; the reconstructed tracks found by the CA track finder. For the simulation of the events the URQMD generator was used in conjunction with GEANT3 for the transport of the generated particles through the detector setup.

modified in order to process not event-associated data. Association of hit information with physical events must be performed in software and requires full online event reconstruction not only in space, but also in time, so-called 4-dimensional track reconstruction.

2 Cellular automaton based track finder

Since one of the most challenging and time-consuming part of collision reconstruction is track finding, optimization of a track finder algorithm has great impact on the computational effort needed for the data-analysis chain (see Figure 1). The reason for that is that the track finder usually takes as an input raw detector hit measurements at the very first event reconstruction phase, when no data reduction can be done yet. Therefore one of the major obstacles to be solved by each track finder is a huge and fast growing with track density amount of combinations, which the algorithm has to consider in order to bind together one- or two-dimensional measurements into five-dimensional tracks.

The exponential growth of the combinatorial enumeration at high track densities usually makes it impossible to consider all combinations within a reasonable time. A solid solution for combinatorial optimization is provided by the Cellular Automaton (CA) approach. The CA track finder features make the algorithm an appropriate solution for track reconstruction in the main tracking detector of CBM, the Silicon Tracking System (STS).

The CA method profits from building up so-called cells, i.e. short track segments (triplets in the case of CBM) with a higher dimensionality than measurements, before starting the main combinatorial search. These cells are combined into reconstructed tracks during the CA track finder evolution — the process of defining neighboring cells, which with high probability belong to the same track. The rules of CA evolution in this case are given by the definition of the neighboring segments, e.g. in CBM that is the triplets, which coincide in the position and momentum within a certain precision.

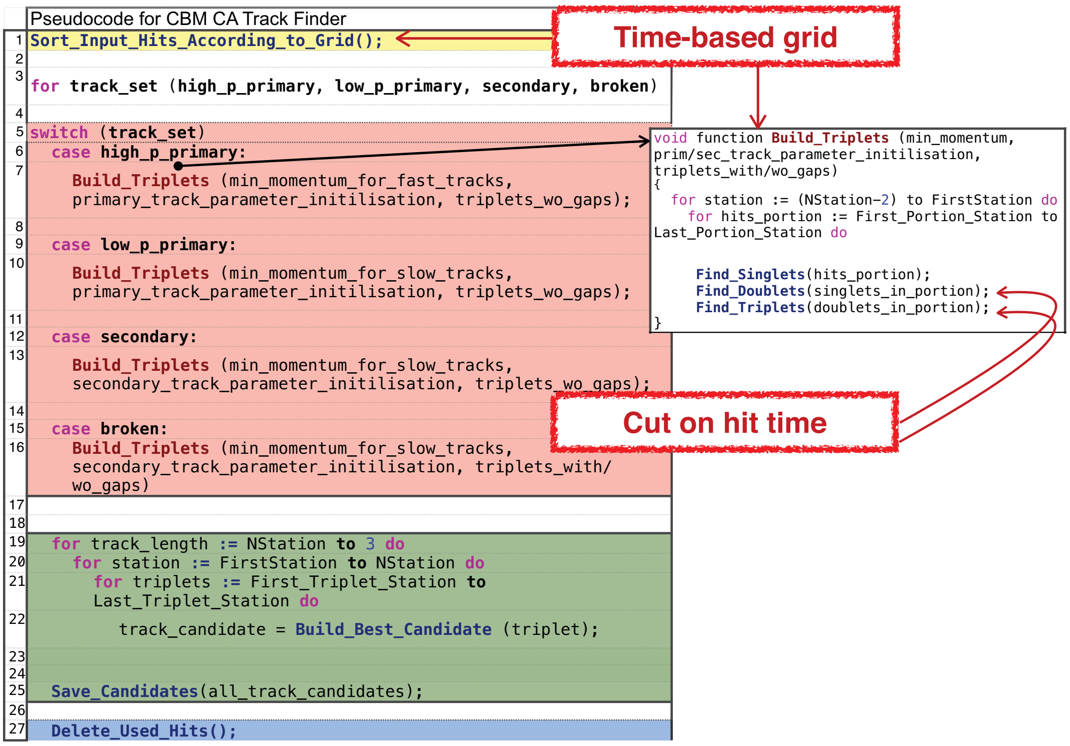
The algorithm is intrinsically local, working with data only in a neighborhood. It provides a steady consolidation of the track information identified in the course of the algorithm evolution. In addition, the CA based algorithms can be parallelized in order to be implemented on many-core CPU/GPU computer architectures. Let us consider some details of the CBM CA track finder algorithm.

2.1 CA track finder algorithm strategy

A charged particle track in a magnetic field has five degrees of freedom. Hence, a particle trajectory can be fully described by five parameters. The CBM state vector was chosen in the following form: $(x, y, t_x, t_y, q/p)$. Here, x, y are the track coordinates at a particular z position; t_x, t_y are the slopes of the track with respect to the z-axis; and q/p is the particle charge to momentum ratio.

Let us define a triplet as a group of three hits on adjacent detector stations, potentially produced by the same particle. Since a hit consists of two position measurements (x_i, y_i) of the intersection of a particle with a certain detector plane, the triplet represents a set of six coordinate measurements and allows to uniquely determine all five parameters of a reconstructed track segment. For this reason in CBM the short track segments to be built at the first stage are triplets.

The parameters used to describe a triplet are the same as the track parameters: $(x, y, t_x, t_y, q/p)$. Thus, the Kalman filter equations [2] for the estimate of the χ^2-deviation between the hit measurements and the parameters of the reconstructed track segments are the same as for the full track. It gives the opportunity to reject according to the χ^2-value most of the random three-hit combinations already at the triplet level.



```
1  Sort_Input_Hits_According_to_Grid();
2
3  for track_set (high_p_primary, low_p_primary, secondary, broken)
4
5  switch (track_set)
6    case high_p_primary:
7      Build_Triplets (min_momentum_for_fast_tracks,
       primary_track_parameter_initilisation, triplets_wo_gaps);
8
9    case low_p_primary:
10     Build_Triplets (min_momentum_for_slow_tracks,
       primary_track_parameter_initilisation, triplets_wo_gaps);
11
12   case secondary:
13     Build_Triplets (min_momentum_for_slow_tracks,
       secondary_track_parameter_initilisation, triplets_wo_gaps);
14
15   case broken:
16     Build_Triplets (min_momentum_for_slow_tracks,
       secondary_track_parameter_initilisation, triplets_with/
       wo_gaps)
17
18
19   for track_length := NStation to 3 do
20     for station := FirstStation to NStation do
21       for triplets := First_Triplet_Station to
         Last_Triplet_Station do
22          track_candidate = Build_Best_Candidate (triplet);
23
24
25   Save_Candidates(all_track_candidates);
26
27   Delete_Used_Hits();
```

```
void function Build_Triplets (min_momentum,
prim/sec_track_parameter_initilisation,
triplets_with/wo_gaps)
{
  for station := (NStation-2) to FirstStation do
    for hits_portion := First_Portion_Station to
Last_Portion_Station do

      Find_Singlets(hits_portion);
      Find_Doublets(singlets_in_portion);
      Find_Triplets(doublets_in_portion);
}
```

Figure 2. A pseudocode scheme for the CA track finder algorithm.

The strategy of the algorithm can be explained with the help of the pseudocode scheme that is shown in Figure 2. The tracking procedure starts with the initialization (Figure 2: pseudocode line 1). At this stage the algorithm allocates memory for the input information and prepares a grid data structure to store hits (Figure 3).

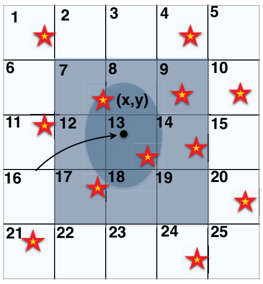


Figure 3. The grid structure provides fast access towards hit measurements in the area of track extrapolation within the extrapolation errors.

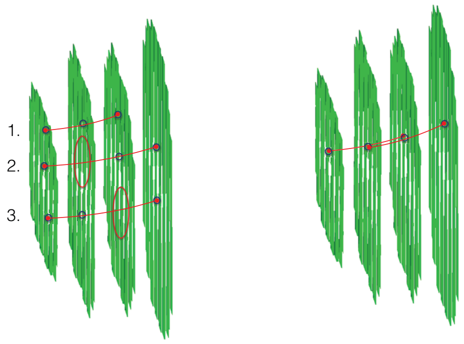


Figure 4. Left: three types of triplets built by the CA algorithm: 1) with no missing hits, 2) with the second hit missing, 3) with the third hit missing. Right: two neighboring triplets are combined into a track.

The grid structure plays an important role for the speed of the algorithm, since it provides fast access to hits for the most time-consuming part of triplets construction. While binding hits into triplets, one often faces the task of finding a hit of a triplet in a certain spatial region of a station. In order to do it fast, a regular 2D grid is introduced in each station, with the grid cell size inversely proportional to the hit density on the station. All hits are sorted and stored in the memory in the order of their grid cell numbers. For each grid cell a pointer to the first hit is stored. After extrapolation of the track segment to the next station, the algorithm obtains an estimated position (x, y) with extrapolation errors. It defines a certain region of interest on the station, where potentially a next hit can be located. One can use the grid structure to avoid going through the entire list of hits on the station and to check whether they lie in the region of interest.

After determining the region of interest, the algorithm builds a special object called HitArea, which is defined by the area of bins fully covering the region of interest (shown by the blue square in Figure 3). The grid structure allows a fast access to the hits lying within the HitArea. Since the grid stores indexes of the first hits in the bin for each bin, to check all the hits in the HitArea, the algorithm just needs to check several groups of hits, sorted and located locally in the memory, instead of going through the whole list of hits on the station.

The most time-consuming part of the algorithm (90.4% of the total time) is the triplet building stage. Triplets are built out of hit measurements that can potentially belong to the same track. Since the tracking algorithm is designed for online reconstruction, the algorithm speed plays a crucial role and a lot of optimization efforts were devoted to speed up the execution. One such optimization comes into play at the very beginning.

Monte-Carlo (MC) simulations show that a major part of particles of a particular physics interest comes from the region of the primary vertex with momentum greater than 0.5 GeV/c. Knowing in advance that a particle is emerging from the primary vertex with large momentum drastically improves the time needed to reconstruct such a track, due to a better track parameter initialization and a small curvature of the track. These observations have led to a decision to split the reconstruction procedure in several stages in order to perform the fast and computationally easy parts first and to reduce the combinatorics in the later stages.

Thus, the CA track finder consists of the following stages (pseudocode line 3):

1. search for tracks of high-momentum ($p > 0.5\,\mathrm{GeV}/c$) quasi-primary (emitted from the target region) particles,

2. search for tracks of low-momentum ($0.1 < p < 0.5\,\mathrm{GeV}/c$) quasi-primary particles,

3. search for tracks of secondary particles with arbitrary momentum,

4. search for tracks with missing hits.

For each set of tracks the same procedure is repeated with a different initial approximation and a different set of cuts. After each stage the input hits included in reconstructed tracks are flagged as used and removed from consideration at the next stage. This approach suppresses possible random combinations of hits and improves the speed of the algorithm at high track multiplicities [3].

After the type of initial triplet parameter initialization (for the primary or secondary track search) and cut values (for the high- or low-momentum track search) are defined by setting the desired track type (pseudocode line 3), the actual triplet building procedure is to begin (pseudocode lines 6–16). Triplets are built station by station, moving upstream starting from the sixth station to the first station.

At the beginning, in order to use Single Instruction Multiple Data (SIMD) registers, all input data is packed into single-precision SIMD vectors, so that all later calculations are vectorized. For reasons of memory optimization, all hits on each station are split into portions of N hits. N is a multiple of the number of elements in the SIMD register, since the hits are processed in a SIMD-ised manner.

Each triplet is built in three steps: to the starting hit second and third hits are added consecutively, rejecting non-physical combinations according to the track model at each step. Thus, at first each hit on a station is considered as a starting hit of a triplet. To the starting hit we add the target with some errors, obtaining in this way a certain spatial direction. This structure is called a singlet. The singlet together with its errors is extrapolated to the next station, taking into account the track model and effects of multiple scattering. While searching for the next hit of this station, the algorithm creates a HitArea object. All hits within the hit area potentially belong to the same track as the singlet hit. Adding each hit from the hit area to the singlet we obtain an array of doublets. The obtained doublets are fitted with the Kalman filter method and some of them are rejected due to a high χ^2-value. In the same manner each doublet is propagated to the next station and the array of triplets for the starting hit is obtained. Most random combinations are rejected due to their χ^2-value while adding the third hit to the triplet.

The triplets discussed above allow to reconstruct tracks with consecutively registered hits in each station. However, because of detector inefficiency some hits may not be registered. In this case a hit in a certain station will be missing. In order to make the algorithm less sensitive to detector inefficiency there is one additional stage of building triplets with one hit missing (Figure 4).

When all triplets are constructed, they have to be be grouped into track candidates. Since momentum is conserved in the magnetic field, in order to find triplets potentially belonging to the same track, one needs to take into account all five state vector parameters. Such so-called neighboring triplets should coincide in position, slope and momentum. The easiest way to require it, is to define triplets as neighbors whenever they share two common hits and have the same momentum within estimated errors (Figure 4). Thus, in the next step the algorithm loops over all triplets and finds and stores for each one the list of possible neighbors.

In order to further simplify the procedure of combining triplets into tracks one can also estimate a possible position of each triplet in the track. This is done with the help of the so-called *level*, which is calculated for each triplet during the search for neighbors. The level of a triplet is defined as the length of the longest continuous chain of neighboring triplets in the direction to the target. The level of a triplet helps to locate the triplet on the track: the greater its value, the longer is the track then can be constructed with the triplet.

Having estimated the position in the track, the triplets can be easily connected to track candidates (pseudocode lines 9–25). The main aim of this procedure is to build the best set of tracks, according to their χ^2-value, that do not share hits. This allows to suppress random combinations of triplets. If two track candidates share a hit, the preference is given to the longest track, since the probability of random hits to be compatible with the track model decreases with the number of hits.

Thus, the procedure starts with building the longest tracks first. Their hits are flagged as used and removed from further consideration. After the longest tracks have been found, the algorithm consecutively searches for tracks one hit less and so on until the number of hits in a primary track is three. Starting with the triplets with the maximum level, a chain of neighboring triplets is constructed, where the level of a triplet is smaller by one than the level of its predecessor. The final chain is considered as a track candidate. A treelike structure of potential track candidates is formed in this manner, from which the best track is selected according to the χ^2 criterion. Having constructed all candidates of a certain length, the algorithm sorts them according to their χ^2. After this procedure is finished, candidates are sequentially checked to contain used hits. If at least one used hit is found, the candidate is discarded; if not, the candidate is stored.

When the stage of track reconstruction for the current set of tracks is finished, the final stage of the algorithm is executed (pseudocode line 27): all the hits used in the reconstructed tracks are removed from the input to the next stage.

2.2 CA track finder algorithm performance

Let us briefly go through the four CA track finder iterations, outlining the initialization parameters and the performance achieved after each iteration.

For evaluation purposes a reconstructed track is assigned to a generated particle if at least 70% of its hits have been caused by this particle. A generated particle is regarded as found if it has been assigned to at least one reconstructed track. If the particle is found more than once, all additionally reconstructed tracks are regarded as clones. A reconstructed track is called a ghost if it is not assigned to any generated particle according to the 70% criterion.

In the first stage the algorithm searches for high momentum primary tracks. Since searching for almost straight tracks originating from the target region is relatively easy due to smaller extrapolation errors and, thus, less combinatorics, this iteration is relatively fast and supposed to reduce combinatorics for the later search. The track finding performance after the first iteration is presented in the table in Figure 5. As one can see, the reconstruction efficiency of this category of tracks 94.9% already after the first iteration, while the reconstruction of low momentum tracks and secondary tracks is not satisfactory. The track reconstruction efficiency as a function of the track momentum shows that the first iteration is able to reconstruct tracks with momenta above 0.5 GeV/c.

The main aim of the second iteration is to search for low momentum primary tracks as well. The resulting performance us shown in the table in Figure 6. After the second iteration, the reconstruction efficiency for the low momentum primary tracks has increased from 56.8% to 91.4%. However, the ghost and clone rate has increased as well because of increased combinatorics. After the second iteration the algorithm is able to reconstruct tracks with momenta down to about 0.1 GeV/c.

The third iteration is targeted at the search for secondary tracks. Figure 7 shows that the reconstruction efficiency of secondary tracks is improved from 84.5% after the second iteration to 86.6% after the third iteration for high momentum tracks, and from 54.2% to 54.7% for low momentum tracks.

In the last iteration the algorithm searches for tracks with missing hits in the STS because of detector inefficiency. The resulting performance is presented in the table in Figure 8. The overall reconstruction efficiency after the last iteration has improved by about 2%.

A special procedure is implemented in the algorithm to suppress clones, merging potentially doubly reconstructed tracks. Also, there is a special extender option, which tries to extend tracks in both directions by searching for unused hits that can be attached to the already reconstructed track. The reconstruction performance after switching on the merger and extender options is presented in the table in Figure 9. The clone rate has decreased by more than a factor of two from 1.0% to 0.4%.

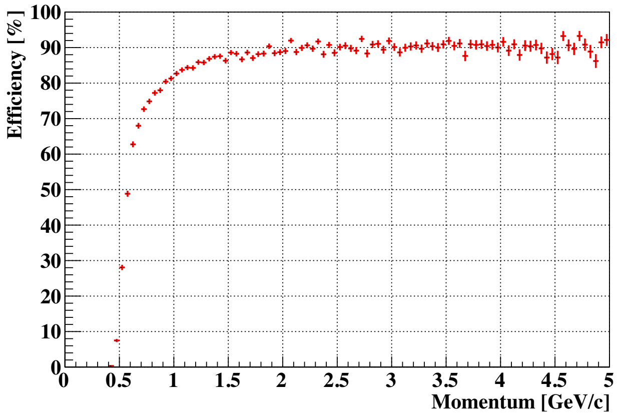
The majority of signal tracks (decay products of D-mesons, charmonia, light vector mesons) are particles with momenta higher than 1 GeV/c, originating from the region very close to the collision point. Their reconstruction efficiency is, therefore, similar to the efficiency of high-momentum primary tracks, which is equal to 97.5%. The high-momentum secondary particles, e.g. from decays of K_s^0 and Λ particles and cascade decays of Ξ and Ω, are created far from the primary vertex, therefore their reconstruction efficiency is lower, namely 91.1%. Significant multiple scattering of low momentum tracks in the material of the detector system and large curvature of their trajectories lead to lower reconstruction efficiencies of 92.6% for primary tracks and of 63.8% for secondary low momentum tracks. The total efficiency for all tracks is 90.9% with a large fraction of low momentum secondary tracks. The rates of clones and ghost tracks are 0.4% and 5.9%, respectively.

2.3 4D CA track finder

Since resolving different physical events in CBM is a non-trivial task, the CA track finder has been further modified to take into account time information. The algorithm was adjusted to take time-slices containing STS hits as input. Each STS hit carries the time measurement t in addition to two spatial coordinates x and y, measured at a certain z-position of the detector plane [4].

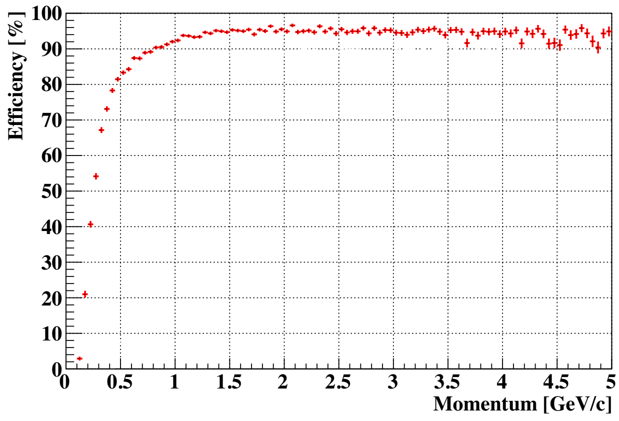
Switching to 4D tracking formally implies that the time coordinate be added to the state vector of a track:

$$(x, y, t_x, t_y, q/p) \to (x, y, t_x, t_y, q/p, t),$$



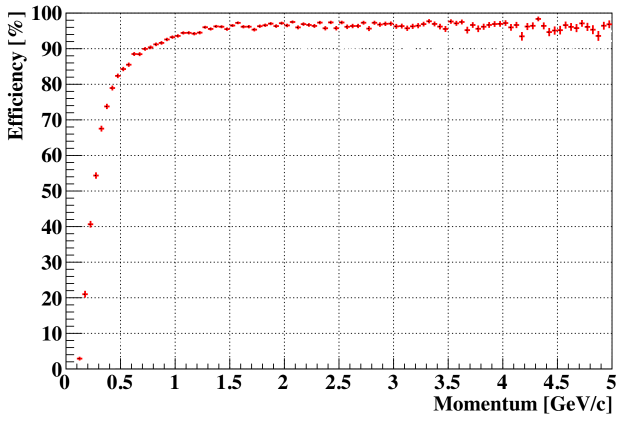
Track category	Eff, %
All tracks	70.4
Primary high-p	94.9
Primary low-p	56.8
Secondary high-p	49.7
Secondary low-p	13.0
Clone level	0.3
Ghost level	0.3
MC tracks found	103
Time, ms/ev	4

Figure 5. Track reconstruction efficiency as a function of track momentum and track finder performance after the search for primary tracks with high momentum.



Track category	Eff, %
All tracks	87.8
Primary high-p	95.8
Primary low-p	91.4
Secondary high-p	84.5
Secondary low-p	54.2
Clone level	0.9
Ghost level	5.6
MC tracks found	129
Time, ms/ev	6

Figure 6. Track reconstruction efficiency as a function of track momentum and track finder performance after the search for primary tracks with low momentum.

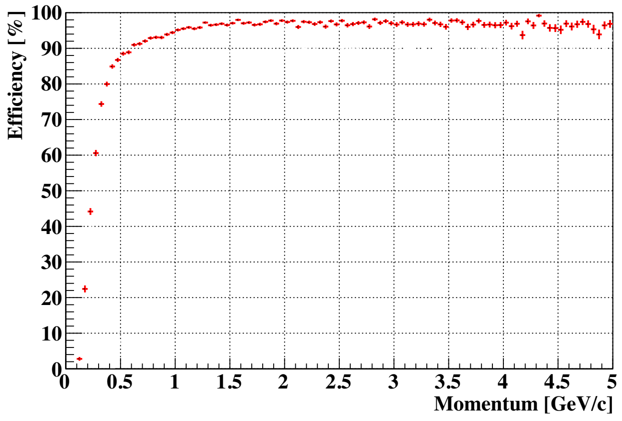


Track category	Eff, %
All tracks	89.2
Primary high-p	97.5
Primary low-p	92.4
Secondary high-p	86.6
Secondary low-p	54.7
Clone level	1.0
Ghost level	5.5
MC tracks found	131
Time, ms/ev	7

Figure 7. Track reconstruction efficiency as a function of track momentum and track finder performance after the search for secondary tracks.

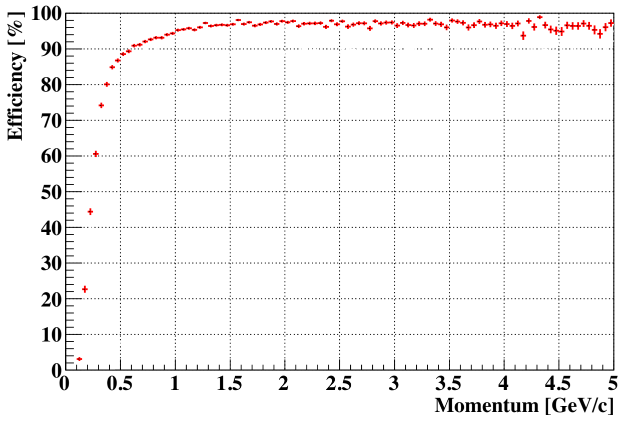
and that all operations with the state vector (e.g. propagation, fitting) involve the time coordinate on the same footing as the spatial coordinates. The feature will be implemented only after taking into account time-of -flight (TOF) detector measurements, since they are several orders of magnitude more precise and should drastically improve the time coordinate resolution. For studies involving the STS detector alone, the time is included into the track parameters, although the propagation and fitting procedures take into account only the spatial coordinates.

The STS hit time measurement information was used in the CA track finder algorithm to improve the speed and the performance. Since the triplets are to be built of three hits, potentially produced by the same particle, these hits should correlate not only in space, but also in time. Neglecting the time of flight between detector planes, hits that belong to the same track should have time measurements that are compatible within the detector time precision. For the sake of speed the additional time dimension was added to the grid structure, so that the time measurement is treated absolutely in the same way



Track category	Eff, %
All tracks	90.9
Primary high-p	97.5
Primary low-p	92.6
Secondary high-p	91.1
Secondary low-p	63.8
Clone level	1.0
Ghost level	5.9
MC tracks found	134
Time, ms/ev	8

Figure 8. Track reconstruction efficiency as a function of track momentum and track finder performance after the search for tracks with missing hits due to detector inefficiency.



Track category	Eff, %
All tracks	90.9
Primary high-p	97.5
Primary low-p	92.6
Secondary high-p	91.1
Secondary low-p	63.8
Clone level	0.4
Ghost level	5.9
MC tracks found	134
Time, ms/ev	10

Figure 9. Track reconstruction efficiency as a function of track momentum and track finder performance after merging clones.

as the spatial coordinates in the final version of the algorithm. All combinations of hits whose time measurements differ from each other by more than the expected time of flight plus the STS time precision are rejected (Figure 2, cut on the hit time measurement).

After modification of the algorithm the time-based CA track finder was tested to reproduce the results of the event-by-event analysis. Table 1 shows a comparison of the reconstruction efficiencies and the CPU performances between the event-by-event analysis (3D) and time-slice reconstruction with the 4D CA track finder.

As one can see, including the time and optimizing the 3D CA algorithm towards 4D reconstruction have made it possible to achieve a speed comparable to the case of the event-by-event analysis. Moreover, the track reconstruction efficiency has been improved after taking into account the STS time measurement compared with the event-based performance. The efficiency improvement is present even for the extreme case of 10 MHz interaction rate. This can be explained by the presence of slow particles, which create random combinations of hits in case of the event-based approach. These ran-

Table 1. Track reconstruction performance for Au+Au minimum bias events at 25 *A*GeV with the event-by-event analysis (3D) and with the time-slice reconstruction (4D), assuming a 10 MHz interaction rate. Track merging and extending procedures are included.

Efficiency, %	3D	4D
All tracks	90.6	92.2
Primary high-*p*	97.6	97.9
Primary low-*p*	93.2	93.5
Secondary high-*p*	84.4	92.0
Secondary low-*p*	53.3	65.9
Clone level	8	3.1
Ghost level	7	4.2
MC tracks found	145	145
Time/event/core	11.7 ms	13.6 ms

dom combinations can be rejected in the case of time-slices due to the hit time measurement cut, thus improving the performance.

Conclusion

The First-Level Event Selection (FLES) package for the CBM experiment contains all reconstruction stages: track finding, track fitting, short-lived particles finding, event building and event selection. For the most time-consuming part of the reconstruction procedure, the Cellular Automaton track finder is used. In order to process large time-slices, the CA track finder was vectorized using SIMD instructions and parallelized between CPU cores. The CA algorithm was further adapted to time-slice reconstruction, which is a requirement in case of CBM for the event building. The 4D CA track finder algorithm is now able to achieve the efficiency and timing performance comparable to the event-by-event analysis even at the extreme interaction rate of 10 MHz. The FLES package fulfills the experimental requirements and is ready for time-slice 4D reconstruction in the CBM experiment.

Acknowledgments

The work is supported by BMBF, HIC for FAIR and HGS-HIRe for FAIR.

References

[1] P. Senger, J. Phys. G: Nuclear and Particle Physics **28**, 1869 (2002)
[2] I. Kisel, I. Kulakov, M. Zyzak, IEEE Trans. Nucl. Sci. **60**, 3703 (2013)
[3] I. Kisel, EPJ Web Conf. **108**, 01006 (2016)
[4] V. Akishina, I. Kisel, J. Phys. Conf. Ser. **599**, 012024 (2015), http://stacks.iop.org/1742-6596/599/i=1/a=012024