Contents lists available at ScienceDirect

Nuclear Instruments and Methods in Physics Research A

journal homepage: www.elsevier.com/locate/nima

Track reconstruction by GPU in 3D particle tracking detectors

Cristiano Bozza [a,*], Chiara De Sio [a], Umut Kose [b], Simona Maria Stellacci [a]

[a] *Department of Physics of the University of Salerno and INFN Gruppo Collegato di Salerno, via Giovanni Paolo II 132, Fisciano, SA 84084, Italy*
[b] *CERN, Geneva, Switzerland*

ARTICLE INFO

Article history:
Received 2 February 2015
Received in revised form
31 March 2015
Accepted 12 May 2015
Available online 19 May 2015

Keywords:
GPU
GPGPU
Tracking
Detector
Ionising particles
Algorithm

ABSTRACT

3D detectors for high-energy physics have always needed large computing power. Its availability has sometimes determined the statistics and performance of experiments. The increasing specific computing power of GPUs in recent years offers new opportunities for this field of application that should not be missed. The paper shows a novel algorithm that supports, as a by-product of speed, wider angular acceptance with respect to established techniques based on CPUs. While the algorithm has been developed in the environment of nuclear emulsions, it has been conceived from the very beginning as a tool for general tracking in 3D detectors. The overall logic can apply to many operational contexts in which tracking occurs in high combinatorial background. The performances of the algorithm are evaluated from different points of view, describing the details of the computing technique that are common to tracking problems and discussing measurements and data from a test-beam exposure. Computing speed has been evaluated on a broad variety of hardware, investigating an approximated scaling formula.

1. Overview

Graphical Processing Units (GPUs) are used for general purpose computing providing a streaming data parallelism in which simple repetitive algorithms with few logical branches process lots of data. The paradigm of "*Single Instruction*, *Multiple Data*", or "*Single Instruction*, *Multiple Threads*", consists in executing the same instruction/transformation on a vector of data elements, usually keeping its total number of elements after the process. Particle track recognition ("*tracking*") algorithms usually have opposite features because:

- the number of tracks "hidden" in a set of primitive data elements (electronic digits, cloud chamber bubbles, nuclear emulsion grains, etc.) is not known *a priori* from the number of elements (in the following, each primitive element will be denoted as a digit);
- the number of tracks is not known *a priori*;
- the number of digits in the track is not fixed;
- tracks have global features that, rather than being a simple transformation of the features of the digits, involve one or more reduction operations, such as a least-squares-fit.

Nevertheless, the need for high-speed data processing software, pushed by the search for higher statistics in high-energy physics experiments, makes the low cost/performance ratio of GPUs intriguing. Adaptation, recoding or development of new algorithms suitable for GPUs is a worthwhile effort in many cases. The present paper describes a track recognition algorithm that has been born in the field of nuclear emulsions, after the experience of the SySal system [1] in the CHORUS experiment [2–16] and more recently with the development of the ESS [17–21] for PEANUT [22] and OPERA [23–30] to perform massively automatic topological reconstruction of events. The algorithm is not restricted to the field of nuclear emulsion, having been conceived in terms of a set of 3D digits with a weight, which are fully general concepts applicable to other detectors such as LAr time projection chambers, silicon pads, etc.

Fig. 1 shows one typical image from a microscope tomography of nuclear emulsions. It is a challenging working field, given the very high amount of combinatorial background that cannot be reduced by time triggers. In the picture shown, each high-energy track orthogonal to the plane appears with no more than one dark grain. Since scattering is negligible with respect to the measurement errors in a small region, a minimum ionising particle (*m.i.p.*) track is assumed to be straight. All the steps of the reconstruction for the algorithm presented were designed specifically for GPU and do not come from adaptation of former serial code. The performances of the algorithm are reviewed for typical problems in a nuclear emulsion technique. Details of the implementation are given when they are general enough to be ported also to other environments.

* Corresponding author. Tel.: +39 089969132.
E-mail address: kryss@sa.infn.it (C. Bozza).

http://dx.doi.org/10.1016/j.nima.2015.05.021

2. General features of tracking algorithms

Track recognition algorithms fall into two categories:

1) *global* methods attempt to work out all tracks as a whole, by seeking recurring features in the set of digits;
2) *local* methods define each track starting from a seed that progressively grows while the track is built by accumulation.

2.1. Global tracking algorithms

The archetypal method in this category is the technique of multidimensional histograms. The 3D coordinates of the digits are transformed and summed into bins of suitable size in the transformed space, possibly assigning weights (e.g., deposited energy). The bins whose contents exceed the background of random fluctuations correspond to possible tracks. Normally this information is very raw and needs further refinement. Two examples of this class of algorithms are the well-known Hough transform and the "*shift-and-sum*" algorithm described in [31,32,33]. The main advantage of such methods is that their computational load scales linearly with the number of digits. However, they have two notable problems:

1) heavy memory requirements for the multidimensional histograms;
2) poor discrimination power when the random combinations of uncorrelated noise elements compete with signal.

The latter effect is more likely to occur if the tracks have strong bending (due to magnetic fields) or deviations (due to scattering). The refinement step might be heavy. In some cases, it consists of a local tracking algorithm, of course with the reduced complexity of the small population of digits surrounding the triggering bins. The stage of bin filling in the multidimensional histogram is naturally fit for GPUs; the refinement step tends to be more often sequential than parallel.

2.2. Local tracking algorithms

Local methods try to define two-digit or three-digit alignments in a small region, which are used as seeds of a longer track. Examples of this logic are found in [1,18,34]. This class of algorithms has the advantage that at any intermediate stage tracks are rather clean, because very tight local acceptance can be set to append digits to an existing seed. On the other hand, since ionisation is a stochastic process, depending on the specifications and performances of the detector, fluctuations can leave relatively large distances empty. Moreover, if the track is bent or scattered it may be non-trivial to reconnect the initial piece with others in the neighbourhood. The concept of a track seed made of two digits leads to (at least) a quadratic dependence of the computation time with respect to the total number of digits.[a]

There are two reasons why such algorithms require special care for GPU implementation:

1) the number of trial seeds is not known in advance and depends on local spatial configurations;
2) each seed has its own evolution, which may be very long or very short; as a consequence, if each seed is being followed by a single thread, it may happen that some threads are done while others still have a lot of work to do.

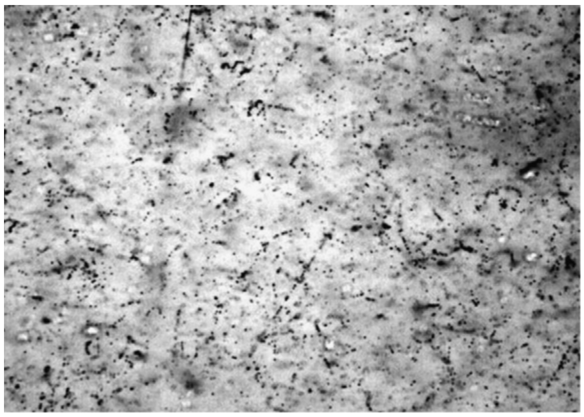
Fig. 1. 124 μm × 87 μm extract of a wider field of view of a nuclear emulsion. Grains of metallic Ag show up as dark spheroidal objects on a light background in transmitted light. 3D tomographic sequences are obtained by stacking 10–100 such images to detect straight sequences of aligned grains in space.

A common, unwanted, feature of local tracking algorithms is that they produce, for the same track, as many reconstructions as possible seeds, which may result in a large number of clones. This can be solved by preventing usage of the same digit in multiple seeds (as in [34]), but in this case the set of reconstructions depends also on the order in which seeds are considered. Another possibility is to attempt track merging at various stages; the fastest implementations of the latter solution usually also produce a dependency on the ordering of the set of primitive objects.

Despite such problems, local tracking algorithms are very intuitive and offer naturally a deeper control over track formation and validation, and often appear as finalisation steps in global algorithms. Proper implementation can mitigate the computational complexity and make a local tracking algorithm faster than global ones in specific operating conditions.

3. A tracking algorithm designed for GPU

The tracking algorithm described in this paper belongs to the class of local algorithms, and is specifically suited for "thin 3D detectors", e.g. layers of nuclear emulsions or multi-layered silicon pixel detectors. From the point of view of the algorithm, a volume can be defined "thin" if the effects of scattering and/or curvature on tracks along one direction, denoted as Z in the following, can be neglected, and a track can be reasonably assumed to cross the whole thickness without stopping or bending significantly. The maximum allowed scattering or bending is to be tuned together with the probability to collect random digits and depends on data quality and data post-processing. It is also possible to relax slightly the requirement for a track to cross the whole thickness, but one has to deal with the chance that it be reconstructed in several pieces to be reconnected later.

The original need for this method comes from application of nuclear emulsions to tracking problems over large areas (10^2 cm^2/film, 1–100 m^2 of total detector area). In a typical field of view of a microscope in emulsion (0.1–0.4 mm^2 area and 50–200 μm thickness) one may find 5000–300,000 so-called "*fog*" grains[b] and no more than 2–10 minimum ionising particle tracks crossing the plate almost orthogonally with respect to the film plane, each involving 6–20 grains. The algorithm has no other optimisation or reference to nuclear emulsions, and can be used for any detector

[a] The time is proportional to $\rho N = N^2/V$ where N is the total number, V is the volume where the digits are contained and ρ is the density. Depending on the details of the implementation, higher-order powers of the number of tracks might be involved, e.g. through $\rho^2 N$ or $\rho^3 N$.

[b] Grains developed by the statistical nature of tracking without having been touched by any ionising track. This is a flat, uncorrelated background, and the name of fog comes hereafter.

with similar geometry. Unlike other algorithms (e.g. [31,33]) designed for data from images, there is no link with emulsion image handling,

3.1. Algorithm overview

Under the aforementioned assumptions on tracks and thickness, all the digits that can be used to define a track should lie on the same line. Because of measurement errors (possibly systematic, due to distortions of the detector) and of particle-matter interaction, non-vanishing alignment tolerances have to be set. Such parameters have to be carefully tuned as they affect purity and efficiency, and generally depend on the features of the detector and on the data quality. Usually efficiency increases with wide acceptance while purity improves with small tolerances; but in the case of systematic distortions of the detector volume, too small tolerances would "deselect" real tracks. Alignment tolerance is an operational parameter of the algorithm being discussed, and there is complete freedom to set it. Setting the Z axis along the beam direction and X/Y orthogonally to it, the following conventions are used:

- The direction vector of the track is denoted as $\boldsymbol{t}$, with components t_X, t_Y, t_Z.
- The distance vector of a digit to be tested from the first extent of the track is denoted with $\boldsymbol{v}$ (and components as above).
- Δ_{XY} and Δ_Z are tunable alignment parameters.

Three different conditions have been implemented, producing different flavours of the same algorithm; the choice of the best condition depends on the features of the detector.

1. 3D alignment condition:

$$\boldsymbol{d} : \boldsymbol{v} \times \boldsymbol{t};\ \frac{\boldsymbol{d}_X^2 + \boldsymbol{d}_Y^2}{\Delta_{XY}^2} + \frac{\boldsymbol{d}_Z^2}{\Delta_Z^2} < 1$$

2. 2D alignment condition:

$$d_\perp : v_X t_Y - v_Y t_X;\ \frac{d_\perp^2}{\Delta_{XY}^2} < 1$$

3. 2.5D alignment condition:

$$\boldsymbol{u} : \boldsymbol{v} - v_Z \boldsymbol{t}; d_\perp : u_X t_Y - u_Y t_X;\ d_\parallel : u_X t_X + u_Y t_Y;$$
$$\frac{d_\perp^2}{\Delta_{XY}^2} < 1 \text{ AND } \frac{d_\parallel^2}{\Delta_{XY}^2 + \frac{v_X^2 + v_Y^2}{v_Z^2}\Delta_Z^2} < 1$$

The geometrical meaning of condition 1 is a generalisation of the 3D distance of a digit from a straight line, using possibly different weights for X/Y and Z (segmentation or readout precision may be different in-plane and off-plane). Condition 2 is useful for very thin detectors, and tracks nearly in the XY plane. Condition 3 is an evolution of the concepts in Ref. [1,18], especially suitable for data from nuclear emulsion readout microscopes: they have a sub-micrometric precision in the focal plane, but a relatively large depth of field (consequently typical values for alignment precision are $\Delta_{XY} \sim 0.3$ μm, $\Delta_Z \sim 3$ μm). Alignment on the XY plane projection is tight, but the larger error on Z implies a larger error on the position along the direction of the track, increasing with the track slope with respect to the Z axis (see [18] for more discussion).

The assumption of a thin detector is also used in the process of track formation. Each pair of digits can be used as a seed, but in practice it is better to require a minimum distance to have a good guess of the track direction. This has to be balanced with the number of available digits, because putting too strong constraints may lead to missing a track as a whole. For example, in the case of emulsion images, with a sensitive thickness of 50 μm and an average sensitivity of 30 grains/100 μm, requiring 15–20 μm prevents missing tracks while skipping a sizable fraction of seeds with insufficient aiming precision.

The seeds are propagated along the whole thickness to search for aligned digits. The geometrical parameters (position and slope) of the seed are not updated, but the track is built with the full list of aligned digits. The weight of each digit (usually related to energy deposition, ionisation, etc.) can be used to produce an overall track quality. The way to build global track parameters is typical of the detector, and the implementation of the algorithm should be customised according to the needs. In the case of nuclear emulsion data, each digit is a grain with volume and the number of sampling planes hit; the track has the number of grains, the total number of planes hit and the total volume as quality parameters. The final position and slope can be derived by a linear least-squares fit.

In all common applications, an angular acceptance window is defined. For this reason, additional geometrical constraints are set; usually a large fraction of the possible seeds (10–90% depending on the needs) will not fit in the window. Because the number of combinations scales with the square of the number of digits, it would be highly desirable to avoid even the initial computation of the unwanted seeds. A convenient way to ensure that not too many useless combinations are generated has been developed for our algorithm and is described by the following procedure:

1. The desired angular space is discretised with a suitable binning (ΔS=slope unit).
2. The centre S_X, S_Y of each bin is a possible *skewing slope*.
3. For each skewing slope, the whole volume is covered by a grid of square-based prisms (ΔP=length of the square side) as tall as the thickness of the volume (L_Z).
4. Each prism has a list of the digits contained.
5. The seeds that can be generated in each prism have a slope distribution that is centred on S_X, S_Y and is at least as wide as $\Delta P/L_Z$; in order to cover all possible slopes, the condition $\Delta S < \Delta P/L_Z$ must be satisfied.
6. The direction of each generated seed is compared to the specified angular window: it is not possible to avoid the check, but by construction it is relatively unlikely to find that, after computing the slopes, the seed has to be discarded.
7. Track formation proceeds from each seed within its prism as explained above until all combinations are used.
8. The loop continues from point 4 with a new set of prisms for a new choice of S_X, S_Y.

We notice that:

- some seeds are compatible with different sets of prisms corresponding to similar skewing slopes, so they are repeated and duplicated tracks are formed;
- if track formation is restricted to a single prism, point configurations near the edge of two (or four) prisms are scattered in more than one prism, and the track might be missed.

From the first consideration, we conclude that a track merging step is mandatory, but Fig. 2 shows that, as in many local tracking algorithms, that was already unavoidable; the second suggests that if ΔS is small enough, the missed track will reappear in a different set of prisms (Fig. 3). The optimum combination of ΔS and ΔP also depends on the "*noise*" in the position data and on the

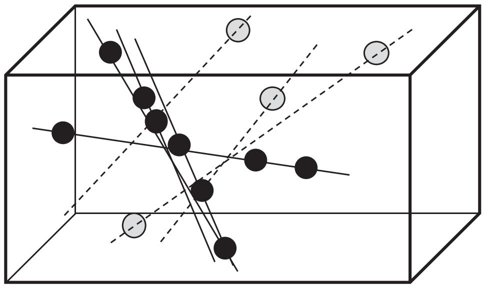

Fig. 2. Seeds are lines passing through the centres of two digits. Black digits are found aligned (within measurement errors and scattering) on one or more seeds. Valid seeds are solid lines, random seeds are dashed lines. Two possible valid tracks are in the volume sketched, but the assignment of grains to tracks may be ambiguous.

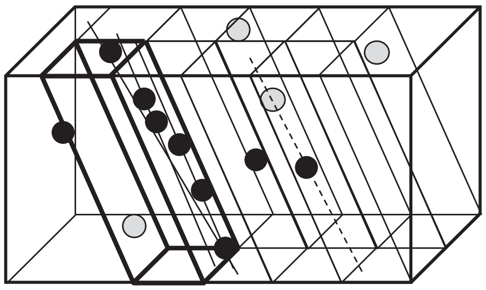

Fig. 3. For each skewing angle, a set of prisms covering the whole volume is defined. The prism with thick lines contains grains of a track and some of the possible seeds are shown. Other seeds are not tried with this skewing angle, but will be attempted in another run with a different skewing angle.

amount of digits available per track; and in general one should try to achieve high efficiency while avoiding too many trials.

One may be tempted to go further on the optimisations, and skip prisms with too few digits; but this would not really pay back because:

- an empty prism has few digits, and the computing time is proportional to n_i^2 (the number of digits in the prism), so it would be a negligible saving;
- regrouping the digits at each step takes time, and with very small prisms ($\Delta P \sim \Delta_{XY}$) the chance that digits do not all fit in the same prism grows quickly.

For nuclear emulsion data, it is convenient to set $\Delta P = 3$–$5\ \mu\text{m}$, $\Delta S = 0.05$–0.15 when $L_Z \sim 50\ \mu\text{m}$.

It is worth noticing that our variable prism geometry establishes a bridge between local tracking algorithms and global ones based on "*shift-and-sum*".

Track merging may take a sizable fraction of the total processing time, and it deserves specific optimisation. Two tracks are merged if their outmost grains are closer than specified tolerances in the *XY* plane and *Z*, usually of the order of a few micrometres. To speed up the search for matching tracks, they are arranged in a 2D grid of *XY* cells, so that only tracks in the same cell and in the first neighbour cells are considered. When two tracks are found to be clones, the one with more grains is chosen; in case of a tie, the one with higher total volume is chosen; in case of another tie, the one with lower identification number survives and the other is deleted.

The amount of clones and noise in the detector is related to its trigger system: the longer the gate time, the higher the density of track clones to be merged and the heavier the merging stage. For example, with nuclear emulsions exposed for months or years, plus random grains due to development, it is common to have a number of pairs of tracks of the order of magnitude of 10^7 to be checked for clones in a single microscope view. Even with the thousands of parallel threads allowed by a GPU, such combinatorial complexity requires careful resource usage. The code that optimises thread scheduling is a crucial part for the implementation to be operational with realistic performances.

3.2. Details of the GPU implementation

Fully exploiting the computational power of GPUs requires care about some pitfalls. One has to avoid the situation of different threads executing different instructions, which is quite usual for conventional processors (so-called *thread divergence*). In a typical hardware configuration of a GPU, (e.g. with 4 multiprocessors and a total potential of 1024 threads), at any instant no more than 4 different instructions can be executed, hence the total fraction of really active threads can be low (even 10% or less).

Memory access should also be optimised in principle, but a tracking algorithm is not "geographically" simple like a matrix multiplication. Unless the data of digits are physically reorganised at every skewing trial, digits in the same prism will not be localised in the memory map, and will retain their original placement. We build instead linked lists of indices to be stored in the prisms. There are two reasons to do that:

1. the chance that many threads are reading nearby memory locations are relatively high, as the whole set of lists is very compact: 1 integer/digit = 4 bytes/digit, whereas the full record of a digit is at least 4 times bigger;
2. accessing even one field of the digit record inevitably implies reading it as a whole: GPU achieve high internal memory bandwidth by reading consecutive blocks; it is then better to do all the combinatorial processing using only the indices stored in the lists and access the digits only when geometry computations are needed, so each digit record is read only once.

GPU processing cores are used efficiently if all threads are active. In many simple applications such as graphical processing or matrix multiplication, the number of cycles to be executed is known in advance, and threads are allocated and executed ensuring that they all complete after the same number of iterations. In the case of reconstruction/recognition algorithms, this is not possible, because real data do not occur with such highly regular patterns. It is instead possible to have all threads statistically equally loaded, and try to minimise the fluctuations. If the number of digits is high (which justifies the usage of GPUs), with an average number of n_i digits per prism, the variance will be also n_i and the standard deviation will be $\sqrt{n_i}$. The number of seeds will fluctuate (1 standard deviation) between $\frac{(n_i - \sqrt{n_i})^2}{2} \simeq \frac{1}{2}(n_i^2 - 2n_i\sqrt{n_i})$ and $\frac{(n_i + \sqrt{n_i})^2}{2} \simeq \frac{1}{2}(n_i^2 + 2n_i\sqrt{n_i})$. Such approximations are written for large n_i; also for small numbers it is interesting to work out an exact calculation in a specific case. For example, with $n_i = 9$ (a very typical order of magnitude) we will have prisms with $\frac{6}{2}5 = 15$ seeds and prisms with $\frac{12}{2}11 = 66$ combinations. When the threads working on prisms close to the lower edge of the distribution have finished their work, they will be idle for 41 more cycles (i.e. 3 times their execution time) waiting for the slowest ones (prisms with most grains). Even a single thread lagging behind the others can keep the whole GPU idle, and even 0.1% computing power usage has been observed in such implementations. This rough computation shows that distributing one thread per prism would be highly inefficient. Indeed, we tested and experienced such behaviour. It is far more efficient to assign one thread per seed. The number of seeds will be much larger than the number of threads supported by the GPU, but one can repeatedly launch

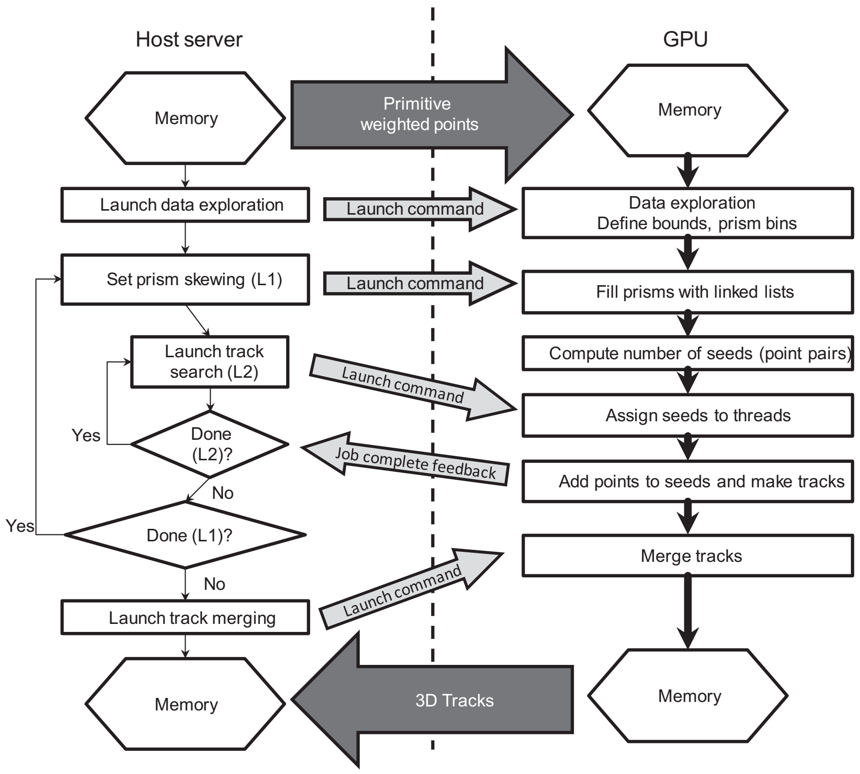

Fig. 4. Sketch of the implementation of the algorithm. The dashed line separates the host server from the GPU memory. Dark grey arrows correspond to large data transfers (100 MB–1 GB). Light grey arrows correspond to small data transfers ($<$ 100 B) to control execution. Thin black arrows in the host server region correspond to the logical flow of the execution control. Thick black arrows in the GPU region connect consecutive blocks of computation.

“swarms” of threads as many times as needed. Then, the fluctuation of the cycles per thread will be just $\sqrt{n_i}$; and with the same numbers as in the above example, some thread will check its seed against 6 digits, whereas another will check 12 digits; furthermore, the seed processing time scales linearly with the number of digits in the corresponding prism.

Likewise, in the track merging step, one thread is allocated to each pair of tracks being checked, for the same reasons explained above in the case of pairs of grains.

It should be mentioned for the sake of completeness that further optimisation is possible, but it would need saving an intermediate state of a seed to share with other threads, with doubtful performance gains. In practice, already about 60% of the code of the algorithm takes care only of distributing the work to threads in an evenly way. Such portion of the code does not take more than 3% of the computation time, but the overall gain usually exceeds a factor 10 speed-up with the datasets from nuclear emulsion images. In a scenario of on-line data processing, long tails in the distribution of processing times are not affordable, and the optimisation shown above is mandatory.

Fig. 4 shows that data exchange between the host workstation and the GPU is minimised: after data uploading, only control commands are sent to the GPU; computation results are output at the end. This is a radical approach (see e.g. Ref. [34] for a more conventional solution) that increases portability by reducing the dependency on the host and opens the door to CPU-less supercomputing scenarios.

4. Performances

4.1. Computing speed

We studied the dependency of the computing time on the number of primitive 3D objects from which tracks are built, i.e. grains for nuclear emulsion or digits for electronic detectors. The absolute tracking time depends on the hardware, and several commercially available GPU boards were tested.

The benchmarks were produced using a fixed dataset of grains read out from nuclear emulsion. The structure of background, due to the *fog*, short twisted tracks of Compton electrons and grain clones[c] is therefore representative of a realistic application. In a typical microscope view (770 μm $\times$ 550 μm), about 5×10^5 grain images are found in 31 Z layers, spanning 60 μm thickness (not all layers are really “in emulsion”, because a safety margin to detect the layer edges is needed). *Fog* makes up for the most part of grains but they are uncorrelated and rather independent of exposure time, whereas Compton tracks increase linearly with emulsion age because of ambient radioactivity. The amount of signal tracks is also proportional to the exposure time and source intensity. The size of the dataset was reduced by applying different thresholds on the minimum size of a grain image, while tracking parameters were chosen to have a realistic working set (Table 1). After reduction of clones due to finite depth of field and noise, no more than 3×10^5 possible grain images survive. The number of real *m.i.p.* tracks in this setup never exceeds 10/view, all others being mostly pieces of Compton electron tracks and random alignments.

The implementation of this tracking algorithm for nuclear emulsion data contains a data preparation step: because of the good alignment that is required for tracks, the effects of optical distortions (spherical aberrations, optical axis tilt, camera rotation) and of mechanical vibrations of the microscope have to be corrected. The former can be measured on a test set and then applied systematically on all data until the microscope setup

[c] Finite depth of field causes the same grain to appear on more than one Z sampling level.

Table 1
Operational parameters for speed benchmark. Alignment parameters are defined according to realistic applications for nuclear emulsion data. The minimum track length chosen is suitable for emulsion films with a thickness of 40 μm or more. The minimum number of grains is relatively low and would produce too much background with some kind of nuclear emulsions, but is a good "stress test". The slope acceptance is typical of many applications of experiments with a fixed target and a beam with momentum of several GeV/c impinging almost orthogonally onto the films; it is also suitable for volcanic edifice radiography using cosmic ray muons (muon radiography).

Parameter	Value
XY alignment tolerance (Δ_{XY})	0.4 μm
Z alignment tolerance (Δ_Z)	1.5 μm
Minimum track length	20 μm
Minimum number of grains (all angles)	7
Max slope per projection (*X*,*Y*)	0.65

changes; the latter are instead stochastic and must be worked out using data themselves, by detecting the displacement of patterns of grains seen on more than one *Z* layer (possible because of finite focus depth). Selection based on grain size occurs after this step. The processing time for this step is shown in Fig. 5. In an electronic detector application such data preparation step can be replaced with on-line calibration or with calibration parameters supplied by a dedicated database.

As shown in Fig. 6, the tracking time quadratically depends on the grain density, because of a similar dependency of the number of tracking seeds. It is worth to notice that the notion of nuclear emulsion images is absent at that stage, and the same performances would be obtained by feeding the algorithm with data from a silicon pad detector or a LAr time projection chamber.

The total tracking time scales linearly with the total slope acceptance (kept fixed in the test) because the slope acceptance region is spanned incrementally in tracking cycles each with its skewing slope (see algorithm overview).

The tracking time is inversely proportional to the GPU board clock. One would also expect it to be inversely proportional to the number of computing cores in the GPU. A look at Fig. 7 shows that more recent and expensive boards perform better than older ones. However, this first-glance impression must face the fact that the number of computing cores depends on the kind of instruction being executed: normally, simple arithmetic or single-precision floating-point instructions have more computing cores than double-precision floating-point instructions. We use integer numbers as much as possible and single-precision floating-point numbers when needed. Moreover, in the case of conditional jumps (loops or checks in the algorithm), *thread divergence* (i.e. the situation of some threads taking one path of the branch and others following another) disrupts the GPU paradigm of SIMT (Single Instruction – Multiple Threads). The conflict is resolved by having some threads stand-by, implying a net loss in the active computing power. Such behaviours depend critically on the hardware and the operating system of the GPU board. Because of this, we introduce the concept of "computing work" (denoted as W) as an integral of computing power over time and available resources:

$W = N_{cores} \times S_{clock} \times T_{processing}$ where N_{cores} is the total number of integer cores (which is usually the maximum number for a GPU board available in the specifications), S_{clock} is the clock rate and $T_{processing}$ is the total processing time. W would be a constant if there were no architecture difference between different boards. Fig. 8 shows that performances are not scaling with cores and expensive, high-end devices (GTX 690, 780Ti and Titan Black) are slower than expected. The reason is easily found considering that the number of cores per multiprocessor in recent architectures is larger than in older ones and hence the chance of thread divergence increases.

A cost comparison with a traditional CPU-based solution can also be sketched. The algorithm cannot be exactly the same, because 60% of the code of this implementation is devoted to optimise thread allocation for GPU processing. Such fraction of code accounts for less than 3% of the total processing time on a GPU, but would be useless and much slower on a CPU. Therefore we compare the algorithm of the system in [18] that implements similar definitions for track alignment. Table 2 shows the result of the comparison. The GPU-based solution is much cheaper than the CPU-based one, running on modern multicore machines. However, the 40 cores are envisaged to be provided by server-class machines with redundant power supply, high-performance ECC memory, and all this affects the gross cost/core. On the other hand, the GPU solution is expected to run on a relatively cheap workstation and NVidia GTX 690 has no memory error correction. This reverses the odds when one considers power consumption: NVidia GTX 690 is not one of the best energy savers and the power used by the host workstation itself is considerable. Of course, specific optimisation in the hardware configuration can be attempted (probably 30% or better on the GPU side), and the numbers we quote must be understood to be critically dependent on the choice of the components coupled to the GPU and CPU. Nevertheless, considering only orders of magnitude, it is clear that the big advantage of the usage of GPU, at the present time, is the purchase cost, when cheap components can be used. A solution based on high-end processors (e.g. NVidia Tesla K40) would be much less effective in terms of purchase cost, while it could probably give some interesting advantage on the side of power consumption.

4.2. Track quality

The quality of tracks produced by the algorithm has been estimated in a practical case using emulsion films exposed to 8 GeV/c pion beams. Films were stacked, each one in contact with others, with no spacers, as shown in Fig. 9. The angular range probed is much wider than common applications, reaching $\theta = 75^\circ$ with respect to the normal to the film plane, which needed a slope acceptance as large as $\tan\theta = 5.5$. It is worth noticing that, with commonly available hardware, the tracking acceptance of the ESS, based on CPU, has to be limited to $\tan\theta = 0.6$ for routine work and sometimes has been extended to $\tan\theta = 1.0$ with considerable slow-down (e.g., see [19]).

Emulsion films were sampled in 2 μm *Z* steps; *XY* resolution (pixel image size) was 0.333 μm. Tracking parameters were adjusted accordingly: $\Delta_{XY} = 0.4$ μm, $\Delta_Z = 1.5$ μm. Because of the relatively high background of Compton electrons accumulated during the film lifetime and randomly developed grains, quality cuts were applied to filter out random tracks:

1. grain samples with at least 4 pixels were considered for tracking; track volume (total sum of all pixels of grain samples) as a function of track length (L): $V \geq 40 - 0.1L$; this cut reduces fake tracks due to shadows of grains close to the optical axis;
2. number of grain samples in the track: $n_g \geq 8$ AND $n_g \geq 6 + 3\tan\theta$. The larger the slope, the longer the possible path length, and the consequent number of grains expected in the track.

In particular, the third requirement implies that on the test emulsion, with nominal sensitivity of about 30 grains/100 μm and 44 μm thickness at the time of exposure, efficiency cannot exceed 96% for vertical tracks ($\theta = 0^\circ$), because 4% of the tracks will have 7 grains or fewer.

The precision, efficiency and purity figures we show fold effects of data taking (microscope optics, grain recognition) and tracking; of course, in different data taking conditions or with different detectors, the results would be affected by the quality of data. Thickness after emulsion development plays a critical role: with

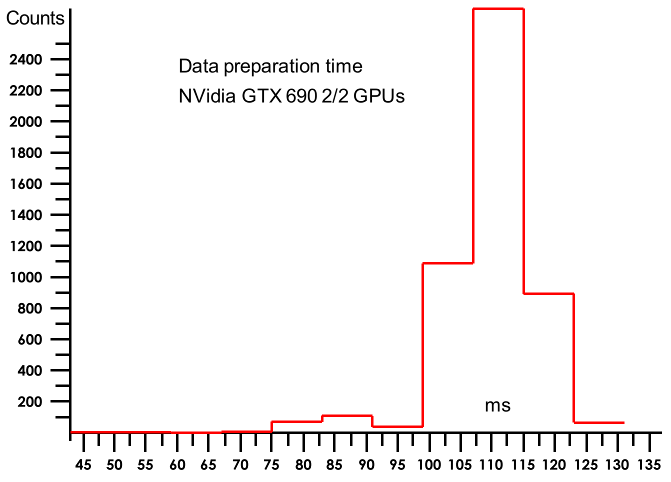

Fig. 5. Data preparation time on about 5×10^5 grain images, resulting in an average number of 2.7×10^5 grains. Performances are measured on an NVidia GTX 690, which is a double-GPU board, using both GPU's for processing.

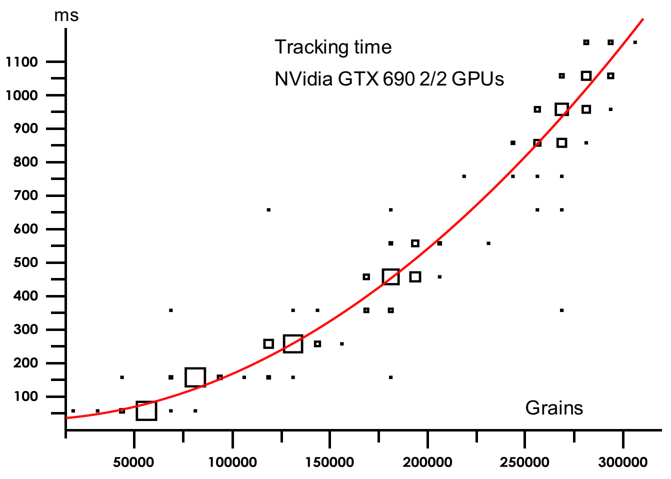

Fig. 6. Tracking time dependency on the number of grains. Data fit well with a parabolic curve, showing that the computational complexity of the algorithm is proportional to the second power of grain density. The same result would apply to the case of electronic detectors, such as LAr chambers, planes of silicon pads, etc. Performances are measured on an NVidia GTX 690, which is a double-GPU board, using both GPU's for processing.

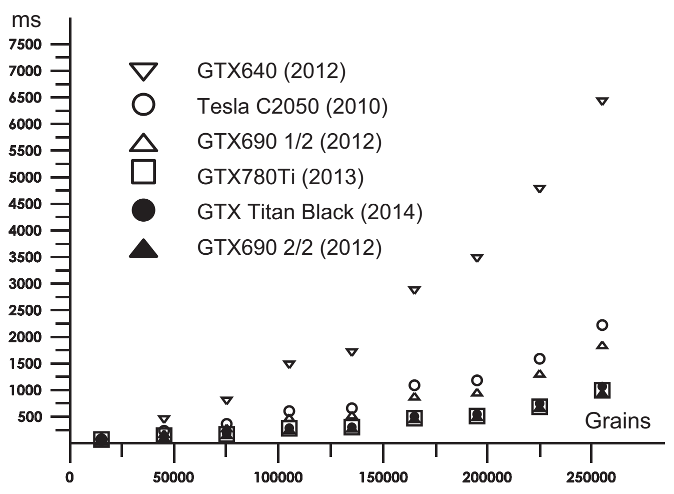

Fig. 7. Tracking time as a function of the number of grains with different GPU boards. The results are very similar for GTX 780Ti and GTX Titan Black. The launch year of each device is reported within parentheses. GTX 690 has two GPU's, so it appears in 1/2 and 2/2 mode. GTX 640 is the cheapest, GTX 690 and Titan Black have similar prices and GTX 780Ti is slightly more expensive. ECC memory and low consumption make Tesla C2050 the most expensive.

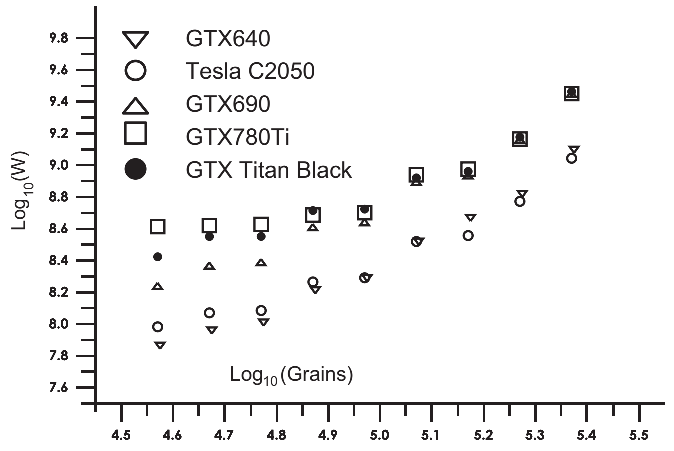

Fig. 8. Compute work as a function of the number of grains with different GPU boards. At high number of grains, two families show up, corresponding to different architectures (NVidia Fermi and Kepler). More powerful boards perform proportionally worse because of the higher number of cores/multiprocessor. At low grain densities, memory access coalescence also decreases.

non-vanishing depth of field, shrinkage blurs the Z coordinate, partially spoiling position information and increasing the confusion of real tracks with random alignments. Moreover, with fixed Z sampling, smaller thickness implies smaller number of possible samples per track, increasing the rejection by the third cut. Results are shown for two cases of shrunk films, with actual thickness of 30 μm (case A) and 34 μm (case B) after development. Slopes and angles shown were the actual ones, i.e. after emulsion shrinkage due to development. Fig. 10 shows the set of reference tracks for efficiency and precision measurements. They were selected requiring the following:

1. track angle on Y axis must be compatible with pion beams: $|\Delta \tan \theta_Y| < 0.010$;
2. reference tracks must be very well isolated, so that there are no ambiguities in assigning a possible candidate track on the film in which efficiency is to be estimated: in case the interpolated positions of two or more reference tracks are closer than 20 μm in the test film, all of them were removed from the reference set.

The position and slope of track candidates were compared to the interpolated position and the reference slope, defining difference quantities ΔX, ΔY for positions and ΔS_X, ΔS_Y for slopes. The following acceptance formula was applied:

$|\Delta X| < 20$ μm AND $|\Delta Y| < 5$ μm AND $|\Delta S_X| < 0.04$ AND $|\Delta S_Y| < 0.03$

Tolerances are wider on the X axis because the reference tracks span a wide angular range on the X axis, and position projection errors increase linearly with the slope; also the slope errors increase with the tangent of the angle. As in the case of [18], the *transverse* and *longitudinal* directions in the XY plane are defined for each track with 3D slope (S_X, S_Y,*1*) as follows:

$$\boldsymbol{t} = \frac{1}{\sqrt{S_X^2 + S_Y^2}}(S_Y, -S_X)$$

$$\boldsymbol{l} = \frac{1}{\sqrt{S_X^2 + S_Y^2}}(S_X, S_Y)$$

Table 2
Performance comparison between CPU-based tracking and GPU-based solution. GPU's turn out to be very cheap from the point of view of the purchase cost. They seem less effective from the point of view of power consumption, but the technology proposed is not the best in terms of energy saving.

	CPU solution	**GPU solution**
Data rate (encoded sets of grains)	6.8 MB/s	32 MB/s
Cores	40	12,288
Technology	Intel Xeon Processor hosted in server	NVidia GTX 690+host workstation
Purchase cost/core (arbitrary units)	0.4	0.0005208
Data rate/total purchase cost (arbitrary units)	0.425	5.000
Power consumption/core (W)	23.4	0.293
Data rate/total power consumption (arbitrary units)	0.0072	0.0088

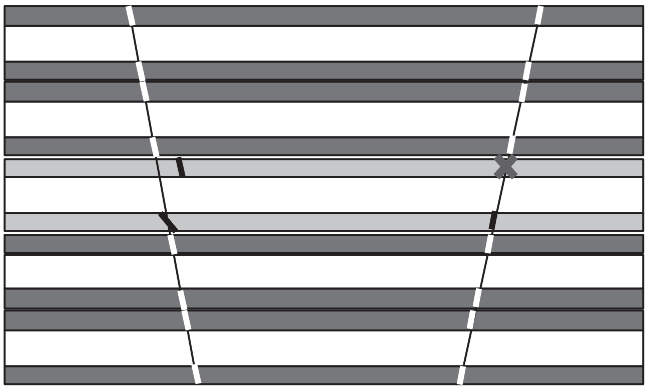
Fig. 9. A minimum ionising particle leaves tracks in consecutive emulsion films. White layers are transparent plastic supports. Dark grey and light grey layers are emulsion films. Reference tracks are seen in the 8 outer films (white marks). Track position and slope is interpolated and compared to the tracks found on the test emulsion films (black marks). Position and slope agreement are measured and tolerances are defined to accept a track candidate. In some cases, no compatible track is found (dark grey cross mark). Scattering is negligible with respect to measurement errors.

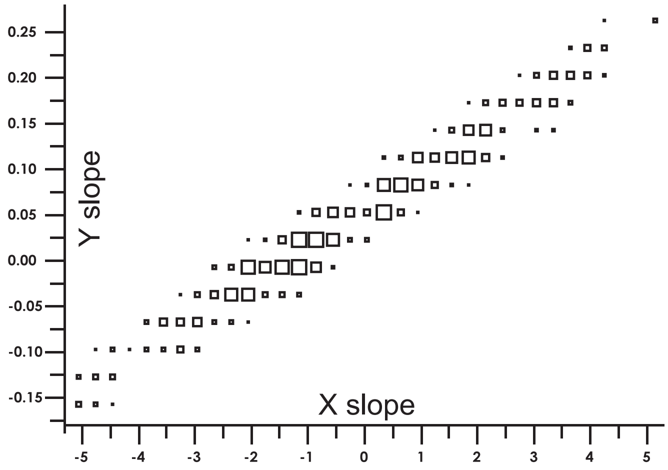

Fig. 10. Reference tracks, measured on at least 4 outer films, are chosen with the requirements of slope compatibility with pion beams better than 10 mrad, straightness (except in the film in which the efficiency is to be measured) and good isolation in position (no reference track within 20 μm) to avoid ambiguities.

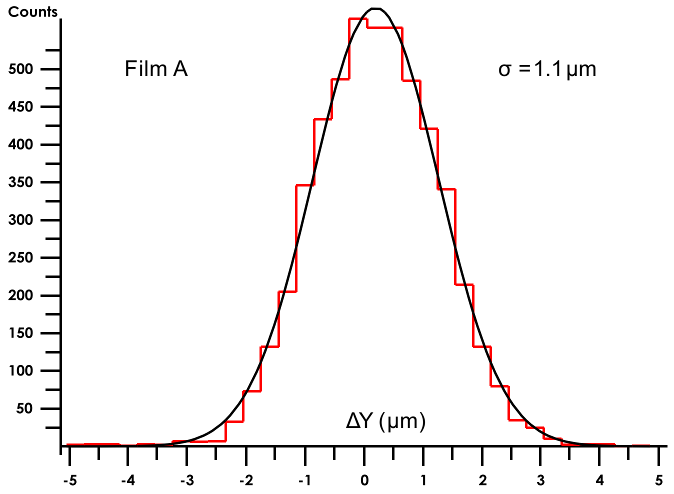

Fig. 11. Agreement of positions of measured tracks with the interpolation of the reference tracks. Given the small slope span on the *Y* axis, a single distribution can be obtained.

Position and slope errors are independent of the overall slope $\sqrt{S_X^2+S_Y^2}$ in the transverse direction, while they depend (usually linearly) on the slope in the longitudinal direction: this is an effect of the finite *Z* resolution of measurements, and the dependence on slope grows stronger for larger depth of field.

In case of multiple track candidates, the best one was chosen by selecting the smallest χ^2 in position and slope agreement. Using ΔP_t and ΔP_l for position agreement in the ***t*** and ***l*** directions respectively, and ΔS_t and ΔS_l for slope agreement, the definition of the quantity reads:

$$\chi^2=\frac{1}{4}\left[\left(\frac{\Delta P_t}{\sigma_{Pt}}\right)^2+\left(\frac{\Delta P_l}{\sigma_{Pl}}\right)^2+\left(\frac{\Delta S_t}{\sigma_{St}}\right)^2+\left(\frac{\Delta S_l}{\sigma_{Sl}}\right)^2\right]$$

$$\sigma_{Pt}=2.3\quad \mu m;\ \sigma_{Pl}=\sigma_{Pt}+(5\quad \mu m)\sqrt{S_X^2+S_Y^2};$$

$$\sigma_{St}=0.007;\ \sigma_{Sl}=\sigma_{St}+0.02\sqrt{S_X^2+S_Y^2}$$

The results of position agreement are shown for the *Y* axis of Film A (Fig. 11) and for the *X* axis of Film B (Fig. 12). In both cases, the agreement folds the measurement error in the test film and the interpolation (projection) error of the reference track. In the case of the *Y* axis, both the reference track and the tracks on the test films have the same error, so we can estimate the position measurement error $\delta_Y=\sigma_Y/\sqrt{2}=0.78\quad \mu m$. For the *X* axis, while the agreement is roughly the same as for the *Y* axis at small slopes, it is seen to worsen with larger slopes, and a linear dependence can be envisaged. The coefficient of this linear law depends on the lever arm of the projection as well as on film shrinkage and effective depth of field, and is mostly a feature of the experimental setup and data-taking microscope rather than of the tracking algorithm.

The results of slope agreement are shown for the *Y* axis of Film A (Fig. 13) and for the *X* axis of Film B (Fig. 14). In both cases, the agreement folds the measurement error in the test film and the error of the reference track. Both the reference tracks and the tracks measured on test films have the same errors, and roughly

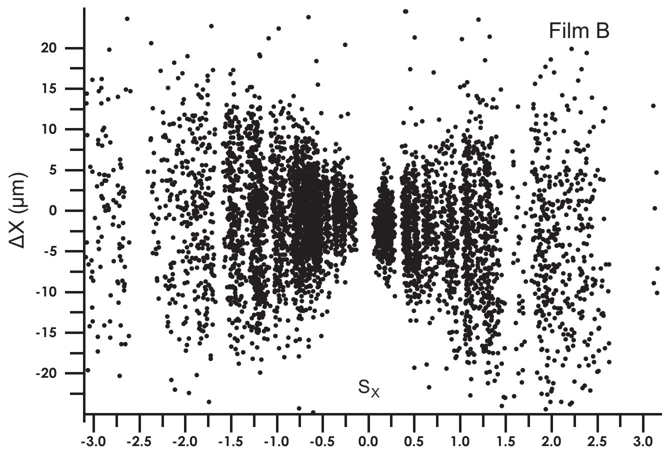

Fig. 12. Position errors on the X axis as a function of the slope on the same axis. A dependency exists, due to the finite Z resolution amplified by the lever arm of the projection from reference films to test films.

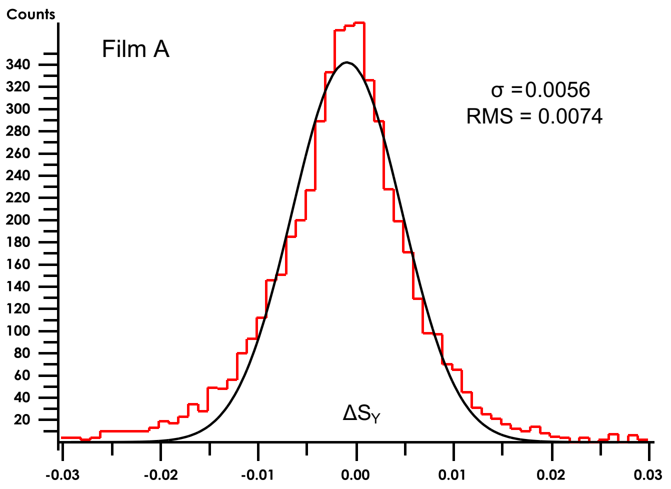

Fig. 13. Agreement of slopes of measured tracks with reference tracks on Y axis. Non-Gaussian tails are due to the increase of slope error with increasing slope.

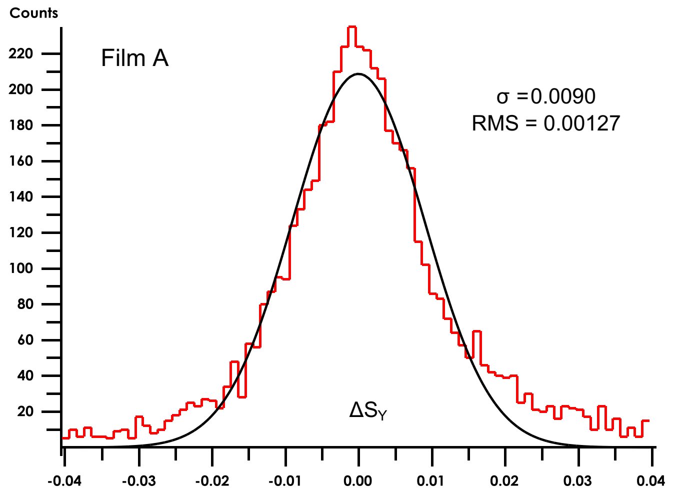

Fig. 14. Agreement of slopes of measured tracks with reference tracks on X axis. Non-Gaussian tails are due to the increase of slope error with increasing slope. Because the angular span of the test beams on the X axis is much wider than on the Y axis, the angular errors are also larger.

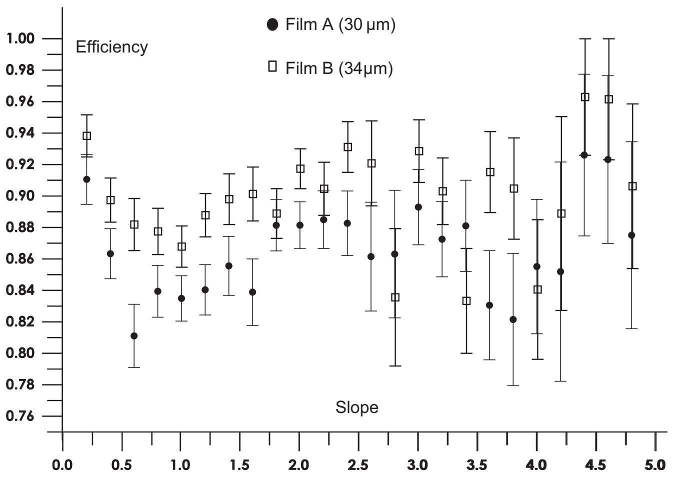

Fig. 15. Track finding efficiency as a function of track slope (quadrature sum of S_X and S_Y). The thicker film (B) has a slight advantage in performance.

one can assume that $\delta S_Y = RMS_{S_Y}/\sqrt{2} = 0.005$; in this case the RMS seems a safer choice than the Gaussian fit, because of the long tails in the distribution of slope agreement (error increasing with slope). On the X axis, the results are apparently worse, but this is related with the wider slope span, which includes tracks with larger slope errors.

The track finding efficiency including all selections is shown in Fig. 15 for the two test films.

As expected, thicker (less shrunk) emulsion film allows obtaining larger efficiency. The theoretical efficiency of 96% (defined by the cuts on the Poisson distribution of n_g for vertical tracks) is almost reached in Film B for vertical tracks, because frequently it can use one or two sampling layers more than the thinner one (A). A minimum in efficiency is observed at slope about 1.0. This behaviour is a feature of emulsion data coupled to the alignment criterion used by the algorithm and was also found in [19]. At small slopes, grain images "pile up" along the optical axis ("*shadow effect*") so that a grain may be seen even where there is none; as a result, efficiency increases for small slopes. When slope increases, the *shadow effect* is lost, but the total available path length is proportional to $\sqrt{1+S_X^2+S_Y^2}$; when the quadratic terms grow significantly, more grains become available to meet the requirements of the cut on the minimum number of digits. In a detector without the vertical pile-up effect (e.g. silicon strips or pads) the efficiency increase at small slopes would not show up; with fixed sampling planes and no "finite depth of field" effect, the total number of hits would always be equal to the number of layers, and efficiency is expected to have no slope dependency at all.

Purity of track candidates is estimated by the following procedure:

1. the X and Y coordinates of reference tracks are replaced by randomly generated coordinates within the same geometrical bounds with a flat distribution;
2. the Y slope is tilted by -0.05 so that there is no overlap with the angular region of test beams;
3. this set of fake reference tracks is used to find candidates in the results of tracking, using the same cuts and the same definition of χ^2 as in the case of real reference tracks.

Figs. 16 and 17 show the resulting distributions of agreement for positions and slopes. As expected, they are flat and are very different from the ones obtained for real reference tracks.

The fraction of fake reference tracks found is shown in Fig. 18. The higher fraction of fake tracks found in Film B is due to the larger thickness, which increases the probability that a random

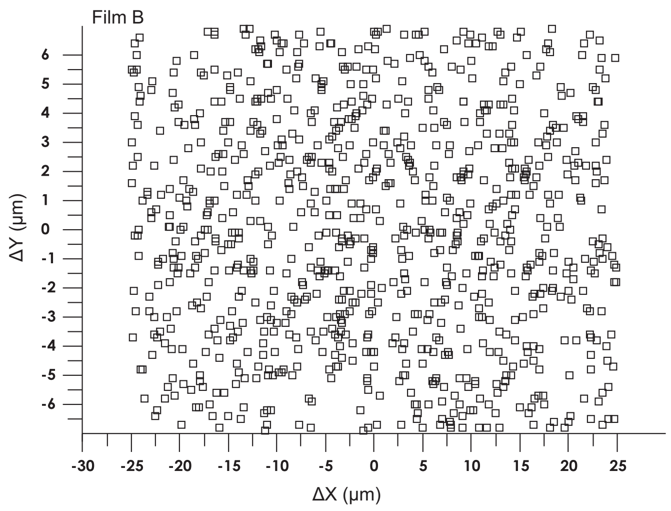

Fig. 16. Agreement of positions of measured tracks with fake reference tracks on test Film B.

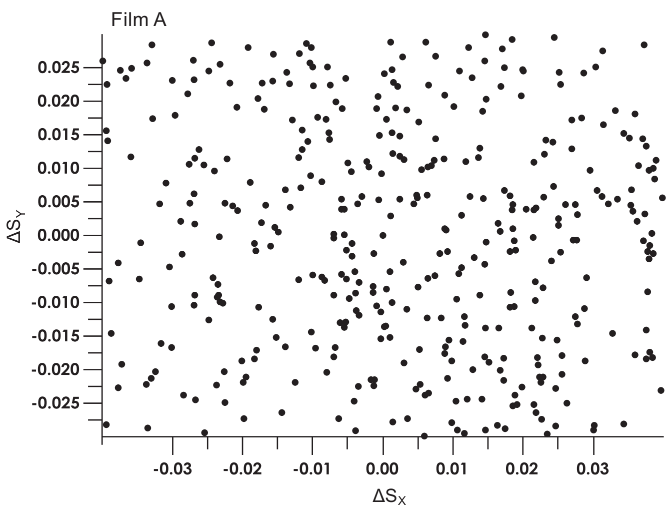

Fig. 17. Agreement of slopes of measured tracks with fake reference tracks on test Film A.

alignment exceeds the cuts on track volume and minimum number of grains.

Emulsion films are commonly used in pairs or multiplets of pairs, and multiple coincidences are required. With the amount of fake tracks seen in the cases of Films A and B, the amount of background (fake) pairs would be less than 1% or 4% respectively.

5. Conclusions and outlook

A highly optimised parallel algorithm to reconstruct tracks has been discussed. The algorithm is able to find tracks in a set of 3D digits with a weight, with or without a fixed Z sampling. The performance of the algorithm has been assessed using nuclear emulsion data, but tracking speed performances would be identical in case of data from other detectors. The algorithm is able to deal with datasets of large complexity, and tracks are found with efficiency and purity that are suitable for implementation in new data acquisition systems, replacing CPU-based solutions. At the present time, cheap GPU boards offer big advantage over traditional CPUs in terms of purchase cost. The introduction of GPU-based tracking algorithm will allow developing faster acquisition systems for nuclear emulsion applications, but can be beneficial also to other fields of high-energy physics in track recognition problems.

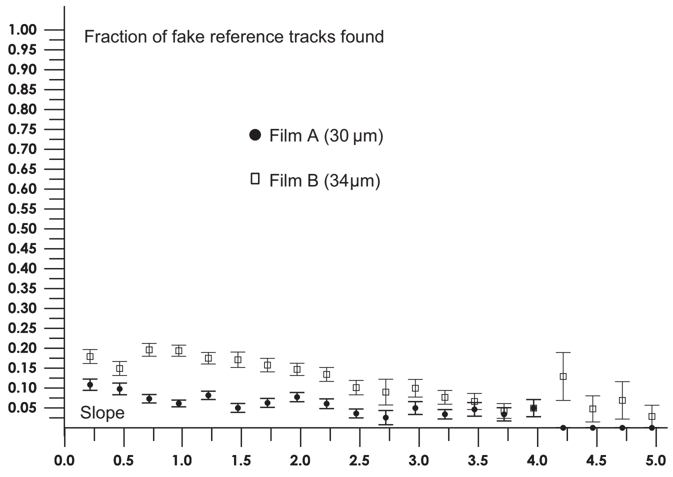

Fig. 18. Fraction of fake reference tracks found.

Acknowledgements

We want to acknowledge the active collaboration and technical support of Maurizio Di Marino of University of Salerno in installing, configuring and maintaining all the hardware needed for the tests shown in this paper.

References

[1] G. Rosa, et al., Nuclear Instruments, Methods A 394 (1997) 357.
[2] E. Eskut, et al., Nuclear Instruments, Methods A 401 (1997) 7.
[3] E. Eskut, et al., Physics Letters B 424 (1998) 202.
[4] E. Eskut, et al., Nuclear Physics B 793 (2008) 326.
[5] A. Kayis-Topaksu, et al., Physics Letters B 626 (2005) 24.
[6] G. Önengüt, et al., Physics Letters B 614 (2005) 155.
[7] G. Önengüt, et al., Physics Letters B 613 (2005) 105.
[8] G. Önengüt, et al., Physics Letters B 604 (2004) 11.
[9] G. Önengüt, et al., Physics Letters B 604 (2004) 145.
[10] A. Kayis-Topaksu, et al., Physics Letters B 575 (2003) 198.
[11] A. Kayis-Topaksu, et al., Physics Letters B 555 (2003) 156.
[12] A. Kayis-Topaksu, et al., Physics Letters B 549 (2002) 48.
[13] A. Kayis-Topaksu, et al., Physics Letters B 539 (2002) 188.
[14] A. Kayis-Topaksu, et al., Physics Letters B 527 (2002) 173.
[15] P. Annis, et al., Physics Letters B 435 (1998) 458.
[16] A. Kayis-Topaksu, et al., New Journal of Physics (2011) 093002.
[17] L. Arrabito, et al., Nuclear Instruments Methods A 568 (2006) 578.
[18] N. Armenise, et al., Nuclear Instruments Methods A 551 (2005) 261.
[19] L. Arrabito, et al., Journal of Instrumentation 2 (2007) p05004.
[20] M.De Serio, et al., Nuclear Instruments Methods A 554 (2005) 247.
[21] C. Bozza, et al., Nuclear Instruments Methods A 703 (2013) 204.
[22] A. Aoki, et al., New Journal of Physics 12 (2010) 113028.
[23] R. Acquafredda, et al., Journal of Instrumentation 4 (2009) p04018.
[24] N. Agafonova, et al., Journal of Instrumentation 4 (2009) p06020.
[25] N. Agafonova, et al., Physics Letters B 691 (2010) 138.
[26] A. Anokhina, et al., Journal of Instrumentation 3 (2008) p07005.
[27] N. Agafonova, et al., New Journal of Physics 14 (2012) 013026.
[28] N. Agafonova, et al., Physical Review D 89 (2014) 051102.
[29] N. Agafonova, et al., Progress of **Theoretical Physics** 10 (2014) 101C01.
[30] N. Agafonova, et al., The European Physical Journal C 74 (2014) 2986.
[31] K. Niwa et al., in: Proceedings of the International Cosmic Ray Symposium on High Energy Phenomena, (1974) 149.
[32] S. Aoki, et al., Nuclear Instruments Methods B 512 (2003) 466.
[33] T. Fukuda, et al., Journal of Instrumentation 8 (2013) p01023.
[34] A. Ariga, et al., Journal of Instrumentation 9 (2014) p04002.