The Problem of False Track Filtering to Improve the Efficiency of Event Reconstruction in the SPD NICA Experiment

Amirkhanova[1a], G.A., Mansurova[1] M.E., Kunelbayev[1] M.M., Ososkov[2] G.A., Shomanov[1] A.S.

[1]*al-Farabi Kazakh National University , 71 al-Farabi Ave., Almaty, Republic of Kazakhstan, 050040.*
[2]*Joint Institute for Nuclear Research, Joliot-Courie str. 6, Dubna, Moscow region, 141980 Russia.*

[a)] *Corresponding author: gulshat.aa@gmail.com*
[b)]*gososkov@gmail.com*, [c)]*adai.shomanov@nu.edu.kz*

Abstract. Reconstruction of trajectories of charged particles (tracking) is one of the actual problems in experiments in high energy physics. In the tracking program developed by a team of authors of al-Farabi Kazakh National University and JINR to process data registered by detectors located in the magnetic field of the experimental setup SPD planned in the complex of JINR NICA collider an algorithm is proposed for sifting out false tracks that arise during neural network tracking. This algorithm is based on a threshold criterion that calculates the quality of the helical line fit to the samples that make up a candidate track recognized by the neural network. In this paper, which continues this research, a method is proposed to significantly speed up the algorithm to weed out false tracks by paralleling it. The results of a comparative analysis of the computational speedup when paralleling them with conditions of a various noise background and preserving the efficiency of track reconstruction are shown.

INTRODUCTION

The SPD NICA experiment is a high-energy physics experiment that seeks to investigate the characteristics of dense matter in extreme conditions. To accomplish this, the SPD collaboration proposes the installation of a universal detector in the second interaction point of the NICA collider, which will allow for the study of the spin structure of the proton and deuteron using polarized beams. The experiment will operate at collision energies of up to 27 GeV and a luminosity of up to 10^{32} cm $^{-2}$ s $^{-1}$[1] The results of the SPD experiment are expected to contribute significantly to our overall understanding of the gluon content of nucleons and to complement other similar studies at RHIC, EIC at BNL, and at the fixed-target facilities at LHC at CERN. Conducting simultaneous measurements using different processes within the same experimental setup is vital for minimizing possible systematic errors. Additionally, the experiment has the potential to investigate polarized and unpolarized physics during the first stage of NICA's operation, with reduced luminosity and collision energy of the proton and ion beams.

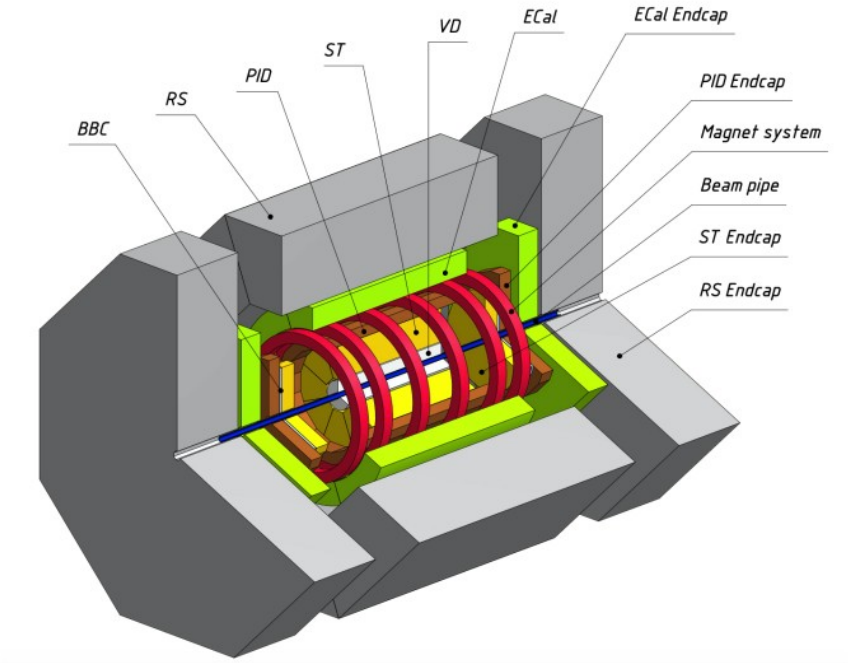


Figure 1. General layout of the SPD setup [1]

As indicated in [1] and shown in Figure 1 the tracking part of the SPD setup includes the silicon vertex detector (SVD) and straw tube-based tracking (ST) system, which together consist of 53 measuring stations.

Our focus is on the reconstruction of track measurements made in the SVD and ST systems. Unfortunately, both these detector systems have one significant disadvantage, caused by the data acquisition hardware itself. For instance, the straw-tubes of every station are arranged in parallel in two layers so that one of the coordinates of the passing track is read by the tubes of one layer, and the other one by the other layer. When the number of passing tracks is more than one, this inevitably leads to the appearance of a large number of false readings - fake hits in addition to the real ones.

Recognition of the trajectories of particles involved in collisions, for all high energy experiments, is the main stage in the reconstruction of events, necessary to assess the physical parameters of these particles and to interpret the processes under study.

Therefore, many articles are devoted to the development of track recognition algorithms and programs. Traditionally, algorithms based on the Kalman filter (KF) have been successfully used for tracking. However, in recent years, due to the increase in both the luminosity of particle beams in HEP experiments and the multiplicity of events, it has become clear that the iterative nature of KF and its poor scalability require new approaches to tracking. The most promising was the application of tracking algorithms based on deep neural networks.

One may note the articles [2-9] devoted to the use of deep learning methods in particle physics, in particular for the problem of tracking objects or particles in high-energy experiments. Gligorov [2] combines deep learning and Kalman filtering to develop a track-finding algorithm that takes advantage of the strengths of both methods. Binua et al. [3] present DeepTrack, a deep learning approach to tracking at LHC with high luminosity that involves using convolutional neural networks to analyze data and predict particle trajectories. Schlomi [4] applies deep learning to tracking problems in particle physics and explores the use of recurrent neural networks for this task. Uchida et al. [5] propose a deep neural network approach for tracking in the ILD experiment, which involves using a convolutional neural network and a recurrent neural network in combination. Antonell et al. [6] describe a deep learning approach for CMS tracker updates that involves the use of convolutional neural networks and long-term short-term memory networks. Al-Turki [7] explores the use of deep

learning for track reconstruction in high-energy physics and discusses the potential benefits of using these techniques in combination with existing ones. Divol et al. [8] present DeepTrack, a deep learning approach to tracking at high luminosity LHCs that involves the use of convolutional neural networks and recurrent neural networks within a particle flow framework. Wenzel et al. [9] also explore the use of deep neural networks for particle tracking in high-energy physics experiments and present an approach involving the use of a convolutional neural network and a graph neural network. These articles demonstrate the growing interest in using deep learning techniques to improve particle tracking in high-energy physics experiments and suggest that these techniques can significantly improve the accuracy and efficiency of particle identification and reconstruction.

Let us also note two more articles that have influenced research on the development of deep-tracking algorithms, conducted at JINR in collaboration with al-Farabi Kazakh National University. The paper [10] presents deep learning methods for the reconstruction of particle tracks in high-energy physics experiments, which is a critical task for determining the type and energy of obtained particles. The proposed methods are aimed at improving the existing methods of track reconstruction by using the capabilities of neural networks. The authors presented a type of neural network, called a "directed message network" (DMPN), which is designed to work with graph structures such as those commonly used to represent particle detector data. DMPN is capable of multi-step message passing and aggregation operations, which allows it to gather information from neighboring graph nodes and incorporate it into its predictions. This article discussed deep learning techniques that can significantly improve the performance of particle track reconstruction in high-energy physics experiments.

The next article written by Vlimant et al [11] provides an overview of HTrkX, a new software package for track reconstruction that uses deep learning techniques to process data from high-energy physics experiments. The authors describe the problem of track reconstruction, which involves determining the paths of particles produced in high-energy collisions from detector data. This task is computationally demanding, requires processing large amounts of data, and is complicated by the fact that different detectors produce data in different formats. To solve these problems, the authors propose to use deep learning techniques. They describe a new neural network architecture called the "fully convolutional network" (FCN), which is able to process data from multiple detectors and produce a single output, which is a reconstructed particle track. The authors also discuss practical aspects of implementing FCN in a real experiment, such as the need to process data from multiple detectors and the importance of optimizing the network to improve performance.

It should be emphasized, however, that the above articles did not pay special attention to the problem of the extreme difficulty of the tracking process due to aforementioned design features of such detectors as the silicon vertex detector and straw tube-based tracking system, which are parts of the SPD setup. The huge background of fake measurements generated in such detectors not only seriously complicates the tracking procedure itself, but also leads to a large number of falsely recognized tracks, as it was clearly demonstrated in [12] for data from similar detectors of BM@N experiment. In [13] a solution to the problem of screening out false tracks, appearing after the application of TrackNet Neural Network model for deep tracking data of the BM@N experiment with a fixed target, is proposed by applying a graph neural network (GNN) to all track-candidates, both true and false at once. The success of such two stages approach stimulated the idea of its application to SPD collider experiment data processing, which was performed in [13] with promising results.

There is, however, another alternative method of sifting out false tracks by applying a statistical criterion that takes into account the proximity of the sought track to some smooth spatial curve. This approach was explored in a joint study [14] by a team of authors of al-Farabi Kazakh National University and JINR, where the algorithms for sifting false tracks have been proposed based on the method of approximating the recognized points of the particle trajectory by a spatial helix line and calculating a metric describing the quality of the approximation. Exceeding some threshold by this metric means a poor match of the points included in the

candidate track to the particle's trajectory, which with high probability means that this candidate track belongs to the false tracks.

Since the trajectories of moving particles in the homogeneous magnetic subfield of the detector form spatial helical lines in the first stage of our study we used helical approximation, and the mean square error of approximation was taken as the quality metric. However, further another approximation of the found candidate tracks was also applied using a third-degree spatial polynomial, which allowed us to consider the case of an inhomogeneous magnetic field.

When solving the problem of particle reconstruction, we faced the problem of computing speed limitations related to the computational power of the computer on which the calculations were carried out. At the same time, on the current stage of the SPD setup design, when the exact characteristics of track detectors are not yet fully known, the development of event reconstruction algorithms in the future SPD experiment can be carried out using a dataset of simulated events in whose model the basic physical information about the nature of spin interactions and a simplified uniform spatial distribution of fake hits is embedded. The use of such simplified data model makes it possible to assess the fundamental possibility of applying the neural network approach to SPD, to evaluate the dependence of the speed of reconstruction algorithms and the value of their efficiency metrics.

Thus, the main goal of this study is devoted to developing the algorithms proposed in [15] for both approximations and accounting for data contamination by fake measurements, and also to overcoming the aforementioned limitations by applying paralleling techniques.
The study will contribute to this area by providing a new and effective solution to this problem.

THE SOLUTION OF A NON-LINEAR PROBLEM FOR THE ESTIMATION OF HELICAL LINE PARAMETERS

The coordinate system of the SPD tracking detectors is arranged so that the direction of the beam of accelerated particles coincides with the direction of the Oz axis and the direction of the magnetic field of the SPD. The coordinates of event vertices, i.e., particle collision points, are located along the Oz axis near the origin of the coordinates. Assuming homogeneity of the magnetic field of the detector, the particle trajectories must be close to a helical line with its axis directed along the Oz axis according to the magnetic field direction. The projection of the helical line onto the xOy plane forms a circle of radius R in the xOy plane, centered at (x_0,y_0), while the helical line itself extends along the Oz axis with an inclination to the Oxy plane defined by an angle with tangent λ. Because the circle touches the origin, we obtain $y_0^2 = R^2-x_0^2$, which reduces the number of parameters of the helical line to three.

Thus, we get the helical equations in the form of

$$\begin{cases} (x-x_0)^2+(y-y_0)^2 = R^2 & (1) \\ y_0^2 = R^2-x_0^2 & (2) \\ z= \lambda \operatorname{arctg}((y-y_0/(x-x_0)) & (3) \end{cases}$$

To estimate the helical line parameters, we used a set of model data of particle trajectory measurements in the magnetic field of SPD detector, simulated according to the pixel detector scheme without considering background measurements. For each track with n measured points on it (x_i,y_i,z_i); i=1,2,...n the values of the helical line parameters R, (x_0,y_0), and λ had to be estimated, which is a highly non-linear problem. Therefore, at first a variant of the maximum likelihood method with partial adoption of the approach from [15] by one of the authors was applied for the estimation of the parameters of the circle R, (x_0,y_0) in the xOy plane with the normalization of the minimizing functional of the form

$$F(R,x_0,y_0)=\sum_{i-1}^{n} \square((x_i-x_0)^2+(y_i-y_0)^2-R^2)^2 \quad (4)$$

by the approximated gradient of the circle R^2. By dividing (4) by R^2 and equating derivatives of R, (x_0,y_0) to zero, we obtain normal equations, which form a second-order nonlinear system:

$$\begin{aligned} F x_0 + Hb - x_0 \gamma = P, \\ H x_0 + Gb - y_0 \gamma = Q, \\ 2P x_0 + 2Q y_0 + \gamma^2 = T, \end{aligned} \tag{5}$$

where $\gamma = R^2 - x_0^2 - y_0^2$

To simplify the notation let us denote by Gauss brackets expressions like

$$\sum_{i=1}^{n} \square x_i^p y_i^q = [x^p y^q] \tag{6}$$

Then the coefficients in (5) are denoted as

$$\begin{aligned} F &= \frac{1}{n}\left[3x^2 + y^2\right], \\ G &= \frac{1}{n}\left[x^2 + 3y^2\right], \\ H &= \frac{2}{n}[xy], \\ P &= \frac{1}{n}\left[x\left(x^2 + y^2\right)\right], \\ Q &= \frac{1}{n}\left[y\left(x^2 + y^2\right)\right], \\ T &= \frac{1}{n}\left[\left(x^2 + y^2\right)^2\right]. \end{aligned} \tag{7}$$

Excluding (x_0,y_0) in the system (5) and making the variable substitution $\gamma = \gamma_0 x$,

where $\gamma_0 = \frac{1}{n}\left(\left[x^2\right] + \left[y^2\right]\right)$, we obtain the a fourth-degree equation

$$x^4 + A_0 x^3 + B_0 x^2 + C_0 x + D_0 = 0 \tag{8}$$

with the coefficients

$$\begin{aligned} A_0 &= A/\gamma_0, \\ B_0 &= B/\gamma_0^2, \\ C_0 &= C/\gamma_0^3, \\ D_0 &= \frac{D}{\gamma_0^4}, \end{aligned} \tag{9}$$

where the coefficients A,B,C,D from (9) are determined by formulae (10):

$$A = -F - G, \tag{10}$$

$$B = FG - T - H^2,$$

$$C = T(F+G) - 2(P^2+Q^2),$$

$$D = T(H^2 - FG) + 2(P^2 G + Q^2 F) - 4PQH.$$

As it is shown in [15], when solving the equation (8) by Newton's method with zero initial value, we get just one of four roots that we need. Newton's method takes only 2-5 iterations to achieve an accuracy of the order of 10^{-3}.

Calculating $\gamma = \gamma_0 x$ we obtain x_0, y_0 from (5) and find $R = \sqrt{x_0^2 + y_0^2 + \gamma}$

Then we can determine the standard deviation of the fitted circle

$$\hat{\sigma}^2 = \frac{1}{n-3} \sum_{i=1}^{n} \left(\sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} - R \right)^2. \qquad (11)$$

To calculate the slope λ of the helical line to the Oxy plane we used linear equation (3), the maximum likelihood method for which gives an estimate

$$\lambda = \frac{\sum_{i=1}^{n} z_i (\varphi_i - \varphi_0)}{\sum_{i=1}^{n} (\varphi_i - \varphi_0)^2} \qquad (12)$$

where $\varphi_i = arctg(y_i - y_0)/(x_i - x_0)$, and $\varphi_0 = 0$.
The total root-mean-square (rms) error of the fitted helical line is calculated as follows

$$\chi^2 = \frac{1}{n-3} \sum_{i=1}^{n} \left((x_i - x_0)^2 + \left(y_i - \sqrt{R^2 - x_0^2} \right)^2 - R^2 \right) + ((z_i - \lambda \varphi_i))^2 \qquad (13)$$

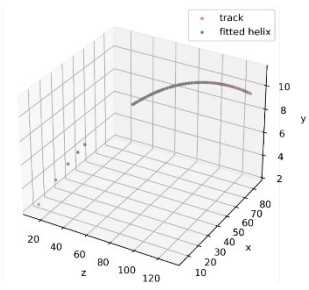
We are going to use this rms range to create a criterion for evaluating the quality of the tracking procedure.

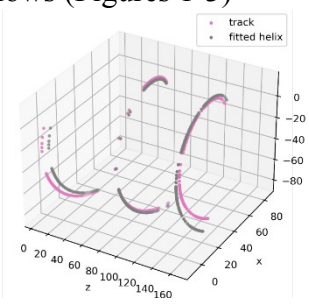
DATA SET DESCRIPTION

The iterative method obtained in the previous section made it possible to estimate the helical line parameters for the model events of the SPD experiment.

For the experiments, some statistical parameters were determined for the readout data:
- number of events: 1000
- numbering of events: from 1 to 1000
- number of tracks in each event: from 5 to 61

The tracks in the events are as follows (Figures 1-3)





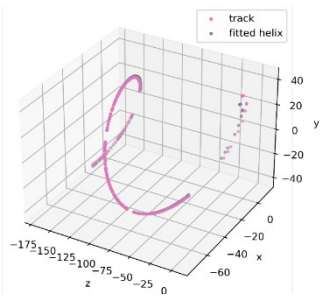
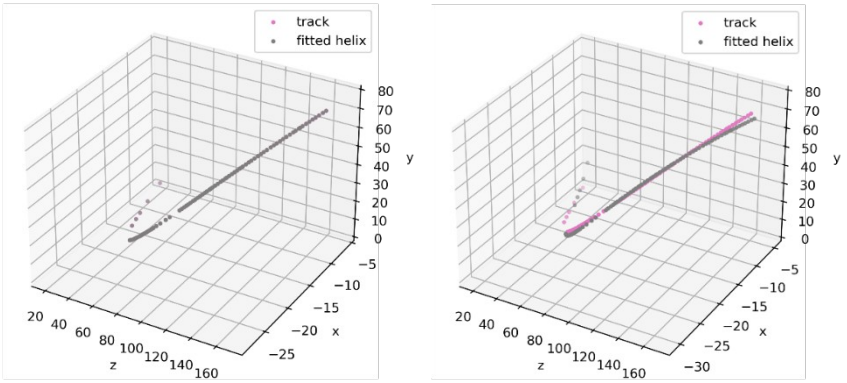


Figure 1. normal track (track 10) Figure 2. track with discontinuities (track 21) Figure 3. track repeatedly twisted (track 26)

There are no fake hits in these figures.

However, for more realistic simulations it is necessary to add fake hits to the data for each detector station. Since at the stage of the SPD setup design the information about the exact distribution of the fake hits is not yet known, it was proposed to distribute them uniformly corresponding to the position of each station. Figure 4 shows that the addition of only 10 fake hits causes a noticeable deviation in the results of helical track fitting.

To compare the effect of the level of noise on the value of our chosen criterion, we used further three levels of noise for the model data of each station: 0, 100 and 1000 fake hits.



(a) (b)

Figure 4. track with fake hits (a) – a sample of fitted trajectory with no noise, (b) – the same track sample fitted with 10 additional noisy hits.

THE DATA PROCESSING STRATEGY

The data processing strategy adopted in [15] to select the most optimal χ^2 value consisted in a preliminary clustering of all the obtained candidate tracks by the proximity of their corresponding χ^2 values and choosing the most populated cluster to estimate further the χ^2 only for tracks from this cluster as the most significant.

Three stages of data processing were used to extend this strategy to three levels of data contaminated by fake measurements.

At the first stage without contaminations described in [15] all tracks in dataset were clustered using χ^2,as a measure of track proximity. X-means clustering [16] was applied to these data The clustering was conducted in 2 phases. In the first phase initial clustering centers have been obtained and computed according to KMeans ++ algorithm. These initial centers provide a close estimate to the optimal centers due to probabilistic approach that stochastically determines the closeness criterion and makes clusters to be far apart from each other. The remaining clustering approach relies on X-Means algorithm that refines clustering decisions by computing Bayesian Information Criterion in each step of the algorithm. In our study the goal was to achieve a stable approach to determine a false track, i.e. tracks that do not correspond to a correct particle trajectory. The results of the first stage are shown in Table 1.

Cluster id	Mean	Std	Number of elements
0	4,5178	0,3570	121
1	0,0006	0,0016	16839
2	0,0225	0,0080	1091
3	0,0585	0,0131	277

4	0,1210	0,0233	419
5	0,2260	0,0378	277
6	0.3811	0,5841	259
7	0,6182	0,0807	211
8	0,9419	0,1005	182
9	1,3246	0,1326	161
10	1,8650	0,1732	144

Table 1 Clustering results for the first 10 clusters with the lowest chi-squared values

30 clusters were obtained, in each cluster the mean and standard deviation values of χ^2 for all cluster tracks were calculated. Table 1 shows the clustering results for the first 10 clusters with the lowest chi-squared values. Analysis of the clusters by mean and number of elements showed that 83% of the tracks fell into cluster #1, where the values of χ^2 have the lowest mean and standard deviation. The other clusters have noticeably higher errors in the helical fitting procedure for the tracks. A visual comparison also showed the peculiarities of the tracks of each cluster. A distinctive feature of the 'good' tracks in cluster number 1 is the absence of large discontinuities and spirals. This noticeable dependence of track properties on the value of χ^2, which characterizes the quality of the helix line fitting to each track, led to the idea of using the value of χ^2 to screen out false tracks arising during their neural network reconstruction.

Then in the first stage we developed our parallelization to speed up the procedure of false track rejection with the help of neural network tracking.

PARALLELISATION METHODS TO SIGNIFICANTLY SPEED UP THE ALGORITHM TO WEED OUT FALSE TRACKS

Parallelization within an event. The ability to execute multiple cores simultaneously from multiple event queues provides additional parallelism within an event. The order of kernel execution within a sector must be maintained, but the different sectors are completely independent [17]. Thus, it is possible to have multiple queues and simultaneously put all cores for one sector in the same queue but distribute different sectors across multiple queues. This works up to the number of queues equal to the number of sectors, although in principle it is better to match the number of queues with hardware constraints [18].

Parallelize multiple events. A completely different and simpler approach is to execute kernels for multiple events simultaneously. The HLT GPU framework allows to run independent processing components, each of which performs track reconstruction on the same GPU if there is enough GPU memory for all of them [19]. This approach can also load the GPU well but multiplies the memory requirement by the number of simultaneous queues.

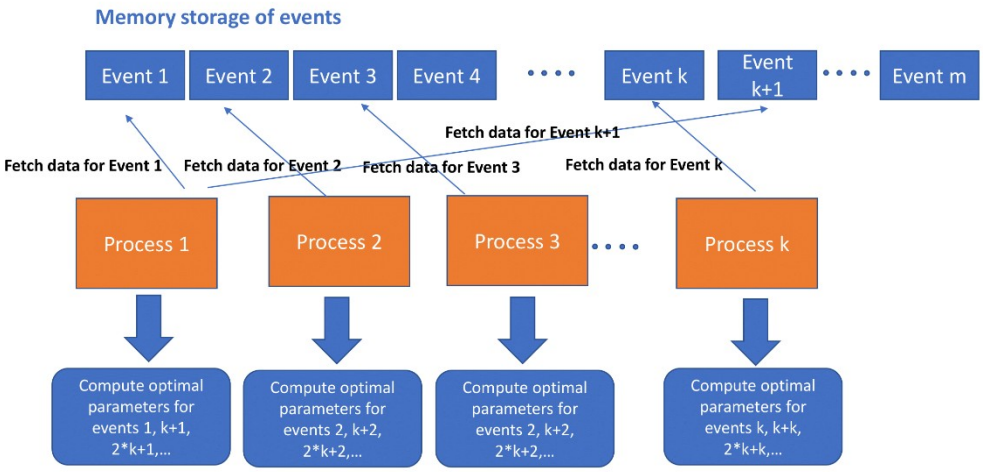


Figure 5. Schematic diagram of the parallel algorithm 1.

In our study, for greater generality, two approaches were used to approximate the found candidate tracks. In the first approach, a spatial helix was fitted, in the second - a spatial polynomial of the third degree. In the algorithm 1 proposed for computing optimal helix-loop parameters in parallel, we used running parallel threads, where each thread performs a fitting procedure for events assigned to the threads according to the round-robin enumeration and circular loop ordering method. A schematic of the parallel algorithm 1 is shown in Figure 5.

The table of all tracks contains the coordinates of the top of the event x, y, z, the identity number of the event N, the station number s, the detector number d, and the track number T within the event. Due to the large number of tracks formed by different convents, the idea of improving the performance of the process of determining the optimum helical parameters for each of the tracks arises. Thus, in our implementation, we propose a partitioning algorithm based on computation using multi-threaded computation. The partitioning algorithm is based on the Round-Robin approach based on the idea of sequential execution of tasks. If we have k threads, tasks are distributed in such a way that the first thread will compute the first track in the given table, the second thread will compute the second element, and so on, until we reach the k-th element (k-th thread is used to compute), after which k+1-th element will be processed by the first thread again, and so on, until all the elements of the table are processed.

The algorithm 1 performance is shown in Figure 6.

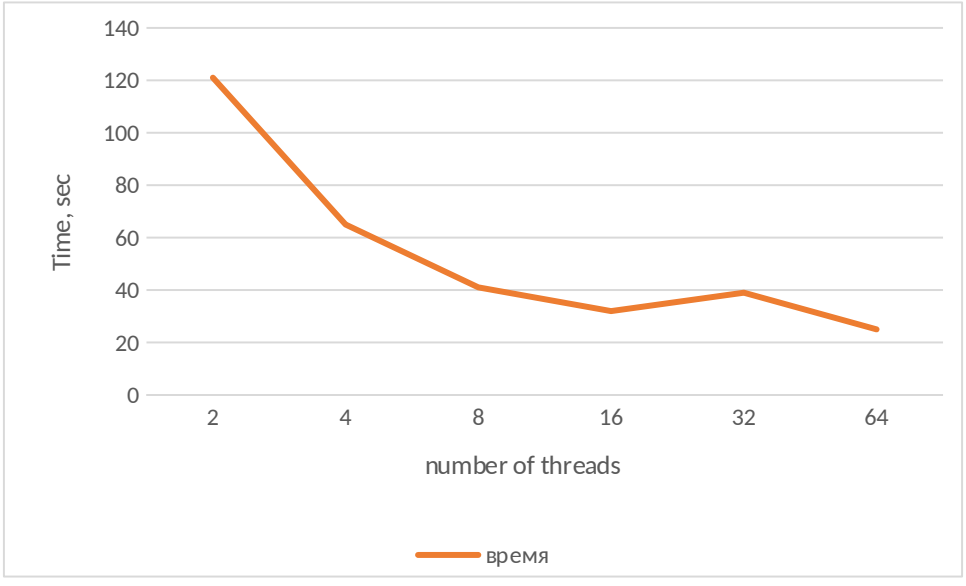


Figure 6. Running time of the parallel algorithm 1 as a function of the number of threads

Parallel Algorithm 2

In reality, due to inhomogeneity of the magnetic field of the SPD setup and the influence of various factors distorting the particle trajectory, such as Coulomb scattering, etc., the helical line in space ceases to adequately describe the trajectory. Besides, the iterative nonlinear fitting method described above proves to be slower than the linear approach and more complicated to parallelise. Therefore another method using a third degree spatial polynomial was applied.

Assuming we have a set of n knot points $P_0, P_1, \ldots, P_{n-1}$, where $P_i = (x_i, y_i, z_i)^T \in R^3$, the parametrization is made, which associates knot points with parameters u_i. Based on that parametrization, we can find an approximation curve, $f(u) = (f_x(u), f_y(u), f_z(u))$. Each of the approximation curves is represented by a polynomial function of certain degree k.

$$f_x(u) = \sum_{r=0}^{k} a_{x_r} * u^r$$

$$f_y(u) = \sum_{r=0}^{k} a_{y_r} * u^r$$

$$f_z(u) = \sum_{r=0}^{k} a_{z_r} * u^r$$

The idea is to fit the curve $f(u)$ such that distance to knot points from the curve is minimized given the parametrization u. To perform this step, we rely on the method of the least squares, where the sum of squares of residuals of the curve points and their corresponding knot points are minimized.

To approximate iteratively parameters, it is required to make an initial parameterization, which is based on centripetal parameterization approach (see formula below):

$$u_0 = 0, u_i = u_{i-1} + \frac{\left|P_i - P_{i-1}\right|^{1/2}}{\sum_{j=1}^{n-1} \left|P_j - P_{j-1}\right|^{1/2}}$$

The input parameters for the algorithm are the same as for algorithm 1 and contain the coordinates of the event node x, y, z, event identification number N, station number s, detector number d, track number T within the event.

The algorithm is based on fitting spatial polynomials of the third degree to track coordinates. To find new estimates of parameters at each iteration the Golden Ratio search method is used for locally minimizing the approximation error computed as described above from the residual values of knot points and the corresponding curve. Due to the more efficient computational cycle, this algorithm is characterized by a fast computation procedure and is therefore less computationally demanding.

Since the golden search method demands a certain number of iterations till convergence, we need to determine the optimal number of iterations for convergence to the required calculation accuracy. To this end, we have applied convergence estimation methods according to the formula:

$$n = \frac{\log\left(\frac{\varepsilon}{|b-a|}\right)}{\log(R)} \quad ((14)$$

Using this formula we can estimate the number of iterations **n** that algorithm will take to find a minimum on the given interval between endpoints u_{i-1} and u_{i+1} for each corresponding dimension in the 3D plane for the estimated parameter value u_i. Golden Ratio search is an effective way of gradually reducing the interval for finding the minimum. The key is to ensure that no matter how many points are estimated, the minimum is within the interval defined by the two points adjacent to the point with the lowest estimated value.

Next, polynomials are constructed using standard quadratic error minimization techniques:

$$E = \sum_{j=1}^{k} \left| p(x_j) - y_j \right|^2$$

Comparison of performances for two algorithms

By comparing the two algorithms the main aspects of each method can be highlighted:

	Number of iterations	**Computational complexity**	**Memory consumption**
Algorithm 1	Small	High	High
Algorithm 2	Average	Average	Average

Thus, it can be noted that Algorithm 2 appears to be more optimal according to the basic performance criteria given above.

Parallelisation is based on the multiprocessing library in the Python programming language. As an implementation, an algorithm for splitting the array of events into threads was used. The running time of the parallel algorithm depending on the number of threads is shown in Figure 7.

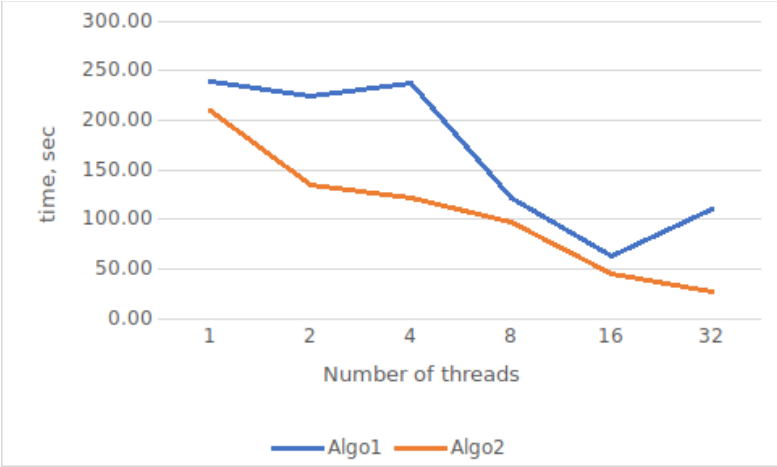


Figure 7. Comparative running times of the two proposed parallel algorithms

The algorithm was tested on a multi-core compute node with the following characteristics: Number of cores -32, Memory - 64 Gb, Processor type - AMD Ryzen, Disk memory (SSD) -2 Tb.

This system supports up to 64 parallel threads, so in our experiment we have shown computation time results for sequential code without parallelization, with 2 parallel threads, 4 parallel threads, 8 parallel threads, 16 parallel threads, 32 parallel threads and 64 parallel threads.

In our experiment we used a track table consisting of 42102 tracks. Thus, the average execution time of a subroutine to calculate the optimum helix parameters for one track from the whole set is approximately $0.5*10^{-3}$ seconds.

As we can see from the results, the algorithm shows a good speed-up, with a six-fold speed-up achieved. It is worth noting, however, that there is some deceleration of acceleration due to the additional memory fill factor.

NEURAL NETWORK ALGORITHM FOR TRACK RECOGNITION AND ITS PERFORMANCE RESULTS WITH FALSE TRACKS BEFORE AND AFTER APPLYING THE TRACK REJECTION CRITERION

TrackNETv3 is a program that implements a deep recurrent neural network designed for local tracking, where each track is recovered individually in sequence, station by station. A detailed description of the model has been described in [20]. The quality of track recovery is evaluated using the two most popular metrics: recall and precision. Recall is the proportion of true tracks that have been fully recovered. Precision reflects the proportion of true tracks among all tracks found by the model.

The neural network was trained on 50,000 model SPD events of simplified geometry. Each sample event, apart from tracks (on average 5), contained false (noise) tracks. Their number reached 60 on average.

The trained model is able to reconstruct 90% and more tracks. But at the same time, along with real tracks, a neural network outputs false tracks, the number of which is much higher. Thus, in the total number of tracks from our set, reconstructed with TrackNETv3, the share of true tracks is only 2%.

In order to increase the share of true tracks in the total number of all reconstructed tracks, we used a value of χ^2 - as a criterion for elimination. A false track is far enough away from the reconstructed helix, as opposed to a true track, that if the threshold for χ^2 is chosen correctly, only false tracks will be screened out among all reconstructed tracks.

The screening threshold was calculated under ideal conditions, on a sample containing only true tracks, as the average value of χ^2 plus three standard deviations, so that its value is 0.023. The distribution of χ^2 for both true and false tracks is shown in Figure 8.

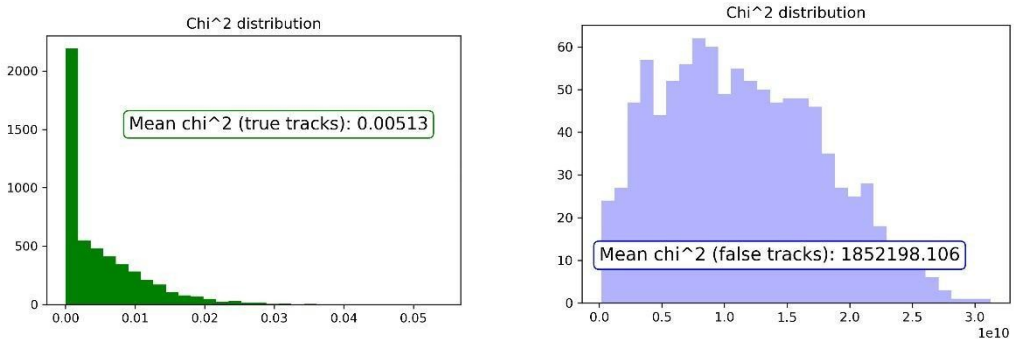


Figure. 8. Distribution of χ^2 for true left and false right tracks

The dependence of the recall and precision metrics on the value of the false track rejection criterion is shown in Figure 9

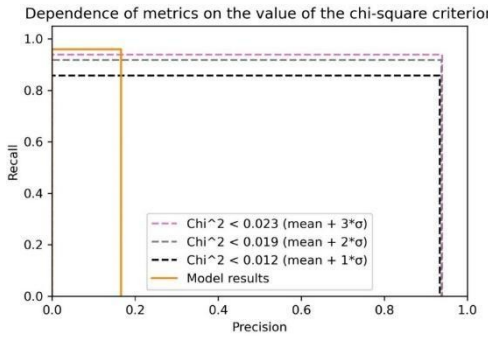


Figure 9: Dependence of the recall and precision metrics on the value of the screening criterion.

In this way, the track sifting experiment increased the accuracy metric from 0.16 to 0.93. At the same time, the recall metric did not fall below 0.93. All experiments using Ariadne library tools [21]. Resource calculations using the heterogeneous computing platform HybriLIT (LIT, JINR) [22].

FALSE TRACK REJECTION IN CASE OF CONTAMINATED DATA

Due to the bias in the original dataset, the model has some limitations in terms of potentially poor reconstruction of true tracks in the presence of highly noisy data that contain a lot of false hits. The actual data from experiments shows that on average the number of false hits in each detector can reach n to n^2, where n denotes the total number of tracks per event. Therefore, to test the introduced criteria of track clustering based on Chi-squared measure, we propose to model more realistic samples and introduce in the dataset with noisy hits corresponding to 2 different noise levels. The first noise level will contain 100 false hits per station, while higher noise level will contain 1000 false hits.

Comparative clustering results are presented in Table 2.

	0 noise level (0 false hits)	1st noise level (100 false hits)	2nd noise level (1000 false hits)
Mean value of the center cluster	0,0005	0,0011	0,0071
Standard deviation value of the center cluster	0,0012	0,0065	0,0142
Number of elements	16422	12319	10343

Table 2. Center cluster chi-squared depending on (a) zero level of noise, (b) 100 false hits, (c) 1000 false hits.

The model takes as input data about events in the form of pandas.DataFrame [23] with columns x, y, z, event (event number), station (station number), track (track number if the hit belongs to the track or -1 if it is a fake hit).

At the output, the algorithm receives candidate tracks found by the model, translates them into hit index format, and compares the resulting candidate tracks with real tracks based on a complete coincidence.

The Ariadne library was used to train and test the model.

The neural network training is based on a 3-step approach:

1) The data in the form of dataframe of coordinates (x,y,z) of hits, event, track and station identifiers fed as an input to a neural network
2) The network generates candidate tracks based on TrackNetv2.1 model
3) After thresholding procedure based on clustering based on chi-squared of track candidates after fitting a helix to a given trajectory, tracks are labeled as false or correct.

The results demonstrated in Table 3 show that the given approach can detect correct tracks with high levels of recall and precision.

Recall is computed as a ratio of number of real tracks found by the model over number of real tracks in the dataset.

Precision is computed as a ratio of the number of real tracks found by the model over the number of all reconstructed tracks.

The testing was conducted on a DGX cluster with A100 GPU 40Gb, 1Tb memory and Dual 64-Core AMD CPU with a dataset of 10000 events.

The training took approximately 25.6 hours to complete with 500 epochs for the training consisting of 100 noisy points and a bit more on a dataset with 1000 fake points

	Recall	Precision	Calculation time for 1 event (seconds)
With fake hits (100 points)	90.2	92.2	0.154
With fake hits (1000 points)	93.5	94.5	0.127
Without fake hits	89.6	91.5	0.211

Table 3. Recall and precision with fake hits (100 and 1000 hits per station) and without fake hits.

CONCLUSION

The paper proposes a method of six-fold acceleration of the algorithm to sift out false tracks by paralleling it under the condition of preserving the efficiency of track reconstruction, and also demonstrates the possibility of using the rms error of helical and polynomial fitting to the samples that make up the found candidate track as a criterion for sifting out false tracks. It is shown that an appropriate choice of threshold for χ^2 increases the fraction of true tracks among the total number of recovered tracks by almost an order of magnitude for noiseless case, although in presence of fake contaminations this speeding up is much lower. A further multiple acceleration of the algorithm is envisaged by translating the program implementing it to C++.

ACKNOWLEDGMENTS

The work was supported by the Program No. BR10965191 (Complex Research in Nuclear and Radiation Physics, High Energy Physics and Cosmology for the Development of Competitive Technologies) of the Ministry of Education and Science of the Republic of Kazakhstan.

REFERENCES

1. Conceptual design of the Spin Physics Detector, 31 Jan 2021. https://arxiv.org/abs/2102.00442.
2. V.V. Gligorov, "Combining deep learning and Kalman filtering for object tracking in particle physics," arXiv preprint arXiv:2104.05797 (2021).
3. M. Binois et al, "DeepTrack: A deep learning approach to tracking at the high luminosity LHC," JINST 14 (2019) no. 12, P12005.
4. J. Shlomi, "Applying Deep Learning to Tracking Problems in Particle Physics," arXiv preprint arXiv:1812.08284 (2018).
5. S. Uchida et al, "Track finding with deep neural networks for the ILD experiment," Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 983 (2021) 164475.
6. P. Antonell et al, "Deep Learning for the CMS Tracker Upgrade," Journal of Instrumentation 14 (2019) no. 06, P06011.
7. M. Al-Turki, "Deep learning for track reconstruction in high energy physics," arXiv preprint arXiv:2101.07825 (2021).
8. F. Divol et al, "DeepTrack: A deep learning approach to tracking at the high luminosity LHC," Journal of Instrumentation 14 (2019) no. 12, P12005.
9. M. Wenzel et al, "Exploiting deep neural networks to perform particle tracking in high-energy physics," Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 983 (2021) 164470.
10. S, Farrell et al, Novel deep learning methods for track reconstruction, https://arxiv.org/pdf/1810.06111.pdf
11.https://indico.cern.ch/event/587955/contributions/2937540/attachments/1685043/2710034/vlimant_CHEP-HtrkX_Jul18.pdf
12. A. Nikolskaia, P. Goncharov, G. Ososkov, E. Rezvaya, D. Rusov, E. Shchavelev D. Baranov, TRACKNETV3 WITH OPTIMIZED INFERENCE FOR BM@N TRACKING, https://ceur-ws.org/Vol-3041/332-337-paper-61.pdf
13. D. Rusov, A. Nikolskaia, P.V. Goncharov, E. Shchavelev and G. Ososkov, Deep neural network applications for particle tracking at the BM@N and SPD experiments, 2022. https://doi.org/10.22323/1.429.0005, https://pos.sissa.it/429/005/
14. G. A. Amirkhanova, Ye. K. Ashimov, P. V. Goncharov, A.S. Zhemchugov, M.Y. Mansurova, G. A. Ososkov, Ye.P. Rezvaya, A. S. Shomanov. Application of the maximum likelihood method to estimate the parameters of elementary particle trajectories in the reconstruction problem of the internal detector of the SPD NICA experiment / Information and telecommunication technologies and mathematical modelling of high-tech systems: Proceedings of the All-Russian Conference with international participation, Moscow: RUDN, 2022, pp. 335-341.
15. N.I. Chernov, G.A. Ososkov, Effective algorithms for circle fitting, Computer Physics Communications, 33(1984), pp. 329-333.
16. Pelleg, D., & Moore, A. W. (2000, June) X-means: Extending k-means with efficient estimation of the number of clusters. In *Icml* (Vol. 1, pp. 727-734).
17. L. Canetti, M. Drewes, and M. Shaposhnikov, ‘‘Matter and antimatter in the universe,’’ New J. Phys., vol. 14, no. 9, Sep. 2012, Art. no. 095012.

18. LHCb Collaboration, ‘‘Framework TDR for the LHCb upgrade: Technical design report,’’ CERN, Geneva, Switzerland, Tech. Rep. CERNLHCC-2012-007. LHCb-TDR-12, Apr. 2012.
19. M. (2016, July 20). Parametric Curve Fitting with Iterative Parametrization. MeshLogic. https://meshlogic.github.io/posts/jupyter/curve-fitting/parametric-curve-fitting/.
20. P. Goncharov, G. Ososkov, D. Baranov, S. Shengsen, and Z. Yao, CEUR Workshop Proc. – Vol. 2507. – pp. 130-134 (2019).
21. Goncharov P. et al. Ariadne: PyTorch library for particle track reconstruction using deep learning / P. Goncharov, E. Schavelev, A. Nikolskaya, and G. Ososkov //AIP Conference Proceedings. AIP Publishing LLC, 2021. Vol. 2377, No. 1, pp. 040004.
22. Adam G. et al. IT-ecosystem of the HybriLIT heterogeneous platform for high-performance computing and training of IT-specialists //English, in CEUR Workshop Proceedings, V. Korenkov, A. Nechaevskiy, T. Zaikina, and E. Mazhitova, Eds. – 2018. – Vol. 2267, pp. 638-644.
23. H. Stepanek, Thinking in Pandas. – 2020. https://doi.org/10.1007/978-1-4842-5839-2_1