MAASTRICHT UNIVERSITY

MASTER RESEARCH PROJECT
FINAL REPORT

Particle Track Reconstruction Using VELOPIX Data

Group 8

Authors
MAXIME DUCHATEAU
SIEMEN GEURTS
TJERK KUIL
JORRIT MILAN NATTEBREDE
MARIUS SOMMERFELD
CARLOS TELLO ORDÓÑEZ

Supervisors
DANIEL HUGO CÁMPORA PÉREZ
KURT DRIESSENS

June 23, 2021

1 Introduction

This project considers the data generated from the VELO Pixel detector (VELO), one of the detectors in the Large Hadron Collider beauty (LHCb) experiment at CERN. It is the detector closest to the particle collisions within the LHCb experiment. The name VELO comes from 'VErtex LOcator', where vertex stands for the collision point of the accelerated particles. The role of the VELO within the LHCb experiment is to locate the particle collisions and find the trajectories of the particles created in the collisions. The observed trajectories of particles that were created by the collisions then serve as trajectory seeds to the other detectors within the experiment. These trajectories consist of a set of pixel hits or hits (henceforth used interchangeably).

Currently, the LHC is inactive and upgraded towards the High-Luminosity Large Hadron Collider (HL-LHC). Part of this upgrade is a new VELO detector. The rate at which data is generated in the experiment has risen to 40 TBits per second, and a new GPU-based data processing system will be implemented. The increased data rate demands faster algorithms for reconstructing the tracks from the detector data to be developed and implemented. The algorithms implemented in a track reconstruction system need to be fast and highly accurate so that no information that is relevant to physicists is lost. Ideally, such algorithms exploit the computational strengths of the new GPU set-up.

This project set out to accurately perform trajectory reconstruction on the VELO data using the following techniques: Hopfield network, Clustering and/or Template Matching. During the process, sorting methods were also considered. These methods were further optimized through knowledge of the data and the detector. The idea behind the project was to test some relatively uncommon approaches, mainly for their physics efficiency. Matters such as computational efficiency and data structures were out of the scope for this project and can be a starting point for further research.

In this report we will first discuss the problem and research questions. In the third chapter we introduce concepts. After this, we give an overview of the previous work, followed by our project approach. Next we present the results that we found and finally, we conclude our project and discuss further research in chapters seven and eight respectively.

2 Problem & Research Questions

The approaches are compared to the current State of the Art (SOTA), with the objective to identify new methods that have the potential for comparable performance. Considered to be the current SOTA is the algorithm & results presented in the paper *Search by triplet: An efficient local track reconstruction 2 algorithm for parallel architectures* (Cámpora Pérez, Neufeld, & Riscos Núñez, in press). This leads to the following research problem:

Research problem: *How can we apply data processing algorithms to reconstruct the particle trajectories from the data of the VELO detector with a high physics efficiency in a computationally efficient way?*

To address and refine this research problem we defined the following research questions:

Research questions:

1. *What properties of tracks, events and the detector can be incorporated in the proposed track reconstruction techniques for better performance?*

   Matters such as what identifies a set of pixel hits as a track, and what sort of patterns are feasible in the detector are important to build good reconstruction methods.

2. *Which way(s) of dividing the problem into subproblems help to reduce computational complexity while maintaining high physics efficiency potential?*

   It's possible that the methods for track finding can be more efficiently utilized or executed in parallel if they only need to consider a subset of the event data. Also, there might be a consistent breakdown between easy and hard to find tracks, for example by region in the detector.

3. *How do the track reconstruction techniques considered in this project compare to the SOTA with regard to the physics metrics?*

   The key metrics we consider are physics efficiency, ghost tracks and clone tracks. Throughput optimization is likely beyond the scope of the project. The entire project is implemented in Python, which may not be the fastest language, but then again our priority is achieving competitive physics metrics.

3 Concepts

In this Section we briefly highlight some key concepts connected to the research problem.

3.1 VELO Pixel Detector in LHCb

The LHCb experiment's detector is constituted by a series of sub-detectors shown in Figure 1. This project is focused on the VELO detector.

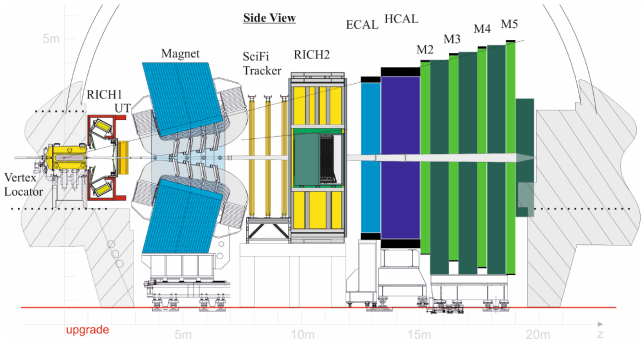


Figure 1: Side section of LHCb detector (Piucci, 2017).

The VELO is the detector that surrounds the proton-proton interaction region in the LHCb experiment (Poikela et al., 2015), being the closest detector of the experiment to the beam-line. This sub-detector is placed outside of the range effect of the LHCb magnet (because of this, trajectories can be considered to be straight lines) (Cámpora Pérez et al., in press). It contains 52 modules of silicon pixel chips arranged along the beam direction as shown in Figure 2.

The detection modules measure the particles that pass through in form of pixel clusters.

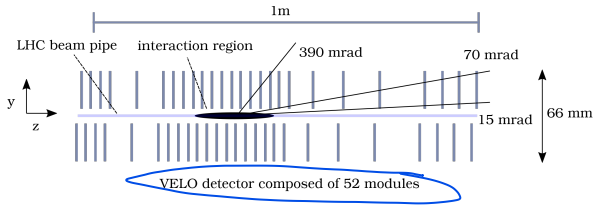


Figure 2: Schematic section in the YZ plane of VELO detector. (Cámpora Pérez et al., in press).

The data collected by the detector is filtered and then grouped by events. Each single crossing of proton beams in the Large Hadron Collider compose an event. The stored data of an event corresponds to a set of hits in the detection panels represented by a 3D coordinate system.

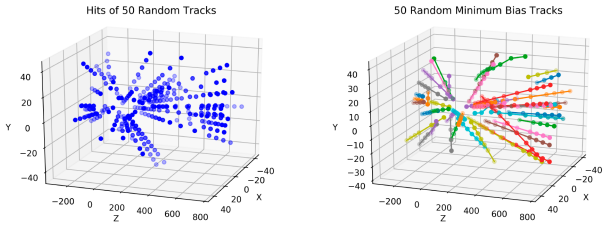


Figure 3: 3D view of an event with 50 tracks. (Left) Individual hits in event. (Right) Tracks with hits identified by colours (Roeder, 2018)

3.2 Tracks

A track is the set of hits of a particle measured in the detectors' panels that represent the trajectory of this particle. Figure 3 shows an event with a reduced number of hits and its particle trajectories. As mentioned before, VELO is outside the LHCb's magnet range effect, which allows to consider the trajectories as straight lines. A track is reconstructible when the information in the data allows to trace the trajectory of the particle. In the VELO detector the tracks are required to contain at least three hits in three different modules of the detector.

The reconstruction of tracks in the VELO is the first trajectory reconstruction step in the LHCb. These tracks are used to find the origin of the proton beams collisions and provide the seeds for reconstruction in the following detectors (Cámpora Pérez et al., in press).

Typically track reconstruction validation is performed with Monte Carlo (MC) simulated samples. The following indicators measure the physics efficiency of the found tracks (Schiller, 2011):

- The *hit purity* of a track is the fractions of hits on the track produced by the same MC particle.

$$\frac{N_{\text{track hits in MC particle hits}}}{N_{\text{track hits}}}$$

- The *hit efficiency* is the number of correctly identified hits that belong to a found track.

$$\frac{N_{\text{track hits in MC particle hits}}}{N_{\text{MC track hits}}}$$

- The track *reconstruction efficiency* can be determined by the ratio of reconstructed tracks of reconstructible particles among the number of reconstructible particles.

$$\frac{N_{\text{reconstructed \& reconstructible}}}{N_{\text{reconstructible}}}$$

- A track is referred to as a *fake track (ghost track)* when a percentage of the hits in the candidate track do not belong to the real track. In LHCb,

candidate tracks are considered to be ghost tracks if at least 70% of it's hits are not part of the real track. The *fake track fraction* is the ratio of fake tracks to the number of reconstructed tracks.

$$\frac{N_{\text{fake tracks}}}{N_{\text{reconstructed tracks}}}$$

- *clone tracks* are those tracks associated to the same particle. The *clone track fraction* refers to the ratio of these tracks to the reconstructed tracks.

$$\frac{N_{\text{clone tracks}}}{N_{\text{reconstructed tracks}}}$$

3.3 SOTA Algorithm: Search by triplets

The algorithm used as a baseline to compare the results of this project is *Search by triplets* (Cámpora Pérez et al., in press). It is a local track following algorithm optimized for the LHCb VELO detector which exploits the task parallelism inherent to the LHCb data gathering policy and the data parallelism of track reconstruction (in press).

The algorithm can be divided into three sub-algorithms:

- *Sort by phi* (Azimuth angle)

  First, hits are sorted by φ which is calculated as the 2-argument arc tangent in the XY plane with respect to the origin of coordinates. This assumes that hits produced by the same particle are likely to have the same polar angle with respect to the origin of the XY plane. (because coming from the same particle collision)

  The computational complexity of this algorithm with implemented parallelization techniques is of $O(m \times n \times \log(n))$, where m is the number of consecutive detector modules and n is the average number of hits in each module.

- *Track seeding and track following*

  The local track reconstruction task is done through two iterative stages: *track seeding* & *track following*.

  At the first stage, *track seeding* finds three hits in consecutive modules with one hit per module. A flagging mechanism avoids clone tracks. To find the triplets it first finds the doublets with non-flagged hits in the middle module and then extrapolates the these doublets to the third module. Found triplets here are passed to the next stages stored in a *seeds* container.

① Find doublets in the middle module
② Extrapolates to the third module
③ Seeds container

	Latest version (10-07-2020			
Particle category	Reco. eff	Clone fraction	Hit purity	Hit eff.
All	98.52	2.14	99.30	96.45
Strange	98.13	1.58	99.48	97.35
From B	99.30	1.16	99.74	98.11
Electrons	97.38	2.74	98.18	97.02
From B Electrons	97.00	3.68	98.42	96.68
Overall fake fractions	0.86			

Table 1: Results of physics efficiency performance indicators obtained by Search by triplet

④ Then *track following* extrapolates, similarly to doublet extrapolation in *seeding*, forming tracks to the next module. This is done by attempting to find compatible hits under a threshold. To deal with the hit inefficiency of the modules, tracks are allowed to miss one module. After the last iteration or when the forming track misses two modules it is stored in the *tracklets* or *tracks* container. *Tracklets* are those with only three hits.

These two stages have a computational complexity of $O(m \times n^2 \times \log(n)^2)$ and $O(m \times n^2 \times \log(n))$ respectively. m # of consecutives detector modules; n # of hits

- *Tracklet filter*

  Finally, the remaining tracklets are filtered under a configurable threshold with a least means square fit over the three hit coordinates of each track. Also the three hits have to be not flagged to accept the tracklets as valid tracks. This has a complexity of $O(m \times n)$.

Table 1 shows the results obtained by this algorithm for a variety of particle categories measuring the above described physics performance indicators.

3.4 Data Set

The data available for this project is generated by MC simulations. The use of simulated data is a common practice when working on track reconstruction algorithms since it allows better validation of the results.

This project will consider 2000 events from which we can differentiate two types of events:

- Minimum bias: These are events as the ones expected to be measured from the detector. Minimum bias events are the result of bunch-crossings without selection criteria and therefore their properties are not biased. These events are used

≈ events conforming to the SM of Particle Physics.

to measure the general performance of the track reconstruction algorithms.

- BsPhiPhi: The LHCb experiment is specially interested in the rare $B_s \rightarrow \phi\phi$ decay. Therefore the performance of our method(s) on this particular decay is important. This dataset has an increased occurence of the $B_s \rightarrow \phi\phi$ decay.

One event contains information about multiple MC particles which is used as test data. This information includes the angle of the particle trajectory with the beamline, info whether a particle originates from decay as well as the hit point coordinates of the particle trajectory thought the detectors.

4 Previous Work

This section contains an overview of some of the recent work in the field of track reconstruction, more specifically regarding: Template Matching, Hopfield networks and Clustering.

4.1 Template matching

Template matching is true to its name: comparing (matching) expected patterns (templates) to the data. This is one of the simplest pattern recognition methods, and requires the amount of possible patterns to be finite and a limited complexity of the detector so that all feasible templates can be handled (Mankel, 2004). The amount of templates that is feasible is estimated to be in the order of 10^5 (Frühwirth et al., 2000). This number might be slightly higher presently because of progress in computing, but more recent references on this matter could not be found. Template matching has been implemented in the ARGUS experiment (Koch et al., 1996). This case also considered a 3D space, but with more physical parameters. In this case, more than 245760 templates (vertical columns in 3D parameter space with certain height, width and depth) were used (Koch et al., 1996).

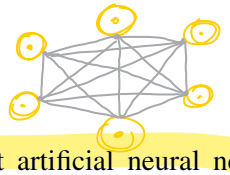
One of the possible problems with template matching is the granularity of the detector: too many templates are needed if the granularity is too high. However, if the templates considered are of low granularity, tracks that occur close to each other are not distinguishable by the method (Mankel, 2004). A way to resolve this problem is to use a search tree: first finding matches to low granularity templates, and applying higher granularity templates on the regions where low granularity template matchings have been found (Hulsbergen, 2007). Tree search has been implemented in the HERMES spectrometer (Ackerstaff et al., 1998). Important for tree search is simplicity and symmetry in the detector design (Mankel, 2004), a condition which the VELO detector fulfils.

4.2 Clustering

The set of pixel hits that form a track can be thought of as a cluster. Pixel hits belonging to such a cluster share certain characteristics/features. As mentioned before, one can express a line by means of angles or slopes. Hits of the same track have the same value for these features, thus cluster together, given the right distance measures.

Clustering as a way of performing track reconstruction is not very common in the literature. The varying number of clusters (i.e. tracks) and the possible different shapes of tracks make clustering harder than a straight forward k-means. Ferenc Siklér found a way to do this, by determining initial track candidates(Siklér, 2019). Initial track candidates are constructed by finding all mutual nearest hits neighbours in the angular space w.r.t. the center of the detector(Siklér, 2019). The approach considered in the paper is designed for tracks that are not solely straight lines. This way of initializing the track candidates could be useful for our clustering method.

An alternative way to cluster is by extrapolating the hits onto a plane far away from the sensors in the positive Z direction (Bogdan & Cámpora Pérez, 2016). Bogdan & Cámpora Pérez apply several clustering methods to the extrapolated hits points on a plane with $z = 10000$ (2016). They find that using HDBSCAN with a post-processing step and combined with the classical algorithm is a viable addition to the track reconstruction system. Clustering alone does not perform sufficiently well (2016). Interesting variations to this can be to consider a different z value for the extrapolation plane, or warp the plane to make it convex or concave and see if this improves clustering.



4.3 Hopfield Networks

Hopfield networks are recurrent artificial neural networks that exhibit remarkable properties (Hopfield, 1982). These neural network structures consist of a single neuron layer, where the neurons are interconnected to other neurons in the network. The Neurons activations are either bound between -1 and 1 or 0 and 1. Additionally, the restriction can be imposed that only bi-

nary states are possible. The states of all neurons define the state of the network, which is assigned an energy value. The energy is calculated as the sum of all active neuron biases/weights and all weights of edges between two active neurons. After any initialisation of such a network an algorithm seeks to lower the overall energy of the network by visiting and updating all neurons in randomised sequence.

The usage of Neural Networks in high energy physics (HEP) was discussed actively in the academic community subsequently to the first publications about Neural Networks and the backpropagation algorithm (Denby, 1993). The experimental results in pattern recognition using Neural Networks as well as potential performance gains in online (in-time calculation) systems made them attractive for HEP. The idea of using Hopfield networks for particle track reconstruction was first introduced about ten years after Hopfield's publication (Stimpfl-Abele & Garrido, 1991; Hopfield, 1982). The approach by Stimpfl-Abele and Garrido to track reconstruction using Hopfield networks uses the property that they are guaranteed to converge to a local minimum when updated using a local rule. The problem is modeled in such a way that good solutions / track reconstructions should be found in the local minima of the energy function. This is modeled using the Hopfield network by associating the neurons in the network with potential track segments between any two detected hit points in subsequent detector modules. If a particular neuron is activated the segment is considered to be part of a particle track.

To make the network converge to a state that models the reconstructed particle tracks, several geometric properties of the problem are used when choosing the weights of the network. To yield the best results Stimpfl-Abele and Garrido also include network 'temperature' to make the network converge to the better minima of the energy function (Stimpfl-Abele & Garrido, 1991). The findings of this experimental implementation of a Hopfield network for track reconstruction were that this method can yield high efficiency with decent speed. The bottleneck of the computation time is located in constructing and updating the network. A similar implementation of the Hopfield network for track finding has been implemented for the LHCb Muon system (the last detectors of the LHCb) (Passaleva, 2008). Zlokapa et al. have investigated the potential speedups in the application of Hopfield networks, which are also called Denby-Peterson networks, by utilising quantum annealing.

5 Data Analysis

This section deals with information, connected to the data, that can be utilised in algorithm design. Initially, the data sets are described briefly followed by the presentation of some detector specific information.

5.1 Hits, Tracks and Noise

There are on average about 2000 hits per event with a high standard deviation of around 1000 in both data sets: BsPhiPhi (Mean: 2380.68, Std.: 1056.05) and Minibias (Mean: 1963.88, Std.: 1099.2).

The number of tracks and the number of noise hits per event are directly connected to the number of hits per event. In Minibias, there are about 275 tracks and 215 noise hits (non-track-hits) per event. Both distributions are slightly skewed to the right as shown in Appendix A Figures 8 and 9. In BsPhiPhi there are 330 tracks per event and 250 noise hits per event on average. Similar to the Minibias data-set, the precise numbers per event vary around the means and the distributions show a slight skew to the right. In both datasets, a linear relationship of about 3 noise hits per 4 tracks is present.

The closest hit of a track to the beam-line has an average polar distance of 9.029 with an standard deviation of 4.365. Figure 10 in Appendix A shows the heat map of these hits in the XY plane.

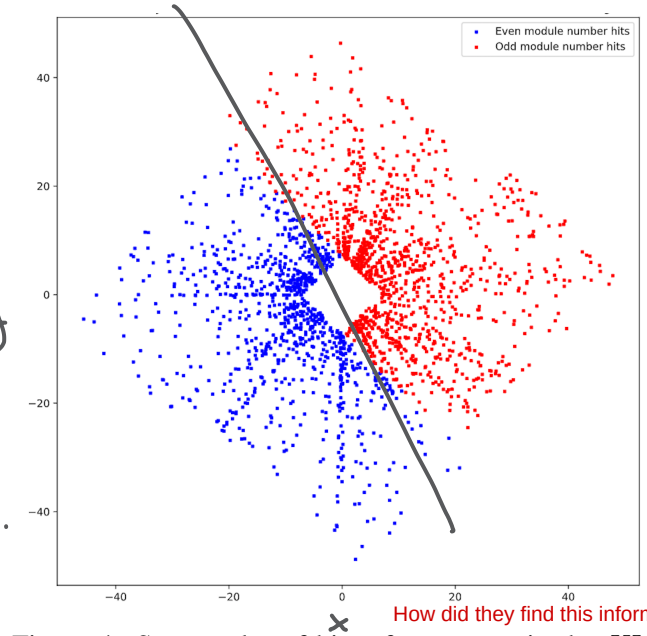
The effect of dividing the hits into k-clusters according to their polar angle is visible in Appendix A. Figure 11 shows the percentage of tracks that have hits in different clusters considering 2 to 16 clusters. Given the high percentage of tracks truncated by the clusters the developed algorithms do not focus their implementations on this parallelization approach.

Figure 12 in Appendix A shows a representation of the hits in an event by their polar angles. These plots show the effect of considering a shifted origin of the Z-axis when looking at the polar angles that involve the Z coordinate. It shows how the polar angle values of the hits that belong to the same track align into a single point. The effect of shifting the origin of the Z-axis can be translated into defining the origin of the collisions in an event, which actually is the main goal of the track reconstruction problem in the VELO.

5.2 Detector Specific Information

There are some detector specific features present in the data that are interesting in terms of algorithm design.

The VELO consists of 52 detector modules. These modules are numbered in increasing order. In Figure 4 the hits of one event are displayed by only the X,Y coordinates of the hits. This can be understood as projecting the hits along the beam-line (Z-axis) into 2D space. In Figure 4 the hits are coloured by the criterion, whether they were detected in a module with even or odd numbering.



How did they find this information ?

Figure 4: Scatter-plot of hits of one event in the XY plane, coloured on the condition: even / odd module number

This Figure displays that the upper right part of the detector is solely covered by modules with odd numbering, while all particles that are detected in the lower left part of the detector are detected by modules with even numbering.
Connected to this pattern every module has two Z-coordinates. A similar colouring technique as in Figure 4 displays that this feature comes from how the modules are built (see Appendix A Figure 7 for the plot showing this and see Szumlak's article for build details). Through this build feature of the modules it is potentially possible that one particle is detected twice within a single module. Because less than 1% of the track hits in the data have a same track neighbour in the same module this potential pitfall will not be respected in the algorithm design.

Dont understand this:
- they spoke about Z coordinates (position of the 52 modules in the beam axis), left and right should correspond to that but then they spoke about odd and even numbering which correspond to the X COORDINATE NOT THE Z
- Here, the odd numbers on the x axis are the left side (in a module) and the even numbers on the x axis are the right side (in a module), Is this graph correct?

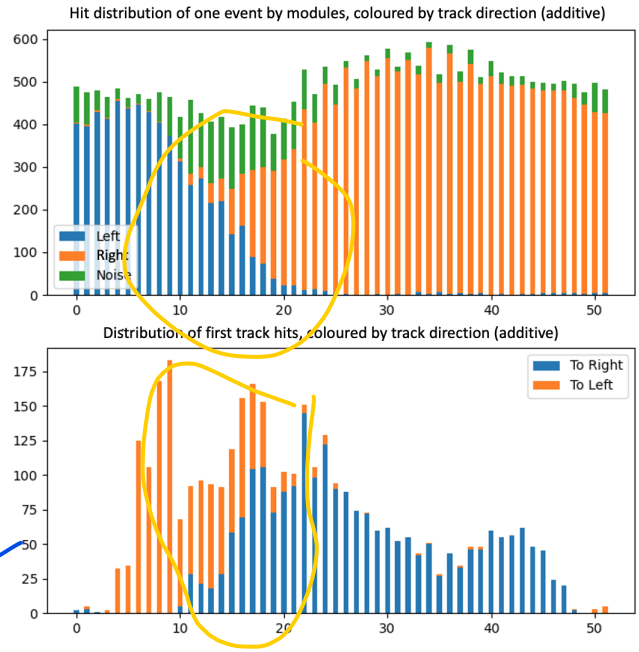


Figure 5: Distribution of hits over the detector modules of multiple events

This is a plot for the positive or negative Z direction (results kind of obvious: impact point between modules 10 and 20, then just particles direction after impact)

In Figure 5, the distribution of the hits of multiple events over the 52 detector modules is plotted as bar chart. In the upper plot the hits are coloured on the criterion whether the particles they belong to went into positive (Right) or negative (Left) Z direction. Furthermore, hits that are interpreted as noise are coloured in green. Interesting in this plot is that only the modules between modules 10 and 20 have a significant amount of hits from trajectories in both directions. This indicates that most of the primary vertices (collision points) must be in that area as well. Between modules 20 and 40 a notable pattern occurs which supports the previous insight that odd modules cover a different region than the even modules. It is notable that the even modules show more hits than the odd modules, meaning that more particles were detected from the even than from the odd modules. In the lower plot, the distribution of the first detected hit per track is plotted. Again the region from module 10 to 20 is the only region where first hits from tracks that go in both directions are present. The spikes in the distribution of these first hits (modules: 8,9,16,17,22,24) are explainable by different distances between the modules.

Noise ?

6 Methodology

This section explains the four different approaches used: Hopfield networks, clustering, template matching and sorting algorithms. Hopfield networks (Section 6.1) models the track reconstruction problem as an en-

ergy minimisation problem where the stochastic search for a global solution is guided by the network's weights that indicate inter segment compatibility. Clustering (Section 6.2) seeks to reconstruct the tracks by minimizing distances between the points that belong to the same track and maximising the distance between any two points that belong to different tracks. Template matching (Section 6.3) looks for patterns in the polar angle of hits in consecutive modules with subdivisions in the range of values to control the allowed angle difference. Finally, Sorting Algorithms (Section 6.4) are a group of methods that address the track reconstruction problem by traversing a sorted list of the hits in the event.

6.1 Hopfield Network

The Hopfield network based approach is modelled similar to the ones presented in the relevant literature (Stimpfl-Abele & Garrido, 1991; Passaleva, 2008). The neuron activations are in the range 0 to 1 and the neurons are updated in a randomised sequence. The problem considered by (Passaleva, 2008) concerns five detectors and the direction of the particle trajectory is known. In the problem at hand there are 26 modules per network and the track direction is not known, thus the approach needs to respect trajectories in both directions of the beam-line. Because of the detector specifics discussed in Section 5.2, one instance of the problem can be modeled using two disconnected Hopfield networks, one reconstructing the tracks from all hits in the even numbered modules and one reconstructing the tracks from the hits in the odd numbered modules.

Neurons are modeled as the track segment connecting two hits in consecutive modules. From the overview in 6, where M denotes the number of modules, it can be deduced that there are $M-1$ neuron layers. Each pair of neurons (n_i, n_j) in neuron layers k and $k+1$ respectively has a weight w_{ij} associated with it.

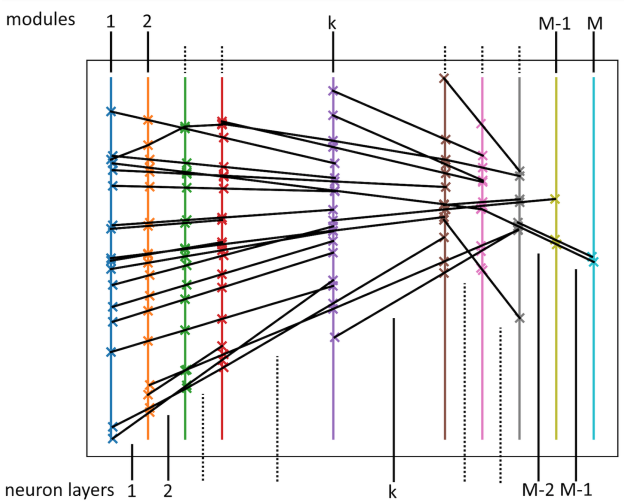


Figure 6: Schematic 2D representation of a problem instance with neuron layers and weights of the Hopfield network implementation

The model seeks to reconstruct particle tracks by converging to a neuron activation configuration that lies in a local minimum of the networks energy function. The Network energy is given by:

$$E = -\frac{1}{2}\Big(\sum_{k=1}^{M-2}\Big(\sum_{i,j\in C_k} n'_i * n'_j * w_{ij} - P_k\Big) - P_{M-1}\Big) \quad (1)$$

where n'_i and n'_j denote the activations of neurons n_i and n_j. C_k denotes the set of neurons modelling the segments which share a hit in module $k+1$. Finally, P_k denotes a penalty term that targets the prevention of bifurcations in neuron layer k. Bifurcation means that a single hit has multiple activated incoming or outgoing segments, but tracks can not have this. Intuitively the energy will be low when there is no bifurcation, and the activated hit-connected neurons share a high weight, thus model segments that are likely to be on the same track.

The neuron update function used to lower the networks energy towards convergence is given by:

$$n'_i = \frac{1}{2} * (1 + \tanh(\frac{1}{T} * U_i - \frac{B}{T} * P_{n_i})) \quad (2)$$

Where U_i models the alignment of a segment (neuron) i with the hit-connected segments modelled by the neighbouring neuron layers. This is calculated by summing over all connected neuron activation's n'_j times the weight defined by $w_{i,j}$. P_{n_i} is the penalty term measuring the bifurcation and calculated by summing up the neuron activations of neurons that model segments parallel to the concerned segment and that share one hit

on either side. T and B are hyper parameters that can be tuned in order to alter the networks behaviour. T influences how extreme the update function should be in terms of promoting potentially good connections and penalising bifurcation. B influences how much bifurcation should be penalised in general.

The weights are set during the initialization of the network and should be high for neurons that belong to the same track. To achieve this the weight between neurons i and j is defined as follows:

$$w_{ij} = \max(0, \alpha(1-\sin\theta_{ij})^{\beta}(1-\sin\varphi ij)^{\gamma}+\Lambda_{ij}) \quad (3)$$

where θ is the absolute difference of angles in the XZ plane between two segments. φ is the absolute difference of angle in the YZ plane. α, β, γ are hyperparameters that have to be chosen. This first part of the weight is large if the difference in angles is small. It was used by Passaleva (2008), and worked well for the addressed reconstruction problem. For the instances at hand the network performed better when adding an additional term to the weights:

$$\Lambda_{ij} = \max(-\delta, \min(\delta(-2(\tanh(c_i - c_j) * \eta)^2), \delta)) \quad (4)$$

with

$$c_i = \frac{\sqrt{(x_l - x_r)^2 + (y_l - y_r)^2)}}{\sqrt{(z_l - z_r)^2}} \quad (5)$$

where subscripts l and r denote the left and right hit of neuron n_i respectively. Essentially, Λ_{ij} is large if c_i and c_j are similar and η (≥ 1) controls how similar these values should be to increase Λ_{ij}. δ controls how large the effect of Λ_{ij} is on the weight calculation and clamps the value between $[-\delta, \delta]$. A second variant of Λ_{ij} (referred to as Λ^*_{ij}) uses

$$c_i = \frac{\sqrt{(x_r + y_r)^2} - \sqrt{(x_l + y_l)^2}}{z_l - z_r} \quad (6)$$

both of which are a useful addition to the weight calculation. This is because both versions of c_i are similar for neurons within the same track. Weights are bound to positive values as the original formula for weights as used by Passaleva (2008) could not be negative, but Λ_{ij} and Λ^*_{ij} may cause negative weights which performs significantly worse. Due to this, the total weight matrix is in fact extremely sparse, as over 99.9% of its entries are 0. However, this fact is not abused in the implementation but further discussed in 9.1.

The neurons are initialized randomly in $[0, 1]$ which allows the network to find different solutions for different initialisations. To avoid a local minimum, the network is run a number of times before using the final neuron states to extract tracks. In each run, the neuron states are re-initialized and all the converging states are stored. They are then processed (eg. by choosing the states with the lowest network energy or averaging over the states) to yield the final neuron states.

After the convergence of the network, the neuron activations must be converted to tracks. It is not easy to control the network perfectly and thus bifurcation regularly appears in the final configuration. Before extracting tracks all bifurcations must be removed. This can be done by different methods. The considered methods are *max activation* and *smart removal*. In *max activation* the network is scanned for bifurcation segments and the most activated neuron survives while the other is deactivated. In *smart removal* the neuron that shares the highest weight with any of the active, hit-connected neurons in the neighbouring neuron layer is selected into the final solution, while the other bifurcating neurons are discarded. Finally, there is a sub algorithm that translates segments to tracks. This algorithm can split long tracks based on some criteria that are connected to the aforementioned line-constants.

6.2 Clustering

For the Clustering approach the following algorithms have been considered: Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) using the hdbscan library (McInnes, Healy, & Astels, 2017), Affinity Propagation, Agglomerative Clustering and Mean shift with the scikit-learn library (Pedregosa et al., 2011).

The data has been clustered in the spatial space formed by the three polar angles φ, θ and ψ, and with a custom distance metric. The Mean shift algorithm only ran on the angles space while the other algorithms were tested with both approaches.

In the case of clustering in the polar angle space, because we are dealing with straight lines, the angle values of hits from the same track are similar. For this reason density areas are formed by hits of a same track when the hits are represented in a three dimensional space with polar angles as coordinates. The Mean shift clustering algorithm locates the maxima of the polar angle density function (Cheng, 1995). HDBSCAN is also able to find varying densities.

A pre-computed distance matrix built with the custom distance metric stores the distance between every pair of hits in the event. Even thought different implementations of the distance metric have been tested they all

share a similar intuition. Those hits on the same module are set to have a high distance, and hits on different modules the distance is computed as the difference between the polar angle ϕ of both hits. Some of the tested distance metrics also consider in the euclidean distance of the hit's z coordinate, as well as the difference between the modules.

6.3 Template matching

For the specific problem at hand, a template takes the form of a set of pixel hits in a set of detector modules that are sufficiently close to a line. Note that there might be tracks where not every consecutive detector module has a hit. This is unlikely to occur more then once in a track, due to the high hit efficiency of the detector. As mentioned before, a line can be expressed by means of angles, where hits of the same track have similar angle values since we deal with straight lines. Therefore, the assumption that hits in a track have a very similar polar angle in the XY plane is made.

Five template matching algorithms were implemented, with increasing complexity. First, the 2π range of the polar angle in the XY plane is split into a number of templates (e.g. 900), which determines how large the angle difference is allowed to be. Then, a 2D matrix is created with the columns representing the modules and each of the rows representing one template based on the polar angle. All hits are added in the array at the position consistent with its polar angle and module number. This array is then used to find tracks by looping over the module numbers for the different angles.

- The most basic implementation, *TM1*, finds a track when there are at least three hits in subsequent modules, and allows for skipping one module in between.
- In *TM2*, there is a monotonicity check included which checks whether the hits in the track move away from the beamline. This ensures that tracks with an illogical direction w.r.t. the other hits in the same track, are eliminated (e.g. when the third hit is closer to the beamline than the second hit). This monotonicity check is also included in *TM3*,*TM4* and *TM5*.
- *TM3* also allows for finding a hit in a template or row below or above the template under examination. This means that it is more flexible in terms of the allowed difference in angles between consecutive hits of a track (e.g. instead of only allowing for hits with a polar angle between 1°- 2° in a track, it now also allows hits with an angle between 0°-3° in the following modules if no hit between 1°-2° was found).
- In the TMs up till now, if a hit has the same module number and angle as a previous hit, the latter will be overwritten. In *TM4*, a 3D matrix is introduced to fix this, such that several hits with the same module angle combination can be saved and used for finding tracks.
- Finally, in *TM5*, the same matrix as in *TM4* is utilized, however the four neighbouring templates are considered (instead of just two neighbours as before), thus allowing for even more difference in the angle between consecutive hits in a single track.

6.4 Sorting Algorithms

In the research for viable representations of the data in which the tracks are easily distinguishable, it was found that it might be possible to solve the reconstruction to a large extend by sorting based on the polar angle in the XY plane alone. This would be a very quick method with a time complexity of O(n log n) and hence it was investigated.

The first sorting implementation is a method based solely on phi and the criteria for combining hits into a track would be that the difference in polar angle is less than a certain value, that the hit is at most 4 modules away from the track, that a hit's module number would be odd or even, equaling the hits it is to be combined with and that at most 1 hit could be missing. This method, see algorithm 1, achieves 80,5% VELO efficiency, indicating that sorting has potential. But with a ghost rate of 22,1% refinement is needed. Further implementations were created by first grouping into viable sets and then finding tracks in this to reduce the ghost and clone rate. But these methods reduced efficiency to below 60% and did not improve ghost rate, so they were not investigated further.

To bring down ghost and clone rates, it is helpful to include more information than just the angle and module number. Also, comparing a hit to multiple tracks could improve the results. To this end, three measures expected to be constant for hits in a track were tried:

1. The Euclidean distance between two hits in x and y divided by the difference in z

Algorithm 1: Basic-Angle-Search

```
Data: x,y & z coordinates of pixel hits in an event
Sort the hits by polar angle θ_xy w.r.t. to the origin
  Initialize a current track, and add the first hit to this
  track skipped = false
for hit i in sorted list after the first hit do
    if difference in polar angle between hit and first/last
     hit of the current track is less than 0.005 then
        if module number hit i %2 == (module number
         hit i+1) % 2 then
            add to current track
        if absolute value (module number i - module
         number i+1) == 2 then
            add to current track
        if absolute value (module number i - module
         number i+1) == 4 and skipped = false then
            append to track, set skipped tracker to true
    if difference in polar angle between hit and first/last
     hit of the current track is greater than 0.005 then
        if current track >= 3 then
            store it as a track, then start a new track
              using next hit
        else
            discard current track, start a new track
              using the next hit
return the stored tracks
```

2. The increase/decrease in polar distance in the XY plane between two hits as a function of their z coordinates

3. The direction vector, i.e. the 3D vector of the differences in x, y and z between two hits.

Using these measures it was possible to be less strict on module number difference and number of skipped modules. The angle difference was still used as a way to initialize tracks. Now if a hit did not have the right module number or a larger angle difference, one of the constants was used to establish whether a hit fits the current track or a fixed number of previous tracks. The results for the first constant were not good (40% physics efficiency and high ghost and clone rate). This may be in part due to the fact that the constant is indeed constant to hits in the same track, but not exclusively so. The second constant, referred to as monotonicity constant, turned out to be a better performing method, with roughly 64 % VELO physics efficiency and a ghost rate of 5.6%. The third method with the direction vector however, allowed us to find 66% of VELO tracks with a lower ghost and clone rate (3,6% and 2,8% respectively, after clone reduction).

The direction vector seemed to work so well that a third approach was implemented, see algorithm 2. Here the direction vector between two hits in the sorted list is used rather than the polar angle difference. The algorithm checks x hits before and after the two hits that were used to construct the direction vector to see if these align with the direction vector. This method achieves the best results, which will be discussed further in the next section on the results. More details on performance of the other methods will not be further discussed for brevity's sake.

Algorithm 2: Basis-Direction-Vector-Search

```
Data: x,y & z coordinates of pixel hits in an event
Sort the hits by polar angle θ_xy w.r.t. to the origin
for hit i < number of hits in event do
    if hit i has not yet been used and hit i+1 has not
     been used then
        if module number hit i %2 == (module number
         hit i+1) % 2 then
            if absolute value (module number i -
             module number i+1) < = 4 then
                compute direction vector for hit i and
                  i+1
                for j <= 6 do
                    check if hit i-j or i+1+j matches the
                      direction vector
                if there is at least one hit that matches
                  the direction vector then
                    store i,i+1 and the matching hits as
                      a track.
                    flag these hits as used
return the stored tracks
```

7 Results

In this section the performance of the different approaches in varying settings is discussed. [1]

7.1 Hopfield Network

Due to implementation weaknesses, the experiments can only be conducted on instances where the number of neurons per layer is lower than 2200 (439/995 - minibias, 278/1000 - bsphiphi). The chosen implementation considers all possible segments between two modules, thus resulting in a complexity of $O(N^2)$ for

[1] Implementations and experiment results: `https://github.com/ctelloordonez/velopix_tracking-master.git`

the initialisation and update steps, where N is the maximum number of hits in any of the modules. The infeasibility of the algorithm for big instances arises from the implementation of the weight matrix. The naive implementation has a weight for all combinations of neurons in neighboring layers, letting the weight matrix size grow in $O(N^4)$, which may exceed the RAM storage of the test machine for large events.

In general, the Hopfield network converges after 10 to 35 iterations on any of the tested instances. The processing time is bound to the N^2 complexity, thus increases quadratic with the size of the instances. In the non-optimised implementation, the processing time of one instance takes from a few seconds to a few minutes, which is a very weak time performance when comparing it to the other approaches. However, this also leaves space for further research, as further discussed in Section 9.1.

The algorithm implementation allows for tuning of numerous parameters, divided into method-based and numerical parameters. However, only the most impactful ones will be discussed hereafter. First, a general description of the chosen setup for experiments. The network's initial state is set to randomised neuron activations in the range $[0, 1]$, after which the network converges to a local minimum. After repeating this process 10 times, the neuron's activations of the convergences with an energy level below the mean energy are averaged. Once a final configuration is found, tracks are extracted and bifurcation is removed using *smart removal*.

For the weight calculation the parameters were chosen to be α = 1, β = 10 and γ = 10 same as chosen by Passavela (2008). The addition of both Λ_{ij} and Λ^*_{ij} to the weight calculation heavily improves performance. If either is left out, the network performs significantly worse. δ in both calculations is set to 0.9, as this sufficiently corrects misguided weights as set by α, β and γ without mitigating their effect.

Averaging over the neurons' activations of convergences with an energy level below the mean energy was shown to yield the best results. However choosing convergences with an energy below the median or averaging neurons activations of all convergences had comparable performance. Choosing the convergence with the minimum energy performed considerably worse.

With respect to temperature and bifurcation penalties (parameters T and B), there is no improvement by decaying these parameters with the number of full update step iterations. Choosing a low value (e.g. 1e-8) for both seems to give the best result at the experiment phase. Later experiments show that choosing $B = 100{*}T$ can yield a slightly increased performance. Meaning a high promotion of the bifurcation penalty in the update function can help to improve performance.

The selection of the activation threshold, which controls which neurons are considered active, results in a trade-off between reconstruction efficiency and the ghost rate. A small activation threshold yields better performance but at the same time increases the ghost rate. The best performance seems to be in the region of 0.1 to 0.3 for this parameter.

In regards to the extraction of tracks, the pruning threshold, which controls the removal of unexpected track segments, provides an interesting trade-off. A high threshold increases ghost rates and reduces clone rates. Small deviations can have a large effect on its performance, and it is very sensitive to the activation threshold, so fine-tuning is required. For the chosen activation threshold, 0.01 is found as best value for mitigating ghosts, whereas 0.05 is found to provide a good alternative with fewer clones.

	Latest version (23-06-2021)			
Particle category	Reco. eff	Clone fraction	Hit purity	Hit eff.
VELO	75.0	10.53	99.56	82.93
long	92.5	17.58	99.51	79.77
long $>$ 5GEv	95.0	18.41	99.45	79.97
long strange	85.8	10.06	99.48	88.52
long strange>5GEv	90.4	5.88	99.28	90.43
Overall fake fractions	1.9			

Table 2: Validation output of the Hopfield network for 100 events over the minibias data set with pruning threshold 0.01.

In Table 2 above, the validation statistics of the algorithm with the optimal parameter configuration found through the experiments is presented. Compared to the other tested methods of this research the Hopfield network method performs at average for the VELO category (which involves all tracks). For the other categories it shows comparably high reconstruction efficiencies but at the cost of high clone fractions. The ghost rate of the tracks found by this method is at average of the tested methods. Another test presented in Table 13 with the pruning threshold set to 0.05 shows a slightly higher reconstruction efficiency but at the cost of an increased ghost rate. When considering the test on the BsPhiPhi dataset presented in Table 14 the performance does not vary much but the reconstruction

	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
Mean shift	66.1	18.36	98.54	69.62	78.4
HDBSCAN	55.5	3.99	93.66	89.68	28.5
Agglomerative clustering	62.6	2.49	98.4	83.17	81.7

Table 3: VELO performance indicators obtained by clustering methods on polar angle space

	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
HDBSCAN	25.9	3.08	87.04	86.14	60.2
Agglomerative clustering	52.60	12.23	89.94	74.34	47.60

Table 4: VELO performance indicators obtained by clustering methods with pre-computed distance matrix

efficiency appears to be slightly lower. Interestingly, 'fromB' tracks appear to have a relatively high reconstruction efficiency.

7.2 Clustering

The clustering algorithms are tested with pre-computed distances and on the polar angle space, besides Mean shift which only runs on the polar angle space. Tables 3 and 4 show the best results of the VELO tracks performance indicators obtained by each clustering algorithm. These results show a low reconstruction efficiency in general, below 60%, with relatively high clone and ghost rates.

Besides the overall poor performance of the clustering algorithms, Affinity Propagation does not converge for most of the events. Neither for clustering in the polar angle space nor with the pre-computed distance matrix. Because of this, it can not find (almost) any track, which is the reason why these results are not presented.

7.3 Template Matching

For template matching, the most important parameter to tune is the number of templates. Before the different template matching methods are compared, an adequate number of templates should be determined. Among the performance metrics, the main aim is a low ghost rate and a high reconstruction efficiency. Overall, 900 templates shows a good balance between ghost rate and reconstruction efficiency. Table 5 shows the performance metrics for method 2 on the BsPhiPhi dataset. The other metrics can be found in the Appendix. 900 Templates does not always guarantee both the lowest ghost rate and the highest reconstruction efficiency, since for all methods there seems to be a trade-off between these metrics. However, 900 templates strikes a good balance. It is important to note that no exhaustive search of all possible template numbers was conducted as this was outside the scope of this project. The results for all methods show that the clone fraction positively correlates with the number of templates while ghost rate negatively correlates with this parameter. Therefore, if the objective is to decrease the ghost rate, this can be achieved by using a higher number of templates (e.g. see the last row of Table 7 for 2000 templates). If the objective is to reduce the clone rate, a lower number of templates can be used to achieve this.

Templates	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
700	64.6	17.2	97.1	67.5	13.3
800	63.5	18.3	97.2	65.8	13.1
900	62.7	19.3	97.4	64.2	12.5
1000	61.8	19.9	97.6	62.9	12.1

Table 5: VELO performance indicators obtained by template matching method 2 on the BsPhiPhi dataset using 800 vs 900 vs 1000 templates

In method 4 and 5 of template matching, several instances can be saved for each template-module- combination, such that instances with the same combination are not overwritten as is the case in the methods 1 to 3. This also allows more than one instance to be checked for monotonicity, which increases the probability of finding a fitting hit for the track in question. A comparison between 2 and 3 instances kept per cell is shown in Table 6 for method 4, and in Table 7 for method 5, both with 900 templates. First, using 3 instances increases the processing time, which is a disadvantage for a method in real-time usage. Furthermore, since a fine grid of 900 templates is used, we noticed that the third instance is rarely ever used. Finally, the Tables 6 and 7 show that the balance between a low ghost rate and a high reconstruction efficiency is most adequate with two instances kept. Thus in the following the fourth and fifth method with a maximum of 2 instances per cell will be used. Note that the preferred amount of instances is negatively correlated with the number of templates used (e.g. when less templates are used, more hits will have the same template-module-combination, requiring a higher number of instances per combination).

Since 900 templates shows good performance for all template matching algorithms, the different methods are compared in Table 8 using this number. Overall, the results show that reconstruction efficiency and clone

Instances saved	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
2	64.5	19.1	97.7	65.3	12.0
3	70.1	46.9	96.7	68.3	14.4

Table 6: VELO performance indicators obtained by template matching method 4 on the Minibias dataset using 900 templates and 2 vs 3 instances per cell

Instances saved	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
2	70.6	46.1	97.0	69.1	13.1
3	64.2	19.7	97.5	64.5	13.1

Table 7: VELO performance indicators obtained by template matching method 5 on the Minibias dataset using 900 templates and 2 vs 3 instances per cell

fraction are positively correlated. We see that the reconstruction efficiency increases steadily with the complexity of the methods (from the top to the bottom). Furthermore, the more complex methods achieve overall higher clone fractions and lower ghost rates. The highest reconstruction efficiency and hit purity is obtained using *MT5*. With 900 templates, the lowest ghost and clone fractions are obtained using *MT3* on the Minibias dataset. Note that the ghost rate can also be reduced using a larger template number.

7.4 Sorting-based Algorithms

For the direction vector based searching two versions are implemented: version one, where the direction vector is calculated only if two consecutive hits in the sorted list have not been used, and version two in which, if one of two consecutive hits was previously used, a third hit that is not yet used is found and used to calculate a direction vector with. Version two shows a significantly higher ghost rate for little to no physics efficiency gain. Hence, presented below is only the performance of version one. A method more flexible regarding which hits are considered could be able to do

Template Matching Method	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
TM1	61.6	16.8	94.5	67.5	20.9
TM2	63.0	18.7	97.6	65.0	11.5
TM3	64.2	16.7	97.6	68.3	10.5
TM4	64.5	19.1	97.7	65.3	12.0
TM5	70.6	46.1	97.0	69.1	13.1

Table 8: VELO performance indicators obtained by template matching methods on the Minibias dataset using 900 templates

#	1	2	5	10	80
efficiency	87.0	92.08*	94.6 *	95.9 *	98.5 *

Table 9: Percentage of tracks that have two hits that are at most 4 modules apart within a certain number (#) of hits from each other in the sorted array. A * indicates that the result was obtained on a dataset of 10 events, because the test for maximum possible physics efficiency creates an unworkable amount of ghost tracks on full data sets.

	Latest version (20-06-2021)				
Test	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
Balance	76.5	3.92	99.70	82.15	1.7
low ghost	70.7	5.78	99.78	75.90	1.4
low ghost	70.5	5.76	99.81	75.80	1.0
$<= 2\%$	78.5	4.26	99.63	84.24	2.0
$<= 2\%$	78.5	19.40	99.63	74.23	1.8*
max reco	81.7	19.59	99.54	77.06	2.2*
max reco	82.8	23.48	99.2	72.65	3.5*

Table 10: VELO performance indicators obtained by Direction Search version 1 on minibias. (* - without clone reduction, this rate is artificially lower.)

better than the method presented below. This claim is based on the test results in Table 9, where it is shown that the number of tracks that could be found is significantly higher when considering looking at a certain number of hits after a hit i, not just the first hit after i.

The primary goal is a high efficiency combined with a low ghost rate. We set the targets at 70% physics efficiency and a ghost rate lower than 5%. Low clone rate and high hit efficiency are secondary priorities because they are easier to fix at a later stage. A clone reduction method is implemented, albeit not a sophisticated one.

Because there are multiple metrics, multiple tests are performed to display potential for these metric: The balance test seeks to have a good balance of all rates. The low ghost tests focus on minimizing the ghost rate. $<= 2\%$ tests aim to maximise reconstruction while keeping the ghost rate withing the specified limit. Finally, the max reconstruction tests aim at maximizing physics efficiency of the algorithm, disregarding the other metrics. This is done in order to test how good the method can perform if the other metrics can be reduced at a later stage. The results of our test can be found in Table 10.

Because of the special interest in b-star to $\phi\phi$ decay, Table 20 in Section D of the Appendix contains the VELO

	Latest version (21-06-2021)			
Particle category	Reco. eff	Clone fraction	Hit purity	Hit eff.
VELO	76.1	6.40	99.68	79.45
long	85.8	10.34	99.72	74.64
long $>$ 5GEv	86.8	7.93	99.76	76.43
long strange	38.7	11.77	99.77	63.31
long strange$>$5GEv	43.6	11.45	99.82	65.44
long fromB	84.2	8.86	99.67	76.22
long fromB$>$5GEv	86.4	7.89	99.67	76.40
Overall fake fractions	1.7			

Table 11: Validation output obtained by Direction Search version 1 for the BsPhiPhi data set.

results on the data set biased to this particular decay. The results are quite similar. The results are slightly lower, possibly due to the fact that the reconstruction efficiency for long tracks from B is slightly lower than for ordinary long tracks, as can be seen in Table 11. Most noticeable in the full validator is the low performance on so called strange tracks. This is consistently the case on both data sets and with different settings. It would appear that there is something to these tracks that makes them hard to find using the sorting methods considered.

The throughput rate of the algorithm is about 20 events per second when executed with clone and ghost reduction, or 30 to 40 events per second without. This is when run on an Intel i7-5500U CPU (2.4 Ghz) and implemented in Python.

7.4.1 Parameters

There are 6 main parameters: 3 to tune the tolerable deviation of the direction vector, one to set the number of modules two hits can differ to be considered for initiating a direction vector, one to set the accepted deviation from monotonicity to combine tracks for clone reduction and one to set the number of hits the algorithm will investigate away from the initial two hits that form a direction vector. When it comes to ghost rate and physics efficiency, the most powerful ones appear to be the 3 parameters that determine tolerable deviation from the direction vector. For the ghost and clone reduction, they can either be used or not. For the clone reduction it can be done multiple times, and a parameter can be set so that the number of tracks it checks is limited. The correlations between parameters and metrics is given in Table 12. The ranges tested can be found in Table 21, Section D of the Appendix.

Based on these results, the method can achieve good

Test	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
XtoY	+	-			+
YtoZ	+	-			+
XtoZ	+	-			+
Clone	-	+		+	+
ModDif					+
LookAround	+				
GhostReduc	-				-
CloneReduc	-	-			+
CloneRedTimes	-	-			-
CloneRedCheck					

Table 12: General Parameter change correlations on metrics (if an effect was observed).Some varied within the range,and the relations are not necessarily true true across all possible values for the parameters.

performance on low ghost and high physics efficiency rate at this stage of development. It would likely be good enough to locate the collision point(s). As for finding all the tracks, further refinement is needed. To this end, it will be worthwhile to improve hit efficiency, to make any looking for tracks at a later stage endure less noise. At time of writing, the implementation is not optimized in this metric, and improvement is expected to be possible. How close to SOTA performance this algorithm can be taken is still to be seen. For suggestions on future research & improvements, see Section 9.4.

8 Conclusion

1. *What properties of tracks, events and the detector can be incorporated in the proposed track reconstruction techniques for better performance?*

   The polar angle in the XY plane is a good basis for an algorithm, and more generally angles tend to be useful. Furthermore, several mathematical metrics based on the geometry of three dimensional lines were used in the algorithms. Finally, some fairly standard knowledge such as the number of modules, how many modules a track can skip and so forth were used.

   In the data analysis in Section 5.2 it was discussed that the data shows that most primary vertices occur between modules 10 to 20. Further, the data showed that almost no tracks in negative Z direction are detected in modules 20+. Analogous no tracks that go in positive Z direction are detected in modules $<$10.

   For a potential future implementation of the Hop-

field Network this insight could be helpful to implement the method more efficiently. This can be done by understanding that all collision particles fly away from the beamline (the point where the primary vertices are) and thus the hits of one track need to have increasing polar distances if ordered by detection time. This means that the considered segments in the Hopfield network could be largely reduced for all hits in the modules >20 and <10. This might however hinder the Hopfield network to detect decay track in these regions.

2. *Which way(s) of dividing the problem into subproblems help to reduce computational complexity while maintaining high physics efficiency potential?*

   The effect of binning hits by their polar angle in the XY plane has been considered. It has shown an increasing percentage of tracks with hits falling into different bins when the amount of bins is also increased. Therefore these problem division will require a post processing step to combine fragments of the same track that felt in different bins.

   The Hopfield network splits the instances into segments with both hits in even or odd modules, but the effect in the physics efficiency of this subdivision has not been studied and therefore can not be stated to be a robust approach.

   A potentially useful aspect that was not incorporated is the fact that the vast majority of tracks have a hit very close to the proton beam line, which might help in reducing the number of hits that need to be taken into account in future methods.

3. *How do the track reconstruction techniques considered in this project compare to the SOTA with regard to the physics metrics?*

   On Hopfield Network: The Hopfield network is an applicable approach for the track reconstruction problem but the performance does not reach the SOTA level.

   The main challenges are the high number of parameters to tune and the difficulty of debugging due to the complexity of the network.

   The current implementation also requires excessive amounts of RAM and too many detected hits result in large computation times that are not feasible in practice. While the optimization of the implementation is definitely possible we do not believe it can be fast enough for an online implementation within the VELOPIX. Nevertheless, the experiments show that the Hopfield network can be used to detect particle tracks on small problems fast.

   On Clustering: The results of clustering a list of hits for track reconstruction do not reach an admissible performance threshold and can not compare to the SOTA. The low performance of this approach can be due to the chosen clustering space but also the distance metric that does not capture the similarities between hits of a same track. These two ideas are further discussed in the future research.

   On Template Matching: Template matching does not reach the level of performance of the SOTA. However, it is a fast simple method that can be a starting point to remove the most obvious tracks before using another algorithm that can handle the remaining tracks. The results show that more complex methods (*TM3, TM4, TM5*), perform better in terms of physics efficiency and ghost rate, however, result in a higher clone rate. A higher number of templates seems to reduce the ghost rate, but increases the clone rate. 900 Templates results in good performance, although an exhaustive search was not performed and is a possible starting point for further research. We found that keeping two instances per cell can increase the performance but also increases processing time.

   On Sorting Methods: The results are interesting, albeit not yet at SOTA level. Out of the attempts, searching by direction vector does best. We think it can perform better than the current approach, and out of the methods investigated it is the one we suggest for future research.

9 Further Research

In this section further research starting points will be suggested based on the different approaches.

9.1 Hopfield Network

Future research with respect to Hopfield networks for track reconstruction could go in different directions:

The physics efficiency could potentially be improved by doing further work on the energy and update function. This is mainly due to the fact that bifurcation is still very much present in a converged network, whereas optimally this would not be the case. Another limitation to the physics efficiency of the presented implementation could be the splitting of one instance in even-even, odd-odd module instances. By considering also even-odd tracks the physics efficiency could be improved. Additionally finer parameter tuning could be conducted (e.g. α, β, γ).

The key problem of computationally complexity can be largely improved by using more efficient implementations. This concerns the number of neurons created (e.g. not creating a neuron for every hit pair in consecutive modules), the implementation of the weight matrix (sparse implementation is possible due to extremely large number of zeros, to avoid RAM errors) and an improved update implementation.

Additionally, one could try to cut the problem instance by dividing it into sub instances based on the polar angle, which reduces the number of neurons naturally in a quadratic way. The problem of cutting some tracks could be addressed by using overlapping polar regions and clone reduction (see Figure 11). Perhaps the notion that most tracks have a hit close to the beam line as indicated in Figure 10 in the appendix could be utilized as well.

A final direction of further research could be to model triplets as neurons, since 3 hits fitting neatly on a track is a solid indication of there being a real track. Here it would be even more important to elaborately limit the number of neurons to not create O(n^3) segments per 3 modules. Using triplets might be too similar to search by triplet, which is something to take into consideration.

9.2 Clustering

This project has focused on applying clustering techniques on the hits themselves. For future research it is may be interesting to cluster the track segments formed by the connection of two hits. In this case the affinity measure could take into account segment information such as the direction vector or the constant explored in the sorting algorithms of this project.

Key will be finding a correct measurement of distance and an accurate representation space. To this end, different spaces may be explored. Given that so many angle spaces did not work, perhaps a distortion of angle or x,y,z space is needed. Key to any distance in a projection to the same x,y plane will be to balance the difference in polar angle and the difference in polar distance created by a projection.

9.3 Template Matching

An extensive search of the number of templates would be an advised starting point for further research into template matching. Furthermore, it would be interesting to combine template matching with another method, where template matching can function as a first step for fast and simple track reconstruction.

Another option would be to look into the possibility of direction vector based templates. Also possible is to include more information, such as polar distance or angles.

9.4 Sorting-based Algorithms

Further research can be done on a different theta-sorting implementation, to see if it can be done more accurately, especially a lower ghost rate. For this, new criteria for what constitutes a track and how to find it might be needed. However, for now we shall focus on ideas for direction vector search.

1. Implement a better method that looks at multiple hits for optimal direction vector
2. Parameter experiments to find optimal settings for different targets
3. Parallelization and speed optimization.
4. Better ghost and clone reduction, ours is very simple and add things together that should remain separate tracks.
5. Improve hit efficiency. Somehow the algorithm gets low hit efficiency rates. Figure out whether this is fundamental to the approach or a matter of implementation improvement. Important will be to control the relation between allowed deviation from the direction vector.

make direction vector search look ahead further if the last hits it looks at match the

References

Ackerstaff, K., Airapetian, A., Akopov, N., Amarian, M., Andreev, V., Aschenauer, E., ... Zohrabian, H. (1998). The hermes spectrometer. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *417*(2), 230-265. Retrieved from https://www.sciencedirect.com/science/article/pii/S0168900298007694 doi: https://doi.org/10.1016/S0168-9002(98)00769-4

Bogdan, T., & Cámpora Pérez, D. H. (2016, september). A clustering algorithm for the velopix tracking..

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, *17*(8), 790–799.

Cámpora Pérez, D. H., Neufeld, N., & Riscos Núñez, A. (in press). *Search by triplet: An efficient local track reconstruction algorithm for parallel architectures.* Elsevier BV.

Denby, B. (1993, July). The use of neural networks in high-energy physics. *Neural Computation*, *5*(4), 505–549. Retrieved from https://doi.org/10.1162/neco.1993.5.4.505 doi: 10.1162/neco.1993.5.4.505

Frühwirth, R., Regler, M., Regler, R., Bock, R., Grote, H., & Notz, D. (2000). *Data analysis techniques for high-energy physics*. Cambridge University Press. Retrieved from https://books.google.de/books?id=2mq94rdFd3gC

Hopfield, J. J. (1982, April). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, *79*(8), 2554–2558. Retrieved from https://doi.org/10.1073/pnas.79.8.2554 doi: 10.1073/pnas.79.8.2554

Hulsbergen, W. (2007, 3). *pattern recognition.* https://www.nikhef.nl/~wouterh/topicallectures/TrackingAndVertexing.

Koch, N., Kolander, M., Kolanoski, H., Siegmund, T., Bergter, I., Eckstein, P., ... Weseler, S. (1996). The argus vertex trigger. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *373*(3), 387-405. Retrieved from https://www.sciencedirect.com/science/article/pii/0168900296000691 doi: https://doi.org/10.1016/0168-9002(96)00069-1

Mankel, R. (2004, 3). Pattern recognition and event reconstruction in particle physics experiments. *Reports on Progress in Physics*, *67*, 553–622.

McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, *2*(11), 205.

Passaleva, G. (2008, October). A recurrent neural network for track reconstruction in the LHCb muon system. In *2008 IEEE nuclear science symposium conference record.* IEEE. Retrieved from https://doi.org/10.1109/nssmic.2008.4774532 doi: 10.1109/nssmic.2008.4774532

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Piucci, A. (2017, jul). The LHCb upgrade. *Journal of Physics: Conference Series*, *878*, 012012. Retrieved from https://doi.org/10.1088/1742-6596/878/1/012012 doi: 10.1088/1742-6596/878/1/012012

Poikela, T., De Gaspari, M., Plosila, J., Westerlund, T., Ballabriga, R., Buytaert, J., ... others (2015). Velopix: the pixel asic for the lhcb upgrade. *Journal of Instrumentation*, *10*(01), C01057.

Roeder, J. (2018). *Track reconstruction with cellular automaton: a performance engineering case-study in hep* (Unpublished master's thesis). University of Amsterdam.

Schiller, M. T. (2011). *Track reconstruction and prompt K_S^0 production at the LHCb experiment* (Unpublished doctoral dissertation). Heidelberg U.

Siklér, F. (2019). Another approach to track reconstruction: Cluster analysis. *Universe*, *5*(5). Retrieved from https://www.mdpi.com/2218-1997/5/5/105 doi: 10.3390/universe5050105

Stimpfl-Abele, G., & Garrido, L. (1991, April). Fast track finding with neural networks. *Computer Physics Communications*, *64*(1), 46–56. Retrieved from https://doi.org/10.1016/0010-4655(91)90048-p doi: 10.1016/0010-4655(91)90048-p

Szumlak, T. (2017, October). Plans and status of the LHCb upgrade. In *Proceedings of the 15th international conference on flavor physics & CP violation — PoS(FPCP2017).* Sissa Medialab. Retrieved from https://doi.org/10.22323/1.304.0039 doi: 10.22323/1.304.0039

Zlokapa, A., Anand, A., Vlimant, J.-R., Duarte, J. M., Job, J., Lidar, D., & Spiropulu, M. (2019). *Charged particle tracking with quantum annealing-inspired optimization.*

Appendices

A Data Analysis

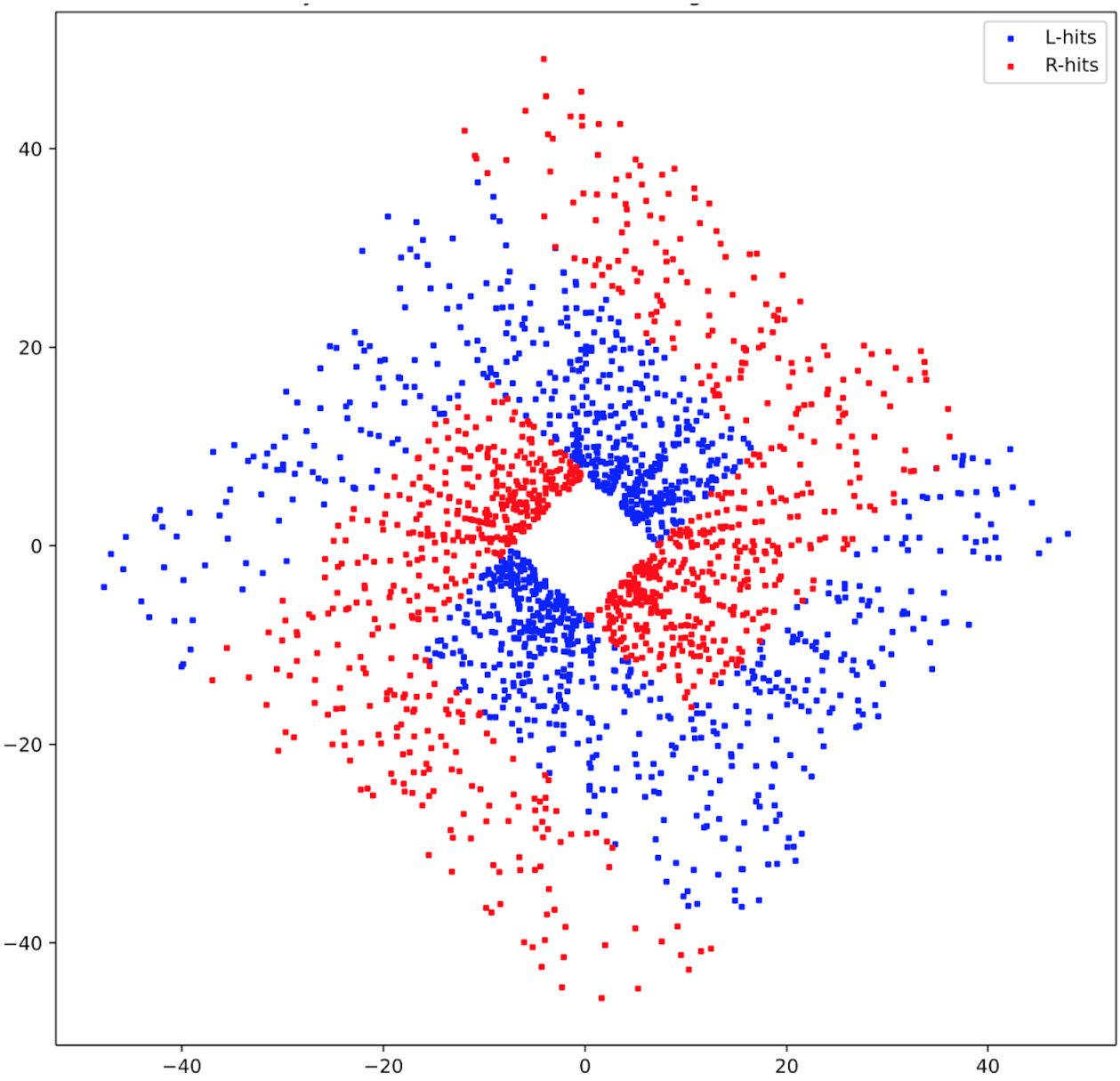
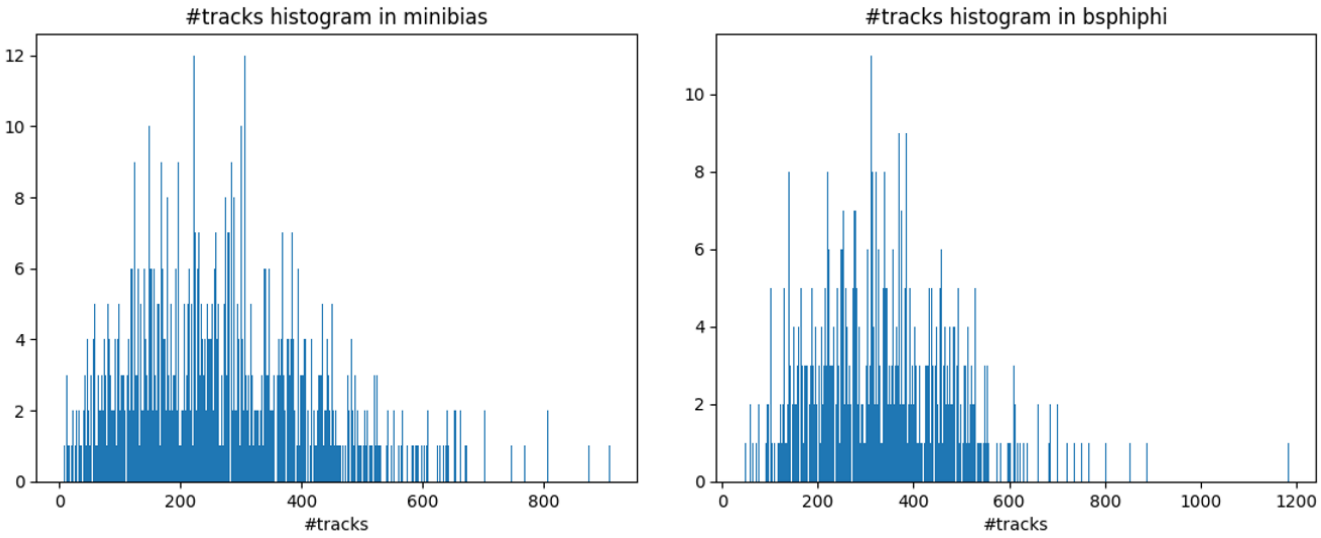


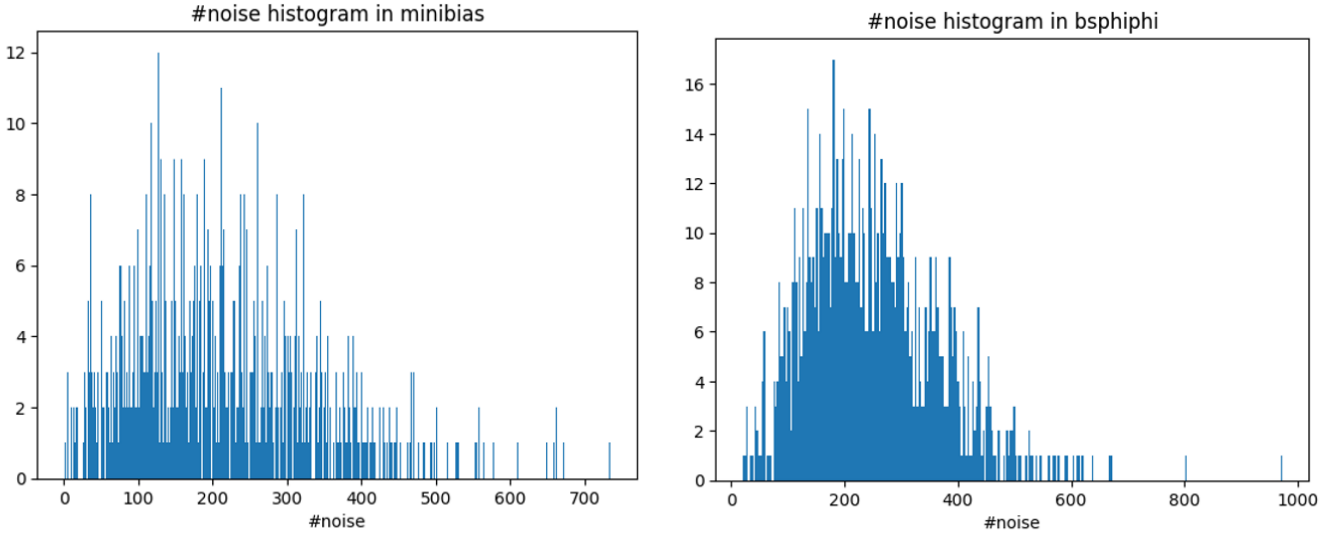
Figure 7: Scatter-plot of the hits of one event in the X,Y - plane. Visualization of the two Z-coordinates in all modules. Hits that were detected in the lower Z-coordinate of one module is labeled as L-hit and coloured in red while hits that are detected in the part of a module with a higher Z-coordinate are labeled as R-hits and coloured in blue



(a) Minimum bias

(b) BsPhiPhi

Figure 8: Histograms with number of tracks in Minimum bias and BsPhiPhi datasets



(a) Minimum bias

(b) BsPhiPhi

Figure 9: Histograms with number of noise hits in Minimum bias and BsPhiPhi datasets

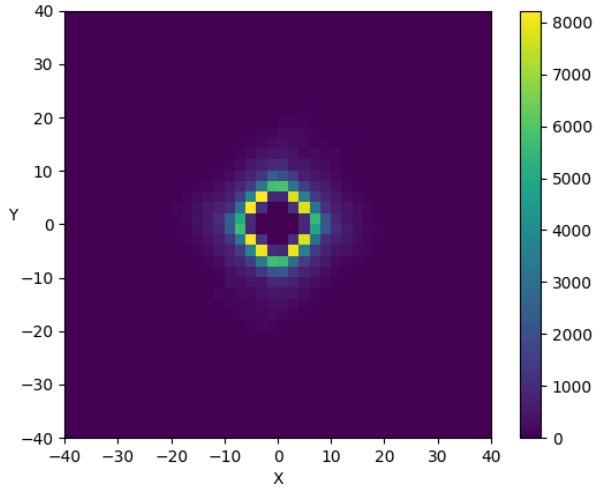


Figure 10: Heatmap of closest hit of tracks to the beam-line in the XY plane

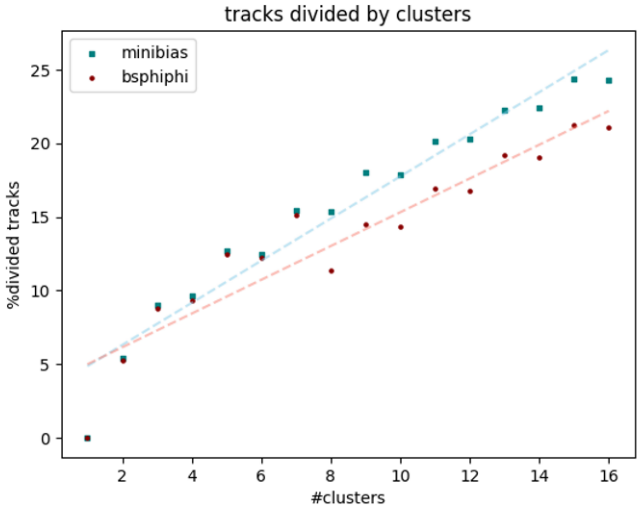
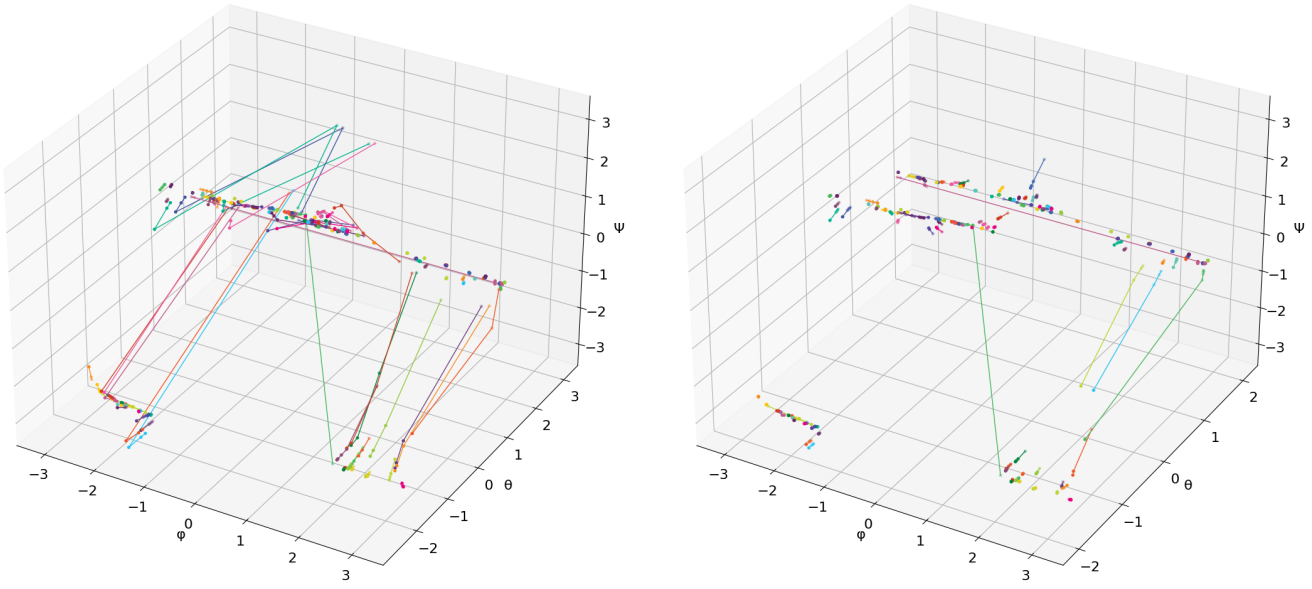


Figure 11: Angle space in plane XY divided in 2 to 16 clusters. The y-axis shows the percentage of tracks with hits in different clusters



(a) Origin of Z-axis at 0

(b) Shifted origin of Z-axis at 40

Figure 12: Scatter plot of hits of one event in their polar angles space. Hits of a same track are connected and share the same color. (Left) Angles θ and ψ computed with respect to origin of Z-axis. (Right) Angles θ and ψ computed with shifting the origin of Z-axis aligning to a single point hits of a track.

B Hopfield Network

	Latest version (23-06-2021)			
Particle category	Reco. eff	Clone fraction	Hit purity	Hit eff.
VELO	74.1	8.44	98.59	83.64
long	91.4	16.17	98.81	79.43
long $>$ 5GEv	93.8	17.04	98.80	79.15
long strange	82.7	9.82	98.42	88.68
long strange$>$5GEv	85.1	6.25	98.39	89.52
Overall fake fractions	5.4			

Table 13: Validation output of the Hopfield network for 100 events over the minibias data set with pruning threshold 0.05.

	Latest version (23-06-2021)			
Particle category	Reco. eff	Clone fraction	Hit purity	Hit eff.
VELO	74.6	8.76	98.53	82.73
long	91.0	15.64	98.68	78.08
long $>$ 5GEv	93.7	16.50	98.49	77.68
long strange	80.0	14.10	98.62	80.22
long strange$>$5GEv	82.1	10.26	97.96	84.03
long fromB	92.6	12.73	98.17	81.99
long fromB$>$5GEv	95.0	14.29	98.39	81.78
Overall fake fractions	5.5			

Table 14: Validation output of the Hopfield network for 100 events over the BsPhiPhi data set with pruning threshold 0.05.

For all results of experiments regarding parameter testing, please refer to the files found on the GitHub repository: `https://github.com/ctelloordonez/velopix_tracking-master/tree/main/algorithms/experim`

C Template matching

C.1 Number of templates

Templates	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
700	61.1	14.6	93.6	70.2	24.3
800	60.7	15.8	93.8	68.4	24.0
900	60.7	17.0	94.0	66.8	23.0
1000	60.6	17.8	94.2	65.5	22.1

Table 15: VELO performance indicators obtained by template matching method 1 on the BsPhiPhi data set using 800 vs 900 vs 1000 templates

Templates	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
700	63.8	17.1	97.4	67.5	11.5
800	63.6	17.7	97.5	66.	11.1
900	62.6	19.3	97.8	64.2	10.2
1000	69.3	67.8	94.9	69.1	20.8

Table 16: VELO performance indicators obtained by template matching method 3 on the BsPhiPhi data set using 800 vs 900 vs 1000 templates

Templates	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
700	65.4	17.6	97.0	67.7	15.0
800	65.0	18.7	97.3	66.1	13.8
900	64.2	19.7	97.5	64.5	13.1
1000	63.3	20.3	97.6	63.2	12.4

Table 17: VELO performance indicators obtained by template matching method 4 on the BsPhiPhi data set using 800 vs 900 vs 1000 templates

Templates	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
700	69.7	40.7	96.2	70.5	16.7
800	70.2	43.9	96.5	69;5	15.4
900	70.1	46.8	96.8	68.3	14.4
1000	70.0	49.0	96.9	67.4	13.6
2000	62.4	62.2	98.0	59.9	10.0

Table 18: VELO performance indicators obtained by template matching method 5 on the BsPhiPhi data set using 800 vs 900 vs 1000 templates

C.2 Comparison different methods BsPhiPhi

Template Matching Method	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
TM1	60.7	17.0	94.0	66.8	23.0
TM2	62.7	19.3	97.4	64.2	12.5
TM3	62.6	19.3	97.8	64.2	10.2
TM4	64.2	19.7	97.5	64.5	13.1
TM5	70.1	46.8	96.8	68.3	14.4

Table 19: VELO performance indicators obtained by template matching methods on the BsPhiPhi data set using 900 templates

D Direction Search

	Latest version (20-06-2021)				
Test	Reco. eff	Clone fraction	Hit purity	Hit eff.	Ghost rate
balance	76.0	4.33	99.67	81.22	1.8
low ghost	70.9	3.19	99.78	75.90	1.4
low ghost	69.9	2.66	99.82	75.56	1.2
$<= 2\%$	77.2	4.51	99.63	82.39	2.0
$<= 2\%$	80.3	26.82	99.52	62.69	2.0*
max reco	82.8	15.76	98.9	79.	4,3*

Table 20: VELO performance indicators obtained by Direction Search version 1 on BsPhiPhi data set. * = without clone reduction, this rate is artificially lower.

	Latest version (14-06-2021)				
Test	low eff	high fraction	low effective	high effective	Ghost rate
XtoY	0.02	0.25	0.03	0.14	+
YtoZ	0.02	0.25	0.03	0.14	+
XtoZ	0.02	0.25	0.03	0.14	+
Clone	0.0007	0.004	0.0011	0.0015	+
ModDif	2	18	4	6	+
LookAround	4	50	6	10	
GhostReduc	0	1	0	1	-
CloneReduc	0	1	0	1	+
CloneRedTimes	0	3	2	3	-
CloneRedChek	0	5	3	5	

Table 21: Parameter effects on metrics if an effect was observed.