Track Finding with the Silicon Strip Detector of the Belle II Experiment

Jonas Wagner

Masterthesis

Institut für Experimentelle Teilchenphysik (ETP)

Advisor: PD Dr. Andreas Meyer
Coadvisor: Prof. Dr. Günter Quast

25.09.2017
Contents

1 Introduction .................................................. 1

2 VXD track finding in the context of the Belle II experiment .......... 3
  2.1 The Belle II experiment ........................................... 3
  2.2 The Belle II Analysis Software Framework – BASF2 ................. 8
  2.3 Track finding at Belle II ............................................ 8
  2.4 Figures of merit in track finding .................................... 10

3 SVD standalone track finding with the VXDTF2 ......................... 15
  3.1 Concept of the VXDTF2 .............................................. 15
  3.2 Track candidate creation ............................................ 17
  3.3 Track candidate quality estimation .................................. 31
  3.4 Track candidate selection ............................................ 39

4 Improved quality estimation based on a multivariate analysis method 51
  4.1 Input features ...................................................... 51
  4.2 Training goal ...................................................... 54
  4.3 Training .......................................................... 54
  4.4 Evaluation ......................................................... 55
  4.5 Feasibility discussion .............................................. 60

5 Overview and conclusion .............................................. 63
  5.1 Performance overview of the VXDTF2 ................................ 63
  5.2 Outlook .......................................................... 67
  5.3 Conclusion ......................................................... 69
1. Introduction

Current collider experiments examine the borders of the standard model of particle physics in two complementary ways, either they accelerate particles to maximum possible energies, allowing to detect heavier particles in the collision, or they accumulate increasing amounts of data at well known energies, allowing to measure tiny discrepancies. Following this logic, the first kind of experiments are called energy frontier, while the latter ones are called precision frontier; the Belle II experiment [1, 2] falls into the latter category.

The core of the data analysis of collider experiments is the identification of the decay tree describing the observed event and the determination of its frequency relative to other decay trees. Therefore, it is crucial to know which particles were present in the event and whether they were produced in the initial collision or if they are the results of subsequent decays of particles.

In order to be able to identify these decays, the trajectory of the particles has to be reconstructed as precise as possible, but this is technically hard to achieve for neutral particles. It is all the more important to obtain this knowledge for all charged particles. In the case of the Belle II experiment, typical collisions are fairly clean with on average only eleven charged particles per event. However, most of those have a momentum below 350 MeV giving the innermost track finding detector — the VerteX Detector (VXD) — a significant role in the reconstruction of them, as they do not reach other detectors.

This thesis is focused on the task of reconstructing the trajectories of the charged particles solely based on the VXD. This detector system suffers from a large background hit contribution mostly caused by low momentum electron-positron pair interactions. The challenge is to find the tracks corresponding to the individual trajectories of the signal particles within the large number of hits. This has to be achieved in the context of a complex detector layout and execution time limitations.

The algorithm that was developed to solve this task, the VXDTF, is described in [3]. However, due to deficiencies in maintainability and in special cases a too high execution time, it was chosen to replace this algorithm with an updated version, which was rewritten from scratch. This thesis is part of the development of this rework, the VXDTF2, which was started in [3].

The aspects of the Belle II experiment, that are relevant in this context, are explained in Section 2, followed by an explanation of the design of the algorithm and the modifications compared to its predecessor in Section 3. A possible use of multivariate analysis methods to
further improve the algorithm is discussed in Section 4. Subsequently, the performance of
the current state of the algorithm is summarised in Section 5, the conclusion of this thesis.
2. VXD track finding in the context of the Belle II experiment

This thesis is focused on an algorithm that is developed for a very specific purpose: track finding in one of the subdetectors of the Belle II experiment. In this section the environment is explained, from a short description of the experiment itself over its hardware, the general purpose of such an algorithm and the software framework in which it is developed, up to a description of the figures of merit used to evaluate it.

2.1 The Belle II experiment

The Belle II experiment is an ongoing major upgrade of the Belle [4] experiment, which was located in Tsukuba, Japan and in operation from June 1st 1999 till June 30th 2010. The upgrade affects both the accelerator as well as the detector and is already in a well-advanced state, with an expected start of the data taking period at the end of 2018 [5]. The information presented in this section is a summary of the aspects relevant for this thesis; it is based on the extensive information available in the technical design report [2].

2.1.1 The SuperKEKB accelerator

The accelerator of the Belle II experiment, the SuperKEKB, is an asymmetric \( e^+ e^- \) accelerator, mainly operating on the \( \Upsilon(4S) \) resonance at 10.58 GeV. It is called asymmetric, because the positrons and electrons are accelerated to different energies and therefore the particles produced in a collision are generally boosted into the direction of the high energy beam. The \( \Upsilon(4S) \) meson produced on the resonance is very short-lived and decays in over 96\% of the cases into a B meson pair. Hence, the accelerators of the Belle and the Belle II experiment are both called B-factories.

The upgraded accelerator is designed to produce a peak luminosity of \( 80 \times 10^{34} \text{ cm}^{-2}\text{s}^{-1} \), which is about 40 times higher than the previous one. To cope with the modified beam setup, also the detector has to be upgraded.

2.1.2 The detector system

The detector of the Belle II experiment is divided into several onion layers featuring different detector types and purposes. An illustration of the detector is given in Figure 2.1.
Due to the asymmetric design of the accelerator, the $\Upsilon(4S)$ meson is boosted along the high energy beam direction; this direction is called the forward direction. In order to optimise the detectors efficiency it is also designed differently in forward and backward direction and can be separated into endcap and barrel parts. The coordinate system, used in the Belle II collaboration, is aligned to the cylindrical shape of the detector. The $z$-axis is pointed in forward direction, while the $x$-axis is oriented in parallel to the ground and out of the accelerator ring, the $y$-axis orthogonal to the ground and towards the sky. Instead of using these right handed Cartesian coordinates directly, they are often transformed to spherical ones, where $\phi$ is the angle inside the $xy$-plane and $\theta$ the angle towards the $z$-axis.

The outermost subdetector, the KLM, is used for $K_L^0$ and $\mu$ detection, based on the low likelihood that other particles activate this subdetector. Further inside the detector an Electromagnetic Calorimeter (ECL) is used to identify electrons and photons and the energy of them. Two more particle identification systems are located further inside than the ECL: In the barrel part, a Time-Of-Propagation (TOP) counter is used to separate...
kaons from pions and in the endcap an Aerogel Ring-Imaging Cherenkov (ARICH) detector is installed. The latter detector cannot only separate kaons from pions, but also improves the discrimination between low energy pions, muons and electrons. Finally, the inner part of the detector is dedicated to particle tracking; the process of reconstructing the trajectory of charged particles. It consists of two subdetector systems, the Central Drift Chamber (CDC) and a silicon based subdetector, the Vertex Detector (VXD). Because this thesis is focused on the VXD, this subdetector is further described in the following paragraph.

2.1.3 The VXD detector

The VXD is divided into four layers of strip sensors, the Silicon Vertex Detector (SVD), and further inside, two layers of pixel sensors, the PiXel Detector (PXD).

In order to reach a high coverage, both of these detector systems feature a complex layout:

Detector layout

All sensors are attached to ladders that are aligned in parallel to the beam axis. In order to obtain a constant $\theta$-coverage of $[17-150]^\circ$ throughout the layers, the size of the ladders increases with increasing distance to the beam. The higher coverage in forward direction is chosen to cope better with the asymmetric collisions. Due to the increase in length, also the number of sensors per ladder increases towards the outer layers.

As it is shown in the lower part of Figure 2.2, for the PXD each ladder holds the same amount of sensors, two. But for the SVD, shown in the upper part, starting from two sensors per ladder, the amount increases by one for every layer. Beside the practical reason to maintain a similar sensor size, this has the additional benefit that the non sensitive material between the sensors is in a different $\theta$-region for every layer. Therefore, over the whole $\theta$-coverage, it becomes very unlikely that a particle traverses the detector without hitting a sensitive area.

An important feature of the SVD layout are the slanted sensors in forward direction of the detector. This layout is chosen to obtain a larger $\theta$ coverage with the same amount of sensors, in order to reduce the material and cost of the detector. However, due to this angle, the slanted sensors have to be shaped trapezoidal.

The shape of these slanted sensors is clearly visible in the transverse section of the detector in Figure 2.3. In this figure the ‘windmill’ design of the ladders positioning is clearly visible: neighboured ladders on the same layer overlap each other, in order to prevent an insensitive area between them. Also visible in this figure is the increasing amount of ladders per layer, within each of the two subdetectors. As stated before, in this way the sensor width can be constant and they are aligned better.

While this layout improves the efficiency of the detector, it becomes more complex to deal with on software side. Both along the beam axis and in transverse direction no global symmetry is present: The slanted sensors and the asymmetric $\theta$-coverage break the symmetry along the beam axis and the different amount of ladders breaks the rotational symmetry around the beam axis.
Figure 2.2: Longitudinal section of the VXD geometry
This drawing reflects the state of the geometry implemented in the Belle II software in August 2017. Only the active area of the sensors is drawn, the numbers encode layer, and sensor number. The star between the numbers indicates that the layout is identical for all ladders.

Figure 2.3: Transverse section of the VXD geometry
This drawing reflects the state of the geometry implemented in the Belle II software in August 2017. Only the active area of the sensors is drawn. In the left image the SVD is drawn and in the right image the PXD including the first layer of the SVD as a reference.
Silicon Sensors

The main difference between the PXD and the SVD is the different sensor technology.

The SVD uses double sided silicon strip detectors. These detectors can be read out on each side, but on one side rotated by 90° compared to the other side. Therefore, a three dimensional position information can only be obtained by combining signals from both sides of a sensor. If there are more signals than a single one on each side, it is undefined which hits correspond to each other. In such a situation all possible combinations of them have to be considered, effectively producing additional ‘ghost’ hits. In order to keep the number of those ghost hits low, the occupancy of the sensors, defined as the fraction of activated channels per event, has to be as low as possible.

This number is not dominated by hits from particles originating from a Υ(4S) event, but from so called background processes. In case of the VXD these are mainly low-momentum-transfer QED processes at the interaction point and interactions of the beam with its surroundings — the so called beam background. This background is proportional to the proximity to the beam and therefore the double sided strip sensors are not feasible anymore for the two innermost layers.

Therefore, the PXD uses a different, more expensive sensor technology than the SVD. As the name suggests, each sensor module of the PXD is divided into a grid of pixels. Due to the higher amount of channels a lower occupancy can be achieved, compared to a double sided silicon strip exposed to the same background rate.

The different sensors not only differ in their layout, but also in their integration time. With a readout time of 20 µs, the PXD sensors are quite slow. This increases its occupancy, because the readout time window is a lot bigger then the time window of an actual Υ(4S) event. The readout time of the SVD however, is roughly 100 times shorter and it can be improved even more by an analysis of the shape of the signal. It is estimated, that this can give a time resolution within the whole readout time of about [3-10] ns. However, the implementation of this within the software framework is still ongoing.

Typically a charged particle passing through the VXD activates several channels on a single sensor and before they are used in track finding algorithms these are combined to clusters. Therefore, a track finding algorithm has access to clusters and information about the shape of those. The average number of clusters per event of the different detector systems, obtained with the current state of the Belle II software and simulation, are stated at the beginning of Section 3.

2.1.4 Data acquisition

The luminosity of current collider experiments is so large that the data, produced by the detector systems, cannot be stored completely. This is typically solved by so called trigger systems, which try to detect the events of interest, in order to only store those. These systems perform a rough reconstruction of the observed data and based on that, it is decided whether that part of the data might be interesting. If it is, the storing of the data is triggered. Depending on the data rate of the detector and the data rate that can be stored in the end, these systems can consist of several stages, which often even implement a rudimentary reconstruction in hardware.
The Belle II experiment uses such a hardware trigger as the first level trigger and a software trigger, the High Level Trigger (HLT), as the second level. Some of the track finding algorithms of the Belle II experiment are already used on the HLT to classify the event; this is also true for the algorithm discussed in this thesis.

In order to allow a classification of all incoming events the HLT poses execution time limits to the individual algorithms. Although these are only rough limits at the moment — an execution time in the order of magnitude of 10 ms per event in case of the algorithm covered in this thesis — they are already tight enough that they have to be considered in the design of the algorithm.

2.2 The Belle II Analysis Software Framework – BASF2

BASF2 is the successor of the analysis framework used for the Belle experiment – BASF. Because the new experiment would have required extensive modifications within BASF, it was instead decided to develop BASF2 from scratch.

The core of the framework is a pipeline to chain different modules together and a datastore used for communication between these modules. Although the framework is written in C++, it provides a powerful python interface to configure the pipeline and the modules.

The modules are building blocks for I/O, simulation, reconstruction or any other task that should be performed within the framework. Typically they are written in C++ to be execution time efficient, but the python interface also allows to write modules to some extend.

In the pipeline each registered module is first initialised according to their order in the pipeline and subsequently each module after another processes a single event. The framework itself manages the parallel processing of multiple events. After all events have been processed, the modules are terminated in their reverse order.

In addition to this core functionality, the framework has several other important features, such as error handling and logging; more information about the framework than is contained in this short summary can be found in [7] and [8].

2.3 Track finding at Belle II

As stated before, the process of reconstructing the trajectory of charged particles based on their energy deposition in a detector system is called track finding. In combination with a magnetic field it is possible to not only reconstruct the trajectory, but also infer the momentum and charge sign of a charged particle from the curvature of the trajectory.

This information is crucial for every analysis that will be performed at the Belle II experiment. It is not only required to identify the decay tree of an event, but also for the precise measurement of the corresponding decay vertices. In order to obtain an optimal understanding, there are usually dedicated detector subsystems for this purpose. In the case of the Belle II experiment, these are the CDC and the two silicon based system, the PXD and the SVD, as described in Section 2.1.
2.3 Track finding at Belle II

Due to the different sensors and layout of these detectors, they feature different advantages. The CDC is furthest away from the interaction point and therefore will be considered high transverse momentum track finding in the context of this thesis. One advantage of this system is that it introduces very few material and that it is relatively cheap to cover a large area and . Due to the large size, even small curvatures of particles with a high transverse momentum can be measured with a reasonable precision.

The silicon subsystems are closer to the collision point but only consist of 6 layers in total. Due to the small dimension of these subsystems the momentum resolution for particles with a high transverse momentum is significantly worse than in the CDC. However, these subsystems feature a high position resolution, which is important to determine decay vertices for very short-lived particles that decay before they reach even the first layer of the detector systems. Some analyses, such as studies based on the time difference between the decays of the two B mesons in a $\Upsilon(4S) \rightarrow B \bar{B}$ event, are only possible due to this precise resolution. Therefore, even for particles with a high transverse momentum, the overall quality of a track increases if hits in the VXD are found additionally to hits in the CDC. Additionally, tracks with a low transverse momentum can only be found within the VXD at all.

**Incorporation of the VXD in track finding**

For the Belle experiment, track finding was performed in the CDC and the hits in the VXD were added by extrapolating the reconstructed trajectories. This approach has the advantage, that there is already a starting seed for the addition of VXD hits, which drastically reduces combinatorial issues. But this approach has the disadvantage that low momentum particles that do not produce enough hits in the CDC cannot be reconstructed.

This drawback can be eliminated by performing standalone track finding in the VXD which is made possible by the increased amount of layers of this subdetector for the Belle II experiment. Furthermore, subdetector specific topics, such as the geometry and dominating background signatures, can be incorporated more naturally if they are treated with a dedicated algorithm. On the downside, however, the tracks reconstructed by the individual algorithms have to be merged afterwards.

In the end, both approaches have different advantages and therefore, algorithms for both of them are being developed and evaluated; The VXD standalone algorithm is the topic of this thesis. Since these algorithms do not necessarily exclude each other, the final track finding for the Belle II experiment will probably feature a combination of them, such as an extrapolation of the CDC tracks and VXD track finding on the remaining hit-set.
2.4 Figures of merit in track finding

Most studies, in particle physics, are at first developed and evaluated based on detailed simulations; the same holds true for track finding algorithms. These simulations require large amounts of random numbers to describe the probabilities imposed by the theory of particle physics. Therefore, they are commonly referred to as Monte Carlo, after the casino with the very same name in Monaco. The major advantage over real data is the rather cheap production costs of massive datasets and the information what truly happened in an event. Hence, the results of the algorithms can be compared to the precise knowledge of the simulation: the so called Monte Carlo truth.

**Trackable particles**

In order to be able to quantify the performance of a track finding algorithm, one defines which trajectories should be found by the algorithm in principle. In the case of the VXD standalone track finding, the requirements to consider a trajectory of a particle as trackable were chosen such that:

1. it should produce enough hits in the detector to have at least five degrees of freedom,
2. the particle has to be a primary particle.

A particle is defined to be primary if it is produced during the event simulation, in contrast to being produced during the detector simulation by interaction with the material or when another particle decays inside the detector.

The first requirement is used for both CDC and VXD track finding, because five degrees of freedom are required to define a helix, the general trajectory of a charged particle in these detectors. The restriction to only consider primary particles is firstly, because these particles define the physics interpretation of the event and secondly, because it is difficult to find tracks of secondaries in the volume of the VXD. The VXD only has six layers to begin with and only some of them can be effective for a particle that is created inside of the detector. In addition, these particles have a different signature. While an algorithm that searches for tracks originating from the interaction point works perfectly fine for primaries, it will perform worse for secondaries. Therefore, it was decided to focus on primaries first.

A track finding algorithm utilising the Monte Carlo information of the simulation is used to reconstruct the trajectories of all particles that fulfil these requirements. In the rest of this thesis tracks produced by this track finder are called Monte Carlo tracks and their corresponding particle is called Monte Carlo particle. On the contrary, a track finding algorithm that is capable of processing real data will be called pattern recognition algorithm in this thesis. Such an algorithm produces track candidates, or simply candidates, as intermediate objects and tracks as the result.

**Figures of merit**

There are several important quantities to determine the performance of a track finding algorithm. These quantities, described below, quantify how well the pattern recognition algorithm reproduces the Monte Carlo tracks. In order to allow the calculation of these quantities, the pattern recognition tracks have to be matched to the Monte Carlo tracks.
A pattern recognition track is matched to a Monte Carlo track when the following two requirements are fulfilled.

1. At least 5% of the hits of the Monte Carlo track are also present in the pattern recognition track.

2. At least 66% of the hits of the pattern recognition track are also present in the Monte Carlo track.

These requirements were agreed upon both for CDC and VXD track finding, even though the criteria are not equally important in the different scenarios. In the case of SVD standalone track finding the first requirement is fulfilled easily due to the low number of layers and thus hits per track. Usually a single hit of a Monte Carlo track that is also present in the pattern recognition track is enough to satisfy this requirement. A Monte Carlo track would need to have more than twenty hits in order to require more than a single hit to be found by the pattern recognition track; the average is eight hits, two per layer.

The second criterion, on the other hand, poses tighter restrictions in the case of SVD standalone track finding. With on average eight hits per track candidate only two of them are allowed to be wrong: Either corresponding to a different Monte Carlo track than the remaining ones or caused by background.

If several tracks are matched to the same Monte Carlo track, the track with the highest amount of common hits and the lowest amount of other hits is chosen to be matched, while the others are called clones. And if a track does not fulfil all of the matching criteria it is called fake.

Based on these matching criteria the following figures of merit can be calculated.

**Finding efficiency** The number of Monte Carlo tracks that are matched with pattern recognition tracks; normalised to the total number of Monte Carlo tracks.

**Fake rate** The number of pattern recognition tracks that are not matched with Monte Carlo tracks; normalised to the total number of pattern recognition tracks.

**Clone rate** The number of pattern recognition tracks tagged as clone; normalised to the number of pattern recognition tracks that are tagged either as clone or as matched.

**Hit efficiency** The fraction of hits of a Monte Carlo track that are also present in the matched pattern recognition track. This fraction is averaged over all Monte Carlo tracks that have been matched. Hence, the minimum value of this quantity is defined by the first matching criterion: 5%.

**Hit purity** The fraction of hits of a pattern recognition track that are also present in the matched Monte Carlo track. This fraction is averaged over all pattern recognition tracks that are either matched or a clone. Hence, the minimum value of this quantity is defined by the second matching criterion: 66%.
Therefore, the ideal track finding algorithm would have a finding efficiency of 100\%, as well as a fake and clone rate of 0\%. Additionally the tracks that are found should feature 100\% for both hit efficiency and hit purity.

However, in the context of a real experiment there are a lot of factors influencing these figures of merit. Not only is it likely to mix two tracks that are close to each other, there are also background hits that could be added to tracks. These background hits can be produced by interactions of the particle beam with the detector or by noise of the readout electronic of the sensors.

**Statistical uncertainties**

The figures of merit, discussed above, follow different distributions and therefore, different statistical uncertainty models are applied.

The finding efficiency follows the Binomial distribution. There are \( n \) Monte Carlo tracks present in the sample and \( k \) of them have been reconstructed successfully. In this thesis the uncertainty on this ratio \( \epsilon \) is calculated with the normal approximation standard deviation [9]:

\[
\sigma_\epsilon = \sqrt{\frac{\epsilon (1-\epsilon)}{n}}.
\]  

(2.1)

This approximation features deficits for the borderline cases, an efficiency of almost zero or one and low statistic. But none of these cases are relevant in the context of the numbers presented in this thesis.

A similar argument can be made for the fake and the clone rate: \( k \) track candidates of all \( n \) track candidates fall into the respective category. Therefore, these quantities are statistically treated the same as the finding efficiency.

The hit efficiency and hit purity are harder to treat properly. Even for each track alone, the underlying distribution of this ratio is unknown and building the average over all tracks does not improve the situation. Therefore, we treat these quantities with bootstrapping [10]:

The quantity under investigation is based on a set of measurements \( m_{\text{obs}} \). Several new sets of measurements \( m_i \) are created by randomly taking values from \( m_{\text{obs}} \), explicitly allowing to take the same element multiple times. Subsequently the quantity is calculated for each \( m_i \) producing a distribution for the quantity under research. Based on this quantity either a confidence interval can be calculated or, if the distribution resembles a Gaussian, the respective standard deviation can be used instead.

In the case of the hit efficiency, the set of measurements is represented by the hit efficiencies of all matched track candidates and, based on that, the mean hit efficiency is calculated. The same is true for the hit purity, but based on the individual hit purities of the track candidates.

Based on one of the sets of measurements discussed later in this thesis, column \( \text{MC Greedy} \) of Table 3.6, the distributions obtained for these quantities are shown in Figure 2.4. The original set has been resampled 5000 times. Mean, median and observation are in good agreement and the intervals cover the same area as the standard deviation.
2.4 Figures of merit in track finding

Based on the sample referenced in this section, the hit efficiency and hit purity have been bootstrapped in order to determine the uncertainty on the observed values.

Table 2.1: Uncertainty model comparison
The uncertainties of the observed figures of merit have been calculated both with the corresponding uncertainty model, column Model, and the bootstrapping method, columns beginning with a B.

<table>
<thead>
<tr>
<th></th>
<th>Observation</th>
<th>Model</th>
<th>B-upper</th>
<th>B-lower</th>
<th>B-std</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finding efficiency</td>
<td>94.484</td>
<td>0.073</td>
<td>0.073</td>
<td>0.073</td>
<td>0.072</td>
</tr>
<tr>
<td>Fake rate</td>
<td>19.810</td>
<td>0.116</td>
<td>0.116</td>
<td>0.117</td>
<td>0.117</td>
</tr>
<tr>
<td>Clone rate</td>
<td>0.413</td>
<td>0.021</td>
<td>0.021</td>
<td>0.022</td>
<td>0.021</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>97.207</td>
<td>-</td>
<td>0.033</td>
<td>0.034</td>
<td>0.034</td>
</tr>
<tr>
<td>Hit purity</td>
<td>99.756</td>
<td>-</td>
<td>0.008</td>
<td>0.008</td>
<td>0.008</td>
</tr>
</tbody>
</table>

The bootstrapping method can also be used to verify the uncertainty models of the other approaches. This is compared in Table 2.1. The bootstrapping method produces results compatible with the normal approximation interval. Since there is no significant underestimation, the simple model should be sufficient in the context of this thesis.
3. SVD standalone track finding with the VXDTF2

The Vertex Detector Track Finder, VXDTF [3], is designed to perform standalone VXD track finding for the Belle II experiment. It is estimated that the PXD alone will detect roughly 10,000 clusters per event and only a small fraction of these clusters, on average 25, are produced by the particles originating from the collision. To reduce the amount of data that has to be stored for later analyses, only clusters in regions of interest are considered. These regions of interest are determined on the software high-level trigger by extrapolating from the SVD to the PXD. Therefore, the VXDTF also has to be capable of SVD standalone track finding to be able to reconstruct tracks that can be extrapolated. The complete VXD track finding is performed only on the already stored data.

The VXDTF2 is a modularised and enhanced version of the VXDTF. Although the PXD is incorporated into its development, as explained below, this thesis focuses on the SVD standalone track finding part.

3.1 Concept of the VXDTF2

The basic concept of standalone track finding in the VXD subdetector has to meet several challenges:

As stated in Section 2.1.3, the VXD subdetector is separated into two layers of pixel sensors, the PXD, and four layers of strip sensors, the SVD. While three dimensional coordinates can be obtained directly from the pixel sensors, this is more complex to achieve with the strip sensors, because two clusters from both sides of a sensor have to be combined. However, it is unknown which combination of clusters represents the true hit if there are more than two clusters per sensor. The additional combination possibilities effectively produce additional hits, called ghost hits.

Another difficulty is the complex geometry of the layers of the VXD, as it is described in Section 2.1.3. Due to the windmill design, small parts of the sensors overlap. Thus, particles can activate two sensors in a single layer when they pass through the detector. Additionally, within each subdetector the number of ladders increases with each layer in such a way that there is no rotational symmetry. The symmetry of the VXD is further broken by the slanted sensors of the SVD, introduced to improve the performance in the context of the asymmetric collider. Therefore the detector has no global symmetry in either $\phi$, $\theta$ or $z$ direction to utilise.
Finally various circumstances pose execution time and memory limits to the algorithm. Because the SVD standalone track finding will be running on the software high level trigger system of Belle II, the target execution time is in the order of magnitude of 10 ms on a single process of the target machine. This constraint is given for a single core, because particle physics are embarrassingly parallel: Every event is independent of each other. Therefore it is sufficient to process every event in parallel instead of incorporating more complex methods such as multi threading. Memory limitations then come naturally from the available memory for each process on the high level trigger system. These limitations influence how the algorithm can cope with the huge amount of data produced by the detector system:

The VXD detector suffers from a high occupancy due to its special proximity to the beam. With current background estimations the SVD detects roughly 350 clusters per event. Although the PXD has only two layers, there are more readout channels due to the division into pixels. Due to that, the PXD adds another 1000 clusters per event, even if the above mentioned data reduction is utilised. These clusters have to be assigned either to tracks or background. With on average eleven tracks in an event and an average of eight clusters per track in the SVD, this means roughly $1 \times 10^{218}$ possible combinations for SVD track finding; even more if the PXD is included. This makes a naive approach, such as fitting all combinations, too time consuming.

In order to meet these challenges, the track finding algorithm is based on three key concepts:

1. For the purpose of treating the two subsystems, PXD and SVD, in a unified way, the candidate creation is based on so called SpacePoints. A SpacePoint is based on a single cluster in case of the PXD and two clusters for the SVD. Therefore, it can provide access to three dimensional coordinates independent of the detector system. Additionally it stores information about layer, ladder and sensor ID of the corresponding clusters. Apart from unifying the two subsystems, this also handles the complex geometry of the SVD in a nice way. While it provides the ability to reason about SpacePoints without a detailed knowledge of the detector geometry, it is still possible to access it, which is useful for prefiltering likely connections.

2. Defining so called friendship relations between sectors, based on the geometry of the detector, is a powerful method to drastically decrease the number of combinations to evaluate: Only those combinations of SpacePoints, that correspond to sectors that are befriended, have to be evaluated. As an example each sensor of the detector could represent such a sector. To reduce the number of combinations even further simple filters for 2- or 3-SpacePoint combinations can be defined. Both of these methods are realised with the SectorMap concept, which is described in detail in Section 3.2.1.

3. This prefiltering of combinations reduces the amount of created candidates to a level where more sophisticated methods may be used to determine the quality of a given track. At this point every candidate gets a quality index assigned in order to select those candidates that describe the trajectories of the particles in the event in the best way. The chosen criterion to determine this set of candidates is to require a set of candidates which do not share clusters. The reasoning behind this is explained in detail in Section 3.4, but it allows to rule out clones and a lot of the fake candidates as well.
3.2 Track candidate creation

These concepts for standalone track finding in the VXD were tested with the first version of the VXDTF and proven to be successful [3]. However, it was decided that the VXDTF has to be rewritten due to some deficiencies in maintainability and also execution time in special cases. The reimplementation, follows the same concept but is modularised and features enhanced algorithms. The modularisation increases the maintainability and provides an easy way to detect bottlenecks. The culprit can be identified and optimised without affecting other modules of the algorithm. The communication between the modules is realised via the Datastore provided by the framework (Section 2.2). In case of the VXDTF2, the SectorMap is stored for the whole execution time, while the other artefacts, like track candidates, are only stored on an event by event basis.

In the resulting structure the concepts are clearly visible: The input is independent of the subdetectors of the VXD, because the clusters are immediately converted to SpacePoints for the further processing. As shown in Figure 3.1 the first step when applying the VXDTF2 to data is creating a set of reasonable candidates utilising the SectorMap. Subsequently, these candidates are ranked and reduced to a non-overlapping set of candidates. The final candidates are then converted to a track representation that is used to combine information of the VXD and the CDC. In the following sections the modules of the VXDTF2 and the contributions of this thesis to them are explained in detail.

3.2 Track candidate creation

In the first stage of the VXDTF2, several SpacePoints are combined to form a track candidate. This is realised in two steps that can be explained individually: At first, based on a fast prefiltering, a network of all sensible combinations of SpacePoints is constructed. This prefiltering is realised with the SectorMap. In a second step, a cellular automaton collects paths in this network and creates track candidates from them.
3.2.1 The SectorMap

The desired goal of the SectorMap is to accept all combinations of SpacePoints that represent a track produced by a real particle from the collision and as few others as possible. In general this can be achieved by introducing knowledge about the physical processes to the model as demonstrated in the following simplified example:

If material effects are neglected, charged particles in a constant homogeneous magnetic field move on a helix trajectory. This knowledge about expected trajectories can be utilised by defining friendship relations between sectors. If it is likely that a particle from the collision that traverses sector $A$ will also traverse sector $B$, these sectors are related. These sectors could be the individual sensors of a detector.

This is only a qualitative example and defining these relations in the context of the complex geometry of the VXD would be a gigantic effort without the use of automation: Friendships within the same layer as well as friendships skipping a layer have to be defined without the possibility to utilise any symmetry. Therefore, the SectorMap is trained based on simulated collisions and the complete detector simulation. This has the advantage that the geometry can be considered in a very precise fashion and no assumptions have to be made about material effects: the simulation incorporates them to the best knowledge.

Furthermore, because the whole simulation chain is set up and ready to use, it becomes easy to train additional filter criteria tailored to the friendship relations: For each sector friendship the properties of the traversing particles can be analysed and used to tune filters specific to these sectors. This would be impossible without the automation.

There are three types of filters stored in a SectorMap:

**Sector friendships** Sector friendships are the name-defining part of the SectorMap. The use of Monte Carlo tracks makes it possible to use an even finer structure than the individual sensors; each of them is divided into a grid of sectors. Which sector a Monte Carlo track traversed can easily be determined from the local coordinates of its corresponding clusters. Subsequently if at least one track traverses two sectors they become related. These relations not only encode that two sectors are befriended, but also in which order particles traverse them.

**2-SpacePoint filters** 2-SpacePoint filters define requirements for various properties that can be calculated from a pair of SpacePoints, for example the distance between them. These filters are stored with regard to the associated sectors. This makes it possible to consider different behaviour in different parts of the detector.

**3-SpacePoint filters** 3-SpacePoint filters are analogous to 2-SpacePoint filters, but based on a triplet of SpacePoints. Therefore, they provide access to more sophisticated properties, like the radius of a circle through the x-y-projection of the SpacePoints.

Since tracking based on only the SVD incorporates only four detector layers, requiring four SpacePoints would be quite a hard requirement even without any filter variables. On the
other hand, introducing 4-SpacePoint filters only for tracks that are long enough would add an unnecessary amount of complexity to the algorithm. Therefore, filters with $n$ greater than three are not used in the VXDTF2.

### 3.2.2 Network creation

The first step of the candidate creation is the network creation. It is based on the SectorMap and separated into three stages analogous to the three types of filters stored in the SectorMap. In fact there are three directed networks created where each is based on the previous one. The idea is that the stronger but more complex and therefore more time consuming filters are only applied to combinations that already passed the weaker filters.

1. The **SectorNetwork** contains sectors as nodes; a sector is included if it contains at least one SpacePoint in the currently processed event.

   In this network all sectors that fulfil the trained friendship relations stored in the SectorMap are connected. The direction of the connection is defined by these relations according to the order the particles in the training traversed these sectors.

2. The **SpacePointNetwork** contains SpacePoints as nodes.

   It can be constructed by looping over the edges of the first network. All SpacePoints that are in the sector connected to the edge on the one side are combined with the SpacePoints present in the sector connected to the other side of the edge. If a pair of SpacePoints passes the requirements imposed by the 2-SpacePoint filters they are both included in the second network and connected. The direction of this connection is inherited from the edge of the first network.

3. The **SegmentNetwork** contains pairs of SpacePoints as nodes.

   It can be constructed by looping over all paths of the second network. For each path, two successive nodes represent a single node in the third network; these are called segments. Only if two consecutive segments of a path pass the 3-SpacePoint filter requirements, they are included and connected. As before, the order of the nodes is based on the previous network, the SpacePointNetwork in this case.
Therefore, two nodes of the final network that are connected represent a SpacePoint triplet that has a clear ordering and fulfils all requirements of sector relations, 2-SpacePoint filters and 3-SpacePoint filters.

3.2.3 Path finding

The second step of the candidate creation is gathering possible paths from the network. The naive implementation starts at a random node in the network and collects all possible paths through the connected inner nodes. From all paths that pass a minimum length requirement the SpacePoints are extracted and combined to a track candidate.

This can be optimised by taking physical knowledge into account. The layers of the detector that are furthest away from the beam have a lower background rate than the inner ones. Therefore, a SpacePoint in such a layer has a lower probability to be a background SpacePoint. Hence, the outermost layer of the network is a better starting point for the path finding than any inner one. However, it is possible that no SpacePoint was produced in the outermost layer and therefore, instead of the outermost layer, the outer leaves of the network should be used.

Also, the friendship relations will allow skipping a layer completely. This is important in order to cope with defective sensors or inefficiencies. However, this generally allows to produce paths that skip a node even if it is present. One approach to only select the longest path is to first apply a cellular automaton to the network:

Each node of the network corresponds to a cell in the cellular automaton and has the initial state zero. The cellular automaton increases the state of a cell if it has at least one incoming edge that is connected to a cell that has the same state. All cells are updated simultaneously in such an update cycle and the algorithm terminates if there are no more changes of states. For increased clarity, this is visualised in Figure 3.2.

After the cellular automaton is applied to the network, any edge, that connects two cells with states that differ by more than one, can be neglected. Hence, the path finding is only allowed to use the longest path. If multiple paths fulfil this criterion for a given starting cell, all of them are accepted.

With the naive implementation the shown network could produce five track candidates, while there are only three candidates likely to be correct. One of the candidates is discarded because the skip of the node is unnecessary and the other one is discarded because a single matching node in an inner layer of the detector is more likely to be caused by background than two matching nodes.

A special problem of path finding in a directed network are cyclic connections. Without a dedicated stopping criterion the algorithm will be trapped in an endless loop. Therefore, the cellular automaton also stops after a fixed maximum number of cycles. This limit is set to a value that is on the one hand high enough to only trigger if such a circular dependency is present in the network and on the other hand low enough to prevent a serious execution time delay. Because there are typically five nodes per track in the SegmentNetwork described in the previous section, if the full VXD is used, the maximum number of cycles was set to 25.
3.2 Track candidate creation

FIGURE 3.2: Cellular automaton
In this figure a cellular automaton is applied to a directed network. Each circle represents a cell in the context of the automaton and a node in the context of the network. The state of a cell is defined by the numbers, the neighbours of a cell by the edges of the network. Each box shows the state of the cellular automaton after the stated update cycle; the cell state of each cell that has an inner neighbour with the same state is increased simultaneously. In the draws scenario, the cellular automaton will stop after cycle four, because there will be no more changes. Red edges highlight connections that are forbidden for the path finding, because of incompatible cell states.

Additionally, it was decided to implement options that allow not only storing the longest path, but also storing shorter subsets of this path. These options are visualised in Figure 3.3 and explained below:

**Flexible seeding** allows not only starting the path finding from the outer leaves but also any inner node that produces a path that is long enough,

**Store subsets** allows verifying after each addition of a node if the path is long enough already and if so, to store a copy of it up to this point.

The combination of these options stores every continuous set of three or more SpacePoints in the path.

These options can be useful in the presence of background, because such a shortened track candidate might have one background SpacePoint assigned less. All track candidates produced in this way are then passed on to the next stage of the VXDTF2, the quality estimation described in Section 3.3.
3.2.4 Validation of the candidate creation

To verify that the algorithms described in the previous three sections work in the expected way, we performed several tests on them. During this process some additional modules were required to validate the components separately.

Candidate conversion

The most basic of the additional modules are converters between the track candidate format dedicated to VXDT tracking, SpacePointTrackCandidates (SPTCs), and a more general track representation: RecoTracks (RTs). RTs are used for functionality that is useful for various track finding algorithms and for the merging of candidates of different algorithms. Each track finding algorithm may use its own specialised track candidate representation, but at least has to provide a method to convert the specialised format to RTs.

In fact, in the case of the VXDTF2, both of these conversion modules are required in the module chain even without the validation performed in this section. Not only, as stated above, to convert the final track candidates of the VXDTF2 to the generally used RTs, but also in the context of the training of a SectorMap in order to convert Monte Carlo RTs to SPTCs.

These modules can be used to validate each other by converting back-and-forth between the formats and comparing the results. While RTs are capable of dealing with the two dimensional information from the SVD directly via clusters, SPTCs are based on SpacePoints which encode the full three dimensional information. Because it is not supported to convert a single SVD-cluster, which only provides a two dimensional position, to a SpacePoint, the
3.2 Track candidate creation

Table 3.1: Candidate conversion validation
Tracks are converted back-and-forth between two formats, RecoTracks (RTs) and SpacePoint-TrackCandidates (SPTCs), twice in a row. The complete conversion chain can be written as: RTs1 → SPTCs1 → RTs2 → SPTCs2 → RTs3.

The Monte Carlo based track finder is used to obtain the original RecoTracks, RTs1, from 100 events containing 100 muons each.

The different track representations are checked for differences after each back-and-forth conversion. Hit efficiency and Hit purity can be seen analogous to the definition in section 2.4, while the Track efficiency is defined by the number of tracks before and after the conversions, normalised to the original amount. The Hit order efficiency is the number of tracks that have the same hits in the same order before and after the conversion, normalised to the total number of tracks.

<table>
<thead>
<tr>
<th></th>
<th>RTs1→RTs2</th>
<th>SPTCs1→SPTCs2</th>
<th>RTs2→RTs3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit efficiency</td>
<td>99.54</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Hit order efficiency</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Hit purity</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Track efficiency</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
</tbody>
</table>

conversion is not lossless. Nonetheless, the modules can be verified by converting back-and-forth between the formats a second time. By then no problematic candidates or clusters are left anymore and the conversion should be lossless.

To demonstrate this, 100 events each containing 100 muons were simulated. These muons were simulated with a uniform distribution in a momentum range between 0.05 and 3 GeV and the full geometrical acceptance of the detector. The Monte Carlo based track finder was used to obtain RTs from these events. Subsequently, these RTs were converted to SPTCs and back to RTs twice in a row. The results from these conversions are presented in Table 3.1. It can be seen that the first back-and-forth conversion produces some loss of information as described above, while the following conversions do not.

Candidate based network creation

It is possible to replace the network creation explained in Section 3.2.2 with a network creation that is based on SpacePointTrackCandidates. In this case the network creation is straightforward: Only the SegmentNetwork is created. Every consecutive pair of SpacePoints in a candidate represents a node and every consecutive triplet of SpacePoints represents two connected nodes. The other networks could be created in a similar fashion.

In combination with the path finding module described in Section 3.2.3, this allows back-and-forth conversion and therefore validation of the path finding module. However, this time, it is possible to generate additional candidates in the first conversion, if either the candidates shared SpacePoints in the first place or any of the additional options of the path finding is activated. Again, the back-and-forth conversion is performed twice.

This is demonstrated in Table 3.2, where 100 events, containing 100 randomly generated muons each, were converted back-and-forth. If two or more of these muons happen to share a SpacePoint, the network will allow paths that mix the original tracks together.
Table 3.2: Path finder validation
The Monte Carlo based track finder was used to obtain RecoTracks from 100 events containing 100 muons each and subsequently the RecoTracks have been converted to SpacePointTrackCandidates. Finally a SegmentNetwork was constructed from these SpacePointTrackCandidates and the path finder created a new set of SpacePointTrackCandidates from this network.
The table shows the comparison of the two sets of SpacePointTrackCandidates for different configurations of the path finding algorithm.

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>Flexible seeding</th>
<th>Strict Seeding</th>
<th>Subsets + Flexible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.00</td>
<td>53.94</td>
<td>53.94</td>
<td>73.69</td>
</tr>
<tr>
<td>Fake rate</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Hit purity</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
</tbody>
</table>

However, the 0% fake rate makes clear that no SpacePoint was shared in this dataset; but, as described in Section 3.2.3, the different configurations allow the creation of clones.

In the Default configuration of the path finding module only the longest paths are selected. The validation shows that all candidates were reproduced successfully. The additional options create, as expected, further clones. Since the original candidates remain in the set, hit efficiency and hit purity also remain unchanged. Both of the options produce the same number of track candidates, but with different content and if both options are activated at the same time, the clone rate becomes even larger, exactly as it is visualised in Figure 3.3.

To be sure that the amount of additional candidates is deterministic, it was verified that the results stay the same if a second network is created, from the second set of SpacePointTrackCandidates, and given to the pathfinder with the same configuration.

SectorMap based network creation
Based on the successful tests described above, the network creation based on a SectorMap can be validated, by a validation of the complete candidate creation; every error observed during this validation has to be caused by the network creation. In the first step, the SectorMap is trained on Monte Carlo tracks of a specific sample. In the second step, the trained SectorMap is used to perform candidate creation on the very same sample.

Table 3.3 shows that the candidate creation reproduces the same candidates that were used to train the SectorMap for the following three scenarios:

1. In the first scenario the SectorMap is trained and validated on a single event containing five muons. It is expected that the SectorMap learns to accept the corresponding tracks and therefore the evaluation should exactly reproduce them. This is repeated 1000 times with different seeds. The results show that it is working.
3.2 Track candidate creation

Table 3.3: SectorMap based network creation
In the first scenario the SectorMap is trained and validated on a single event containing five muons. This is repeated 1000 times with different seeds. The second scenario ensures that the training works even if it is applied on multiple events. In this scenario the SectorMap is trained and validated on 10 events each containing five muons. This is repeated 100 times. In the last scenario the SectorMap is trained and validated on a large sample of 100 000 events, similar to the way it will be used later on.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Clone rate</th>
<th>Fake rate</th>
<th>Finding efficiency</th>
<th>Hit efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>3</td>
<td>23.36</td>
<td>8.02</td>
<td>100.00</td>
<td>100.00</td>
</tr>
</tbody>
</table>

2. The second scenario ensures that the training works even if it is applied on multiple events. In this scenario the SectorMap is trained and validated on 10 events each containing five muons. In this scenario the SectorMap is expected to learn to accept all tracks of the different events. Hence, during the evaluation all tracks in a single event can be reproduced. The training and evaluation is repeated 100 times and all candidates are reproduced successfully.

3. In the last scenario the SectorMap is trained and validated on a large sample of 100 000 events, similar to the way it will be used later on. In this scenario it is expected that additional candidates are produced, because the single SectorMap has to work for the large amount of different tracks. Therefore, there may be additional combinations of SpacePoints allowed if two tracks in an event happen to be close to each other and a similar track was present in a different event. This is a side effect of the SectorMap concept and cannot be prevented.

Hence, the complete candidate creation module chain was successfully validated and by using the track format conversion modules it is also possible to develop the remaining module chain of the VXDTF2 independent of the candidate creation. However, this section only verified that the algorithms and modules are working as expected. But it is still required to evaluate how well the concept works in a realistic scenario.

3.2.5 Evaluation of the candidate creation

In this section the performance of the candidate creation in terms of the figures of merit, defined in section 2.4, is determined for a realistic scenario. The finding efficiency is in theory strongly dependent on the quality of the training of the SectorMap, while the different options of the path finding algorithm only influence borderline cases. However, it is hard to argue how the SectorMap should be trained to obtain optimal results. The three types of filters stored in the SectorMap alone provide a variety of options on which variables to store. The amount of sectors per sensor and the variables to calculate from 2- and 3-SpacePoint filters are freely selectable. On top of that, there is also the question of selecting the kind of particles/tracks to train on. Should tracks that suffered from an exceptionally high amount of multiple scattering be allowed in the training set and
therefore loosen the cuts applied in the different filters, and, if not, at which amount of multiple scattering should be cut?

Some of these questions have already been examined in the context of the thesis describing the VXDTF [3] and there is a dedicated master thesis about the effects of multiple scattering in progress.

While the functionality of the modules after the candidate creation can be validated independently, their performance has to be evaluated on realistic scenarios. How well the algorithms cope with fake and clone candidates can only be estimated, if they are present in the first place. Therefore, this section not only determines the figures of merit for two different trainings of a SectorMap in order to demonstrate the influence of the training data set, but one of them is also used for the studies in the following sections.

**SectorMap configuration**

For this, a configuration based on what proved to be efficient for the VXDTF was chosen: Because the SectorMap will be used for SVD standalone track finding, only SVD layers are considered. Only transverse momenta in the range [0.02-3.15] GeV are considered and the following filters are activated.

2-SpacePoint filters: 3-SpacePoint filters:

- Distance2DXYSquared (distXY)
- Distance3DSquared (dist3D)
- Distance1DZ (distZ)
- SlopeRZ (slopeRZ)
- Distance3DNormed (nDist3D)
- CosAngleXY (angleXY)
- Angle3DSimple (angle3D)
- AngleRZSimple (angleRZ)
- CircleDist2IP (circleDist2IP)
- Pt (pT)
- DeltaSlopeRZ (deltaSlopeRZ)
- DeltaSoverZ (deltaSoverZ)
- DeltaSlopeSoverZ (deltaSlopeSoverZ)
- HelixParameterFit (helixParameterFit)
- CircleRadius

While the names of these filters give an impression about their functionality, they are also explained by the name given in the brackets in the thesis covering the first version of the VXDTF [3]. The 3-SpacePoint filter CircleRadius is implicitly defined in CircleDist2IP. The cuts of these filters are chosen in a way so that they cover the whole value distribution of the training dataset. In case of the sector friendships, each sensor is divided into a grid of nine sectors defined by the borders [0, 0.3, 0.7, 1].

**Trained SectorMaps**

There are two trained SectorMaps used in this thesis, to obtain at least some kind of estimation about the influence of the performance of the SectorMap. Both of these SectorMaps
3.2 Track candidate creation

share the same configuration but are trained on a different set of tracks. However, independently of the training set the same preselection of tracks is applied:

For every track only the first outgoing part is used and two SpacePoints of a single track cannot be related to the same sensor. If the latter case does not hold true one of the problematic SpacePoints is removed.

The first training set consists purely of muons randomly generated with $[0.05\text{-}3] \text{ GeV}$ transverse momentum and within the full geometrical detector acceptance. In total, the set contains two million events with ten muons each. The SectorMap trained on this sample will be called MuonMap.

The second training set is based on simulated $\Upsilon(4S)$ events and contains 1.8 million events in total. The SectorMap trained on this sample will be called Y4SMap.

Evaluation samples

The evaluation is based on two sets containing 10 000 $\Upsilon(4S)$ events each; one of them including current background estimations. They are called Y4S-Sample and Y4S+BKG-Sample respectively.

At the moment, this background is considered an overestimation compared to the actual expectations for Belle II on full luminosity, because the time information provided by the SVD is not utilised at all. It is predicted that the cluster time resolution, relative to the readout time window, will be precise enough to cut away up to $80\%$ of the SVD clusters produced by the background.

Evaluation of the SectorMap

An overview of the figures of merit is given in Table 3.4. Solely based on the left two columns, the evaluation on the Y4S-Sample, one can see that the ideal goal of finding all signal tracks could not be achieved by either of the SectorMaps.

It is however interesting that for both SectorMaps the finding efficiency is slightly different for the sample including background. This is probably caused by the fact that the Monte Carlo tracks may contain single clusters, while the VXDTF2 requires SpacePoints. If this is the case the track cannot be reconstructed by the VXDTF2 without adding a false cluster to the single correct cluster. Such a combination with a false cluster is significantly more likely to occur if background is included in the sample and therefore the finding efficiency can increase. On the other hand, if too many background SpacePoints are included in a track candidate, this candidate might not match anymore, therefore reducing the finding efficiency. Based on the same arguments, also the slight variation in the Hit efficiency can be explained.

In addition to the suboptimal finding efficiency, there are significant fake and clone rates due to tracks that happen to be in proximity. Adding better discriminating variables to the SectorMap might circumvent this, but will also increase the execution time of the algorithm significantly. Therefore, the fake and clone rates can only be reduced by a more careful selection of tracks to train on.
**Table 3.4: Candidate creation evaluation**

In this table two different SectorMaps have been applied on the same samples. The used SectorMaps and evaluation samples are further described on page 26 and following. The figures of merit are described in Section 2.4.

<table>
<thead>
<tr>
<th>Sample SectorMap</th>
<th>Y4S</th>
<th>Y4S + BKG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>51.78 ± 0.11</td>
<td>85.48 ± 0.04</td>
</tr>
<tr>
<td>Fake rate</td>
<td>28.98 ± 0.09</td>
<td>59.11 ± 0.04</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>95.83 ± 0.06</td>
<td>98.97 ± 0.03</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>98.81 ± 0.02</td>
<td>99.40 ± 0.01</td>
</tr>
<tr>
<td>Hit purity</td>
<td>90.38 ± 0.03</td>
<td>82.10 ± 0.01</td>
</tr>
<tr>
<td>Tracks/Event</td>
<td>27.56 ± 0.05</td>
<td>164.20 ± 0.13</td>
</tr>
</tbody>
</table>

With the addition of background this problems only gets worse. The last two columns of the table show that fake and clone rates increase significantly for both SectorMaps.

While in the case of no background the Y4SMap produced on average almost six times the amount of candidates per event compared to the MuonMap, this factor raises to nine if background is included. From here it is obvious that the realistic goal of the SectorMap training is to find a good trade-off between a high finding efficiency on the one hand, and low fake and clone rates on the other.

Therefore, it is important to understand where the trade-off takes place. One difference between the different training sets is the momentum distribution of the particles. While the Muons are generated following a uniform momentum distribution there are more particles with low transverse momentum in \( \Upsilon(4S) \) events. Figure 3.4 shows the finding efficiency binned by the transverse momentum. It is clear that the higher finding efficiency of the Y4SMap is mostly gained from particles with low transverse momentum. However, this comes with the trade-off of significantly increased fake and clone rates as was already displayed in Table 3.4.

This behaviour is a result of the increased curvature of particles with lower momenta. While trajectories of particles with a transverse momentum over 1 GeV can be approximated reasonably well with straight lines radiating from the interaction point, particles with a lower momentum feature a significant curvature. Due to this curvature and also multiple scattering a lot more sector friendships are possible for low momentum particles and therefore, there needs to be more training data to properly cover these. The disadvantage is that the SectorMap is less strict for low momentum particles and hence combining wrong hits to a track candidate becomes more likely.

Although the MuonMap also performs slightly worse for particles with a high transverse momentum on this evaluation sample, this difference is negligible compared to the difference for lower momenta. Nevertheless, this might be improved with additional high momentum training data. As explained above the additional training should hardly have any effect on the clone and fake rates because they are mainly produced by low momentum tracks.

In Figure 3.5 a SectorMap was evaluated on the Y4S-Sample after it was trained on an
increasing amount of muons. While the finding efficiency approaches the 100\% mark quite fast, the other quantities only begin to rise when the last missing percents of finding efficiency have to be trained. As shown in the lower right part of the figure the required disk space also grows with the training amount. While loosening the filters does not increase the required disk space, learning new sector friendships does. This shows that better filter variables are not sufficient to solve the issue, because the amount of sector friendships does not seem to converge with the current selection of training data. Without an effective first filter stage the filters have to be applied to a lot more combinations. Hence, adding additional filters would have a strong negative impact on the execution time.

The results obtained by this short evaluation show that the SectorMap loses most of its computing advantage without a carefully chosen training sample. Therefore, an optimised training of the SectorMap is a key part to further improvements of the ongoing development of the VXDTF2.

**Evaluation of the path finding**

As stated at the beginning of this section the path finding options only have a significant influence on the total number of track candidates without any further processing.

If a single SpacePoint is removed from a track one of the following cases may occur.

1. The original track has 100\% hit purity and therefore the shorter one too; the shorter candidate is a clone.
2. The SpacePoint that is removed contains background clusters; therefore the shorter candidate has a higher purity. If the original candidate is matched or a clone, the shorter one is also a clone. If the original candidate is a fake, the shorter one can be a match, or a clone or a fake.

3. The SpacePoint that is removed was correct; therefore the shorter candidate has a lower hit purity. The new candidate therefore either is in the same category as the original one, or is degraded to a fake.

The results for the different options are listed in Table 3.5. Based on the cases above all differences in the figures of merit can be explained.

The most obvious differences are the increased clone rate and the decrease of the fake rate. This is due to the increased number of track candidates. While the clone rate increases because of the many shorter candidates that also match, the fake rate is normalised to the total number of track candidates and because most of the new candidates are clones — due to the first two cases — the fake rate decreases.

The differences in finding and hit efficiency are insignificant and can be explained with the second case. The finding efficiency can only increase with additional candidates and only if the original track was a fake because it did not pass the purity requirement. Because the hit efficiency is based on all matched Monte Carlo tracks this is also influenced by the change in finding efficiency.
TABLE 3.5: Path finding options  In this table the candidate creation has been evaluated with the MuonMap on the Y4S sample. The different columns show the results for different configurations of the path finding algorithm.

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>Store subsets</th>
<th>Flexible seeding</th>
<th>Subsets + Flexible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>51.78 ± 0.11</td>
<td>70.70 ± 0.08</td>
<td>73.24 ± 0.07</td>
<td>80.96 ± 0.06</td>
</tr>
<tr>
<td>Fake rate</td>
<td>28.98 ± 0.09</td>
<td>22.04 ± 0.06</td>
<td>22.20 ± 0.06</td>
<td>18.49 ± 0.05</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>95.83 ± 0.06</td>
<td>95.83 ± 0.06</td>
<td>95.84 ± 0.06</td>
<td>95.84 ± 0.06</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>98.81 ± 0.02</td>
<td>98.80 ± 0.02</td>
<td>98.80 ± 0.02</td>
<td>98.80 ± 0.02</td>
</tr>
<tr>
<td>Hit purity</td>
<td>90.38 ± 0.03</td>
<td>92.72 ± 0.02</td>
<td>91.37 ± 0.02</td>
<td>92.89 ± 0.02</td>
</tr>
</tbody>
</table>

The increase in hit purity is also caused by the first case: Because most of the original candidates have a purity of 100 %, most of the shorter candidates are also clones with a purity of 100 %. This effect is further increased by the third case, because some of these short candidates are degraded to a fake and can therefore not have a negative influence of the hit purity.

However, which path finding configuration is generally beneficial has to be evaluated after the final candidate selection. But since the SectorMap produces so many candidates in the context of the Y4S+BKG sample even without these options, they are not evaluated further in this thesis. If this problem is solved at some point of the VXDTF2 development, the influence of these options should be evaluated.

Due to the high number of candidates produced by the Y4SMap it is not used further in this thesis. Instead, the MuonMap is considered to be a more realistic estimate for the current performance of the VXDTF2. If not stated otherwise, all following results are based on the Y4S+BKG sample, the MuonMap and the default configuration of the path finding (no additional options activated).

3.3 Track candidate quality estimation

After the preselection of reasonable track candidates, the set of track candidates, that describes the correct trajectories in the best possible way, has to be found. Therefore, a criterion describing the quality of a track has to be provided. Ideally the criterion would be a probability whether a track candidate correctly describes the trajectory of a particle originating from the collision. However, such a perfect quality estimator is not realistically
achievable, at least not within the given execution time limitations. On average there are 300 track candidates produced by the candidate creation and on average only 11 of them can be correct. Although 300 candidates sounds like a manageable number, the deviation is quite large. In some events the number of candidates is more than 2 orders of magnitude higher than the average and all of them have to be analysed.

In this section, the implemented quality estimators are presented, followed by a brief validation that they produce reasonable parameter estimates. The evaluation in the context of the VXDTF2 will be part of the candidate selection in Section 3.4.

The final fit, used for all tracks reconstructed with the different track finding algorithms, is implemented with a Deterministic Annealing Filter (DAF), described in [11]. This method is based on extrapolations, which include a detailed treatment of material effects and energy losses. In addition, hits can be weighted down if they do not match the remaining hits. In the extreme case of a zero weight, the hit is essentially removed from the track. This fit produces the best estimation of the helix parameters possible with the information gathered from the detector. However, a momentum seed at the innermost hit is required as an initial guess in order to apply the algorithm. This seed has to be determined by the individual track finding algorithms.

In this thesis this fit will be used as a reference at some points, denoted as DAF. In contrast, all quality estimators described in the following, will make certain approximations in order to be fast enough to deal with the large number of candidates. One approximation that all of them have in common is to neglect energy losses, because it would require a particle hypothesis to model the influences correctly.

Instead of aiming for the perfect quality estimator, that provides a probability to be correct for all candidates, the goal is to create a fast quality estimator that determines the affiliation of a cluster shared between multiple track candidates. Following this concept, in the sketch on the left, candidate two is defined to be competing against the candidates one and three, but those are not competing against each other. All three together form a set of competing candidates.

If a quality estimators is based on a fit, usually the $\chi^2$ value is used to determine how well the measurements, the clusters in the case of a track candidate, can be described by the model used in the fit. Therefore, the more clusters are shared between two candidates, the more realistic it becomes that the quality estimator is able to rank them properly. This removes the constraint to have global consistency for all candidates, as well as having a proper distribution of the qualities; only the ranking of competing tracks is relevant.

The ranking is represented by a quality indicator (QI), defined to be between 0 and 1, assigned to each track candidate. In reference to the probability that would be used ideally, a track candidate with a lower QI has a worse quality than a competing track candidate with a higher one. But as stated above, this does not have to hold true for track candidates that are not competing against each other.
3.3 Track candidate quality estimation

In the VXDTF a circle fit was used to determine the QI of a track candidate. This is based on the circular trajectory a charged particle describes in the plane perpendicular to the normal vector of a homogeneous magnetic field, due to the Lorentz force. However, such a circle fit cannot provide a useful QI for track candidates with only three SpacePoints, because three points in a plain merely provide enough degrees of freedom to define a circle, but not enough to allow a deviation from it: the QI for every track candidate with three SpacePoints would be 1. This is a significant disadvantage in the context of SVD standalone tracking, since in the current simulation roughly 8% of the particles produce less than four SpacePoints and on data recorded by the experiment this effect will eventually be higher, due to sensors becoming inefficient or defect.

To overcome the disadvantage of the CircleFit it was decided to use the full three dimensional information provided by the SpacePoints, although a three dimensional fit is more complex and therefore has a higher execution time. With the additional degrees of freedom it is possible to use a model based on a helix trajectory, even for track candidates with only three SpacePoints. It was decided to put two different variants of such a model to proof:

1. Neglect any material effects. Therefore, all uncertainties are assumed to be caused by the limited detector resolution. Such a helix fit can be realised by extending a circle fit with a linear regression perpendicular to the circle plane.

2. Material effects are not negligible but instead dominate the uncertainty. This approximation allows for another fast algorithm, called TripletFit (Section 3.3.3).

Although incorporating both uncertainties into the model would be better, it is far more complicated and computing intensive to do this properly.

In order to allow an easy comparison of the different approaches, the CircleFit used in the VXDTF was reimplemented for the VXDTF2. Furthermore, all quality estimators share the same interface in order to be trivially exchangeable. Below the different approaches are presented, followed by a brief validation in Section 3.3.5. Which of the algorithms performs best in the context of the VXDTF2 is discussed in the context of the candidate selection in Section 3.4.

3.3.1 CircleFit

This fit neglects any material effects and only incorporates the limited detector resolution. For this fit only the x- and y-components of the SpacePoints are considered, because the trajectories of charged particles should describe a circle in this plane, due the magnetic field along the z-axis.

The algorithm described in [12], sometimes referred to as Karimäki circle fit, calculates in a non-iterative way the circle parameters, their Gaussian distributed uncertainties and the corresponding $\chi^2$ value, as a measure how well the hits match to the fit model.

While the parameter uncertainties are neglected in the context of the VXDTF2, the $\chi^2$ value is used to determine the QI of a given track candidate. For a better comparison between track candidates of different lengths and because the QI has to be between 0 and 1 the $\chi^2$ probability is used as QI.
3.3.2 HelixFit

As stated above, the HelixFit is an extension of the CircleFit in order to incorporate the $z$ component of the SpacePoints. This extension is realised with a linear regression as described in [13], whereas the circle fit implementation described in the paper is replaced with the CircleFit implementation described above.

The QI can then be calculated in the same way as for the CircleFit, but now with the summed $\chi^2$ and $ndf$ values of the CircleFit and the linear regression.

3.3.3 TripletFit

In this approach the position uncertainty of the SpacePoints is assumed to be negligible compared to the uncertainty arising from multiple scattering. The algorithm is described in [14, 15], but the implementation for the VXDTF2 features two additions. A brief description is given below.

Any deviation from a helix caused by multiple scattering, can be described by a change of the flight direction of the corresponding particle at the location of the material interactions. This has to happen at the location of the SpacePoints, because the space between the sensors can be approximated to be empty. For a track candidate, it is not possible to determine such changes at the first and the last SpacePoint, because the direction respectively before and after them is unmeasured; only the inner SpacePoints have enough context to determine these changes.

Therefore, if one considers only three SpacePoints, a triplet, only the material at the middle SpacePoint is relevant. In [14, 15] it is shown, that each possible consecutive triplet of SpacePoints of a track candidate, can be treated independently and yet the individual analyses can be combined to form a meaningful analysis of the entire track candidate.

To gain a $\chi^2$ value of such a triplet, the probability that the observed amount of multiple scattering occurs has to be determined. This probability is based on a user defined multiple scattering uncertainty $\sigma_{MS}$.

As suggested in [14] the Highland approximation [16] is utilised, which in turn requires a material estimation in form of the average radiation length $X/X_0$:

$$\sigma_{MS} = \frac{B}{4.5R_{3D}\text{cm}^2} \sqrt{\frac{X}{X_0}}.$$  

(3.1)

In this equation, $B$ denotes the magnetic field strength and $R_{3D}$ is an estimation of the circle radius in the plane defined by the triplet. The average radiation length was taken from the material estimation of an SVD sensor in the design report of the Belle II experiment [2]. Additionally there are two modifications present in the code:

1. Since the amount of relevant material changes with the angle under which a particle traverses the sensor, it is modified to incorporate an impact angle estimation of the particle trajectory. This impact angle is approximated by the angle between the normal vector of the sensor plane and the vector between the first and second SpacePoint of a triplet.
2. Due to the limited space covered by the SVD the estimation of a radius can become quite imprecise, because it becomes almost indistinguishable from a straight line. This can lead to an estimation of the radius that is too large, which in turn results in a vanishing multiple scattering uncertainty. This can be prevented with a maximum threshold for the radius \( R_{3D} \) as it is used in the calculation of the multiple scattering uncertainty.

Both the scale of the multiple scattering uncertainty and the maximum radius threshold have been tuned in order to obtain optimal results during the candidate selection discussed in Section 3.4. This tuning was performed on track candidates reconstructed by the VXDTF2 on 10,000 \( \Upsilon(4S) \) events including background simulation.

Although the values obtained by this tuning seem to be inexplicable from a physics point of view:

a) the uncertainty is scaled with a factor five, equivalent to a 25 times increased material estimation,

b) the maximum threshold is at a radius equivalent to merely 100 MeV,

they proved to be beneficial to the overall evaluation. One possible influence is the neglected energy loss, which also changes the curvature at the centre SpacePoint of the triplet, a similar effect as additional material. Nevertheless, this discrepancy between the physical expectation and the determined values should be investigated further at some point, ideally before the start of the first data taking period of the experiment.

3.3.4 MCQualityEstimator

Additionally to the approaches described above, a quality estimator based on Monte Carlo information was implemented to simulate a perfect quality estimator. The QI is defined as

\[
QI = \begin{cases} 
0 & \text{if } N_{\text{matched}} \neq N_{\text{total}}, \\
1 - \frac{1}{N_{\text{total}}} & \text{otherwise.}
\end{cases}
\]

With \( N_{\text{matched}} \) being the number of SVD clusters a track candidate has in common with a single Monte Carlo track and \( N_{\text{total}} \) the total number of SVD clusters assigned to this track candidate. This guarantees that

a) only track candidates that have a purity of 100% have a QI unequal to zero,

b) longer track candidates have a higher QI than shorter ones,

c) all track candidates are comparable against each other.

This ‘perfect’ quality estimator can be used to validate the performance of the regular quality estimators in the context of the candidate selection described in Section 3.4 and will also be important for the improvement of the quality estimation discussed in Section 4.
3.3.5 Brief validation of the quality estimators

In this section the momentum estimations of the different fits are evaluated in order to determine whether the implementations work in principle. This is important, because the determined QI might not be suitable for the ranking it is used for in the VXDTF2, although the parameters of the model are estimated correctly. In such a scenario, without at least a brief validation, it would be unclear whether the QI cannot be used due to how it is calculated or whether there is an error in the fit itself.

Since the uncertainties of the determined fit parameters are either not used in the VXDTF2, or not calculated at all, instead of the more common pull distributions, the relative residuals \( r(x) \) are calculated to verify the momentum estimation:

\[
r(x) = \frac{x_{\text{truth}} - x_{\text{estimate}}}{x_{\text{truth}}}
\]  

These are calculated for all matched track candidates reconstructed by the VXDTF2 on 10,000 \( \Upsilon(4S) \) events including background simulation. The restriction to matched track candidates is due to the fact that fake track candidates are considered to be inherently wrong and therefore, any momentum estimation is considered inherently wrong as well; also, obviously there is no Monte Carlo truth information available. While clones could be considered, they distort the residuals, because they introduce additional estimations for the very same Monte Carlo track. Therefore, only the best track candidate for each Monte Carlo track is considered.

The results are compared against the results of the DAF with a pion hypothesis, which is the default hypothesis used for all tracks in the framework. This provides a good reference, since this validation is based on \( \Upsilon(4S) \) events and therefore about three quarter of the particles are pions.

Figure 3.6 shows the relative residual of the transverse momentum estimation of the CircleFit. The value displayed in the legend is the fraction of entries inside the shown histogram range compared to the complete dataset.

The CircleFit seems to produce reasonable results compared to the results of the DAF, although it features a slight bias towards lower transverse momenta. A more significant difference is that the CircleFit seems to produce more estimations that are outside of the displayed range. Although the HelixFit only extends the CircleFit to include the \( z \) components, this feature is not visible in Figure 3.7a, which shows the respective results for the HelixFit.

These bad estimations are caused by track candidates with only three SpacePoints. Since the CircleFit cannot produce a quality estimation for such a track candidate, it is not evaluated at all. Still, those track candidates are counted for the calculation of the displayed fraction of track candidates, because it is a significant feature of the CircleFit. If only track candidates with at least four SpacePoints are considered, both fits produce the same results for the transverse momentum.

Additionally to the estimation of the transverse momentum, both the HelixFit and the TripletFit provide an estimate of the three dimensional momentum vector. Therefore,
3.3 Track candidate quality estimation

This figure shows the relative residual of the transverse momentum estimation of the \textit{CircleFit} compared to the estimation of the \textit{DAF} (Pion hypothesis). The value displayed in the legend is the fraction of entries inside the shown histogram range compared to the complete sample.

For these two fits, also the relative residuals of the $z$ component of this vector are calculated and shown alongside the relative residual of the transverse component.

While both of these residuals show reasonable results in case of the \textit{HelixFit} (Figure 3.7), in case of the \textit{TripletFit} (Figure 3.8) a clear bias towards lower momenta is visible. However, the width of the distributions seems to be reasonable. The bias has not been examined further, because it has no impact on the performance of the \textit{VXDTF2}.

Overall, all of the fit methods seem to produce reasonable momentum estimations. Since they are meant to be rough estimates, a slight bias does not pose an issue. Nevertheless, this only shows that the implementations of the fits work in principle, but does not say anything about the usefulness of the QI. This will be evaluated as a part of the track candidate selection in Section 3.4, based on the figures of merit described in Section 2.4.
Figure 3.7: Relative residuals – HelixFit
This figure shows the relative residuals of the transverse and the $z$ component of the momentum estimation of the HelixFit compared to the estimations of the DAF (Pion hypothesis). The value displayed in the legend is the fraction of entries inside the shown histogram range compared to the complete sample.

Figure 3.8: Relative residuals – TripletFit
This figure shows the relative residuals of the transverse and the $z$ component of the momentum estimation of the TripletFit compared to the estimations of the DAF (Pion hypothesis). The value displayed in the legend is the fraction of entries inside the shown histogram range compared to the complete sample.
3.4 Track candidate selection

The goal of this part of the algorithm is to keep as few fake and clone track candidates as possible, while maintaining all the correct track candidates based on the QIs assigned to the track candidates.

A simple criterion would be to cut on the QI in hopes of a perfect ranking. However, as described in the previous section, such is not achievable. The implemented quality estimators do not provide a global ranking, but rather a ranking of competing track candidates. Figure 3.9 demonstrates that the TripletFit cannot be used for such a cut and the same holds true for the CircleFit and the HelixFit. In this figure the TripletFit was applied to all candidates and for the best 1000 of them Monte Carlo matching was performed. The figures of merit can then be evaluated in dependency of a cut on the QIs. It is clearly visible that the fake and clone rates cannot be removed without a significant loss of finding efficiency. Still, they can be reduced slightly.

A different approach was chosen: the criterion to obtain the final set of candidates is that no cluster may be assigned to more than one track candidate. This is an effective way to reduce the number of clone tracks: Clones describe, by definition, the same Monte Carlo track as their corresponding matched track candidate, but with only four SVD layers and a minimum track length of three SpacePoints it becomes very unlikely that they do not share a single cluster.

Only if a particle produced more than five SpacePoints in the four layers, due to the overlapping sensors or due to curling, it is possible to have two independent clones. Both of these options are unlikely, but still, because of them, the criterion described above cannot completely rule out clone candidates.

Additionally a large fraction of the fake track candidates should be removed by this criterion as well. These track candidates usually have a lower QI than their competing track candidates, because they actually do not describe a true trajectory, but instead combine different trajectories or include clusters originating from background. Therefore, if the quality estimator correctly identifies fake track candidates, only those fake track candidates who do not compete with any proper track candidate remain. This may either be because they consist purely of background clusters or because the track finding algorithm failed to add the signal clusters to the proper candidate.

The drawback of a strict realisation of this criterion is that it is possible that energy depositions from two different particles are combined to a single cluster. As a result of
that, the algorithm will produce two candidates that may only consist of correct clusters, but still share this merged cluster between each other. Since only one of them can remain in the final set of candidates, the other track candidate is eliminated as a consequence. An estimation how often this occurs for a typical $\Upsilon(4S)$ event will be given in Section 3.4.3 and one approach how to circumvent this issue will be discussed in Section 3.4.6.

To be able to implement this criterion it is key to know which track candidates are competing against each other. During the design of the VXDTF2 it was chosen to store this information in a symmetrical matrix holding the information which candidate shares how many clusters with each of the remaining candidates. In this way, the computation for each candidate is kept low. However, it turned out, that this approach does not scale good enough in the context of the massive amount of candidates produced under the presence of background clusters.

Instead of refactoring this, it was decided to only treat the $n$ track candidates with the highest QI in this way. As it was shown above, in Figure 3.9, such a cut is not beneficial based on the implemented quality estimators. However, it was concluded that the influence of this is acceptably low, if a large enough amount of track candidates is allowed. Hence, it was decided to select the best 1000 track candidates, which effects only 10\% of the events at all and leaves enough room in comparison to the average of 11 charged particles per $\Upsilon(4S)$ event. The absolute influence of this cut is shown later on in Table 3.6 for the MCQualityEstimator and in Table 3.7 for the remaining quality estimators.

To resolve the shared cluster conflicts, the following two approaches, that were already used in the VXDTF, were reimplemented.
3.4 Track candidate selection

3.4.1 Greedy algorithm

The first approach, called Greedy, is the most simple and straightforward implementation. It can be described by six simple steps:

1. All track candidates are sorted by their QI from highest to lowest and flagged as active.
2. Select the first candidate in this set.
3. All track candidates that compete with this track candidate are flagged as inactive.
4. Select the next candidate that is still active.
5. Repeat steps 3 and 4 until the end of the set is reached.
6. All track candidates that are still flagged as active define the final set of candidates.

3.4.2 Hopfield algorithm

A more sophisticated method to obtain a set of non-overlapping track candidates, called Hopfield in this thesis, is described in [17]. As the name suggests this approach is based on a Hopfield Neuronal Network, whose architecture is defined by the track candidates of each event. It is used to maximise the quality of the resulting set with regard to the requirement that there is no overlap allowed. In this context the quality of a set of track candidates is based on a sum of the QIs of the individual track candidates.

This approach allows to resolve the overlap of three or more competing tracks in a slightly different way than the Greedy algorithm. The difference is clear in the sketched scenario.

If the track candidate two has a higher QI than both of the other two track candidates, the Greedy algorithm always eliminates the candidates one and three. The Hopfield algorithm, however, is based on the quality of the complete resulting set of track candidates. Therefore, if the combined QIs of the track candidates one and three is higher than the QI of candidate two, only candidate two will be eliminated.

Obviously, this is only an advantage if all of the track candidates in this example are correct or if only track candidate two, the candidate with the highest QI, is a fake track. But the latter case is a rather serious problem of the quality estimator and not an advantage of this candidate selection. If either track candidate one or three in the example is a fake track, preferring them over track candidate two would be a bad decision.

While the results of the scenario described above stay the same for the Greedy algorithm, as long as the ranking stays the same, the result of the Hopfield changes with different absolute values of the QIs.

In conclusion, the Hopfield algorithm promises a slightly better overlap resolution on the cost of a slower execution time and an additional requirement on the distribution of the QIs. If this requirement is not fulfilled, it will perform worse than the Greedy algorithm. However, the the paper is more an empirical than a theoretical description and therefore does not provide a precise definition of this requirement.
3.4.3 Evaluation of the candidate selection

In this section the performance of the different quality estimators combined with the overlap criterion are evaluated. These steps are evaluated together, because the goal of the quality estimators is to provide a ranking for competing track candidates. This can be evaluated by verifying that the highest ranked track candidate of a set of competing track candidates describes the truth the best. Because the algorithms described above select these highest ranked track candidates, the performance of the quality estimator can be evaluated, by calculating the figures of merit of the final set of candidates. However, it is important to factor out the influence of

a) the candidate creation,

b) the overlap criterion.

While the drawback of the overlap criterion could be analysed by using Monte Carlo tracks as input instead of the candidate creation of the VXDTF2, there would not be any fake or clone rate. Without fake and clone rates the gain obtained by requiring the overlap criterion could not be evaluated.

Therefore, the candidate selection is, as described on page 31, evaluated on the Y4S+BKG sample. This baseline defining the boundary conditions will be used in Figures and Tables as a reference labelled All Candidates.

The optimal results that can be achieved by requiring a non-overlapping set of candidates are shown in Table 3.6. In this Table the ‘perfect’ MCQualityEstimator was used to be independent from the quality estimation.

The entry labelled Best candidates describes the loss introduced by considering only the best 1000 candidates, without any further candidate selection. Because the ‘perfect’ quality estimator was used, this cut should only affect the clone and fake rate. However, as shown in the table there is a minor decrease in finding efficiency and a slight increase in hit purity. This indicates that for a specific Monte Carlo track not a single corresponding track candidate with a purity of 100\% was created. If by chance all unclean track candidates corresponding to this Monte Carlo track are affected by the cut, the finding efficiency decreases. Because the unclean track candidate that was previously matched to the Monte Carlo track is missing after the cut, also the hit purity increases.

The influence of the overlap criterion is labelled MC greedy and has to be compared to the Best candidates entry. As expected, the clone rate is reduced to effectively zero and the fake rate by roughly 55\% points as well. On the downside the finding efficiency is also reduced by almost 2 percentage points. As already discussed before this is caused by Monte Carlo tracks sharing clusters and the loss can only be reduced by a more sophisticated implementation of the overlap criterion, like the Hopfield algorithm. Another way to reduce this effect will be discussed in Section 3.4.6.

The remaining fake rate is produced by track candidates that do not share any cluster with the track candidates selected utilising the Monte Carlo information. This may be either because some of the clusters corresponding to a Monte Carlo track are not assigned to the track candidate with the remaining clusters, or they consist purely out of background
### Table 3.6: Candidate selection best case
This table shows the result of the two basic steps of the candidate selection based on the `MCQualityEstimator`. In the first column all candidates produced by the candidate creation are evaluated. In the second column only the best 1000 candidates of each event are selected prior to the evaluation. Finally in the third column the `Greedy` algorithm is applied for each event.

<table>
<thead>
<tr>
<th></th>
<th>All candidates</th>
<th>Best candidates</th>
<th>MC Greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>76.71 ± 0.07</td>
<td>75.75 ± 0.07</td>
<td>0.41 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>87.81 ± 0.02</td>
<td>75.03 ± 0.03</td>
<td>19.81 ± 0.12</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>96.24 ± 0.06</td>
<td>96.20 ± 0.06</td>
<td>94.48 ± 0.07</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>97.13 ± 0.03</td>
<td>97.14 ± 0.03</td>
<td>97.21 ± 0.03</td>
</tr>
<tr>
<td>Hit purity</td>
<td>83.78 ± 0.02</td>
<td>84.12 ± 0.02</td>
<td>99.76 ± 0.01</td>
</tr>
</tbody>
</table>

### Table 3.7: Best candidate selection
This table shows the influence of the selection of the best 1000 candidates for the individual quality estimators.

<table>
<thead>
<tr>
<th></th>
<th>CircleFit</th>
<th>HelixFit</th>
<th>TripletFit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>75.92 ± 0.07</td>
<td>75.82 ± 0.07</td>
<td>76.11 ± 0.07</td>
</tr>
<tr>
<td>Fake rate</td>
<td>74.89 ± 0.03</td>
<td>75.05 ± 0.03</td>
<td>74.65 ± 0.03</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>96.04 ± 0.06</td>
<td>95.82 ± 0.06</td>
<td>96.17 ± 0.06</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>97.14 ± 0.03</td>
<td>97.03 ± 0.03</td>
<td>97.14 ± 0.03</td>
</tr>
<tr>
<td>Hit purity</td>
<td>84.09 ± 0.02</td>
<td>83.98 ± 0.02</td>
<td>84.06 ± 0.02</td>
</tr>
</tbody>
</table>

clusters. With Monte Carlo information these fake track candidates can be easily removed by applying a cut on the QI, but such a global cut might be impossible without the Monte Carlo information.

#### 3.4.4 Evaluation of the Greedy selection

As discussed before, the quality estimators that do not rely on Monte Carlo information are not able to achieve such a perfect candidate selection. However, similar to the Monte Carlo case, the 1000 best candidates selection only introduces only small losses. As shown in Table 3.7, the loss is indistinguishable within the statistical uncertainties in the case of the `TripletFit` and slightly worse for the `CircleFit` and the `HelixFit`.

The performance of the greedy algorithm applied with the different quality estimators is given in Table 3.8 and has to be compared with both the `Best Candidates` and `MC greedy` entries of Table 3.6. While the column `MC greedy` describes the best case scenario for hit and finding efficiencies, the column `Best Candidates` describes the worst case scenario for fake and clone rates pretty well.

The first thing to notice is that the differences to the `MC greedy` column are not that large. The finding efficiency for the `TripletFit` is way above 90% and the fake rate is less than a fourth of the fake rate after the candidate creation. With the clone rate at almost 0%, by construction, this is a way better result than what a simple cut on the QI, as shown at the beginning of this section in Figure 3.9, could provide.
**Table 3.8: Greedy selection quality estimators**

This table shows the result of the Greedy algorithm, applied for the different quality estimators.

<table>
<thead>
<tr>
<th></th>
<th>CircleFit</th>
<th>HelixFit</th>
<th>TripletFit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.41 ± 0.02</td>
<td>0.44 ± 0.02</td>
<td>0.43 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>21.73 ± 0.12</td>
<td>24.72 ± 0.13</td>
<td>20.62 ± 0.12</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>91.47 ± 0.09</td>
<td>87.51 ± 0.11</td>
<td>93.58 ± 0.08</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>95.24 ± 0.04</td>
<td>91.53 ± 0.05</td>
<td>96.36 ± 0.04</td>
</tr>
<tr>
<td>Hit purity</td>
<td>97.66 ± 0.02</td>
<td>96.11 ± 0.03</td>
<td>98.85 ± 0.01</td>
</tr>
</tbody>
</table>

Furthermore it is interesting to see that the HelixFit performs worse than the CircleFit, even though the HelixFit extends the CircleFit with additional information and should therefore provide a better quality estimation. Because the momentum residuals of the HelixFit look reasonable, as it was shown in Figure 3.7, this indicates that either the determination of the QI is not extended properly, or the uncertainty of the z component of the SpacePoints is not properly determined to begin with.

However, since the TripletFit produces better results, the search for the cause of this effect was not prioritised. Nevertheless, the HelixFit can still be used to provide a better momentum seed than the TripletFit for the final candidates, since the TripletFit featured a bias, as it was shown in Section 3.3.5.

The ranking provided by the TripletFit is also strictly better than the one of the CircleFit and performs only slightly worse than the ranking based on Monte Carlo information. In terms of finding efficiency the TripletFit has an advantage of 2% over the CircleFit used in the VXDTF and is only 1% below the MCQualityEstimator; the other quantities favour the TripletFit as well.

However, the fake rate cannot be reduced further without influencing the finding efficiency. While a final track candidate has a higher QI than all its competing track candidates, it still occurs that it has a lower QI than a fake candidate that is completely independent to it. Both the fake rate and the finding efficiency in dependence of the QI after the overlap resolution are shown in Figure 3.10. Similar to the Figure 3.9 it is still not possible to find a clean cut to remove all fake candidates after the overlap is resolved. Again, the fake rate could be reduced slightly, but because the number of track candidates is low enough at this point, to perform the final track fit on them, the cut decision is postponed in order to not cut prematurely.

### 3.4.5 Evaluation of the Hopfield selection

As described in section 3.4.2 the Hopfield algorithm can be superior to the Greedy algorithm if the QI distribution is suitable. In Table 3.9 the results of the Hopfield algorithm are shown for each available quality estimator.

As expected from the experience gained with the VXDTF, the Hopfield algorithm produces better results than the Greedy algorithm, if the CircleFit is used as the quality estimator. But it is interesting to see that this holds only true for the CircleFit. This shows that the
3.4 Track candidate selection

### Figure 3.10: Quality indicator cut after candidate selection

The data for this figure was produced analogous to Figure 3.9, but in this case the Greedy algorithm is already applied for every event, based on QIs produced by the TripletFit.

### Table 3.9: Hopfield selection quality estimators

This table shows the result of the Greedy algorithm, applied for the different quality estimators.

<table>
<thead>
<tr>
<th></th>
<th>CircleFit</th>
<th>HelixFit</th>
<th>TripletFit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.42 ± 0.02</td>
<td>0.55 ± 0.03</td>
<td>0.47 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>19.15 ± 0.12</td>
<td>28.10 ± 0.13</td>
<td>25.40 ± 0.13</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>91.69 ± 0.09</td>
<td>83.88 ± 0.12</td>
<td>89.44 ± 0.10</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>94.95 ± 0.04</td>
<td>88.70 ± 0.06</td>
<td>91.45 ± 0.05</td>
</tr>
<tr>
<td>Hit purity</td>
<td>97.66 ± 0.02</td>
<td>94.42 ± 0.04</td>
<td>95.58 ± 0.03</td>
</tr>
</tbody>
</table>

The Hopfield algorithm can improve how the overlap is resolved, but the algorithm is fragile. However, even with the improvement of the Hopfield algorithm the CircleFit still has a worse finding efficiency than the TripletFit evaluated with the Greedy algorithm.

Although the fake rate is lower in case of the CircleFit, the same argument as before can be made: A higher finding efficiency is preferred over a lower fake rate if the problem can be deferred to a later stage of the algorithm. In this case, the reduction of the fake rate is postponed, until at least the proper track fit, the DAF, is performed.

### 3.4.6 Addressing negative effects of the candidate selection

As already mentioned in the first part of Section 3.4 the drawback of requiring a non-overlapping set of track candidates is, that two trajectories of primary particles might be so close to each other at one point of their trajectory, that their energy depositions are combined to a single cluster. In this case, only one of the track candidates describing these two trajectories can satisfy the overlap requirement, thus rejecting the other track candidate. According to Table 3.6 this effect reduces the finding efficiency by almost 2%.

The loss induced by this effect can theoretically be prevented by only removing SpacePoints...
from the track candidates instead of removing the complete candidate. If the track candidate has more than three SpacePoints left, it can still be part of the final set. In this case the focus is on competing SpacePoints instead of competing track candidates as a whole. In order to decide which track candidate should do without the competing SpacePoint different strategies are possible.

Similar to before, the simple approach is to let the track candidate with the higher QI keep the SpacePoint. In the scenario sketched on the left, the drawback of this can be made clear. If both candidates one and two are correct, it is beneficial to let candidate one keep the competing SpacePoint, therefore replacing candidate two with candidate three, because otherwise candidate one would not be long enough anymore. But if candidate two has a higher QI than candidate one, the simple algorithm would eliminate candidate one. Again, it should be better, to take the benefits for the final set of candidates into account.

The approaches described above can be realised by adding subsets to all track candidates, where each track candidate in a subset is missing a single SpacePoint of the original candidate. Therefore, creating candidate three of the above example even before the decision is made. In this scenario the previously introduced candidate selection algorithms can be reused without any changes.

However, the drawback of implementing it in this manner is that there are a lot more candidates to take care of. This could be solved by a stricter best candidate selection, but it was already shown before that this might have a negative influence on the finding efficiency.

A more elaborated approach is to perform the candidate selection in two steps; this will be called the improved candidate selection.

Figure 3.11a shows a representation of the overlap matrix of a single event. Each node represents a track candidate and two candidates are connected if they are competing against each other; the position of a node in the figure has no further meaning. The nodes in this figure are colour coded to represent the matching state of their corresponding track candidate. If a track cannot be matched the node is coloured in bright red. The remaining colours identify the different Monte Carlo particles the track candidates are matched to. Although there are only 65 track candidates in this event, only a fourth of an average event, the structure of the matrix is quite complex with track candidates corresponding to several different Monte Carlo particles mixed within the substructures.

In the first step of the improved candidate selection only those track candidates are competing, that have at least two consecutive SpacePoints in common. This separates the ‘chaos’ into several substructures as shown in Figure 3.11b. The colour coding shows that the sub matrices do not mix together different Monte Carlo tracks anymore and therefore, the set of the best candidates per substructure describes this event pretty well.

Obviously, this is an example chosen to visualise this effect and the separation into clean substructures does not have to hold true for every event; especially not for events with a large amount of candidates. Table 3.10 shows how resolving the overlap of these sub matrices influences the figures of merit based on the complete Y4S+BKG sample and the
### Table 3.10: First step of the two step candidate selection

The first column shows the results without any candidate selection as a reference for the other two columns. The latter ones show the results after the first step of the two step candidate selection has been performed with the respective algorithm and based on the MCQualityEstimator.

<table>
<thead>
<tr>
<th></th>
<th>All candidates</th>
<th>Best candidates</th>
<th>Greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>76.71 ± 0.07</td>
<td>22.82 ± 0.12</td>
<td>22.87 ± 0.12</td>
</tr>
<tr>
<td>Fake rate</td>
<td>87.81 ± 0.02</td>
<td>55.25 ± 0.09</td>
<td>56.36 ± 0.09</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>96.24 ± 0.06</td>
<td>95.94 ± 0.06</td>
<td>95.97 ± 0.06</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>97.13 ± 0.03</td>
<td>97.11 ± 0.03</td>
<td>97.11 ± 0.03</td>
</tr>
<tr>
<td>Hit purity</td>
<td>83.78 ± 0.02</td>
<td>93.02 ± 0.04</td>
<td>93.01 ± 0.04</td>
</tr>
</tbody>
</table>

MCQualityEstimator. It becomes clear that for the substructures a simple best candidate selection is indistinguishable within the statistical uncertainty from a overlap resolving based on the Greedy algorithm. Even compared to the case without any selection, the finding efficiency remains nearly unchanged, but on the plus side, less than a third of the clone rate and two thirds of the fake rate remain. This is equivalent to a reduction of the average number of track candidates from $335.28 ± 0.18$ to $27.48 ± 0.05$, one order of magnitude. The reduction is especially effective for events with an extreme amount of candidates.

After the overlap of the substructures is resolved, the track candidates might still overlap with each other, if the original criterion for competing tracks is applied; this is shown in Figure 3.12a. Such an overlap has to be resolved in the second step. The highlighted two candidates, green and orange, in this figure show that simply dividing the candidate selection into two steps can in fact be counterproductive. While in the original approach candidates corresponding to both of the highlighted Monte Carlo tracks could have survived, now, after the first step, the only two remaining candidates corresponding to the two Monte Carlo tracks share at least one cluster. This is where it becomes important to add subsets to the track candidates. The overlap matrix incorporating these subsets is shown in Figure 3.12b. From the highlighted substructure, consisting of the two original green and orange tracks as well as their subsets, it is clear that with the addition of subsets, for both of the two Monte Carlo tracks a corresponding track candidate can be part of the final set. In general, this holds true if they are long enough and do not share more clusters than what the subset is anticipating for.

The difference between the original and the improved candidate selection for the highlighted candidates is subtle. There is a difference how the two approaches achieve to keep both candidates as a part of the final set of candidates. In the original approach the different candidates that are matched to the same Monte Carlo track include different amounts of fake SpacePoints. In order to have both candidates in the final set one of them has to have a fake SpacePoint instead of the competing one. In the other case, after the first step, ideally, the candidate with the least amount of fake SpacePoints is selected. To be able to keep both candidates one of them ‘loses’ the competing SpacePoint, but does not replace it with a fake SpacePoint. Therefore, even if the improved candidate selection turns out to have no advantage in finding efficiency, this effect should still increase the purity.
In Table 3.11 the final results of the improved candidate selection are compared against the original approach. This table includes results based on both the MCQualityEstimator and the TripletFit.

In the ideal case of the evaluation based on the MCQualityEstimator the improved candidate selection recovers half of the inefficiency originally introduced by the overlap criterion. Although it should technically be able to recover all of the inefficiency, this may require the removal of more than one SpacePoint per candidate, or that the competing track candidates were not long enough; probably due to detector inefficiencies or imperfect candidate creation. With regards to the hit quantities the values change only in the order of per mill. As expected, the hit purity increases slightly and the hit efficiency is slightly reduced, both due to removing SpacePoints from the track candidates. Due to the changed candidate selection the fake happens to be reduced slightly as well.

In the case of the TripletFit the effect is smaller because of the imperfect quality estimation, but nevertheless, an improvement in finding efficiency of roughly half a percent is visible. In contrast to the MCQualityEstimator, the TripletFit does not feature a hard coded favour of longer track candidates, which is probably one reason for the higher drop in hit efficiency; more SpacePoints than necessary are removed.

Although the improved candidate selection features several additional steps, it is in fact slightly faster, because the first step can be implemented in a fast way:
3.4 Track candidate selection

![Figure 3.12: Disentangling the overlap matrix step 2](image)

Both figures encode the information in the same way as Figure 3.11, but include different sets of track candidates. In both figures, two track candidates are connected, if they share at least one cluster – the same criterion as for Figure 3.11a. In figure (a) the candidates remaining after the best candidate selection of the first step are shown and in figure (b) subsets are added to these candidates.

The information which tracks are part of a competing set is already present in the form of the SegmentNetwork created during the candidate creation. In this network each node is a combination of two SpacePoints. Therefore, all track candidates created from an independent sub-graph, represent a track family. Every track in such a subset has at least two consecutive SpacePoints in common with another track candidate in this subset and less consecutive SpacePoints in common with any track candidate from a different subset.

During the candidate creation a simple flood fill algorithm can be used to assign a family to all nodes. When the paths in the network are converted to track candidates, this family is copied from the nodes to the candidate. Therefore, it is known for each track candidate

<table>
<thead>
<tr>
<th>Quality estimator</th>
<th>MCQualityEstimator</th>
<th>TripletFit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection method</td>
<td>Improved</td>
<td>Original</td>
</tr>
<tr>
<td>Clone rate</td>
<td>0.40 ± 0.02</td>
<td>0.41 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>18.56 ± 0.11</td>
<td>19.81 ± 0.12</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>95.37 ± 0.07</td>
<td>94.48 ± 0.07</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>96.97 ± 0.03</td>
<td>97.21 ± 0.03</td>
</tr>
<tr>
<td>Hit purity</td>
<td>99.87 ± 0.01</td>
<td>99.76 ± 0.01</td>
</tr>
</tbody>
</table>

Table 3.11: Improved candidate selection comparison

The keyword Improved denotes the two step approach including the addition of subsets of track candidates.
which family it belongs to and the overlap matrix only needs to include track candidates of one family at a time.

Decomposing the problem into subsets of overlapping candidates therefore boils down to a flood fill algorithm in the network, which is faster than performing the same step in a later stage, because the required information is already present in the network. The computational advantages over the original approach, smaller overlap matrices and a simple best candidate selection in the first step, then outweighs the cost of separating the problem into subproblems and the addition of subsets to the candidates in the second step.

In this last section, no results including the Hopfield algorithm were presented, even though it is supported by the improved candidate selection. This is simply due to the strictly worse performance in any tested configuration, except for the CircleFit. Overall, the TripletFit in combination with the Greedy algorithm produces better results, as it was also the case for the original candidate selection.

The column TripletFit, Improved of Table 3.11 describes the final set of candidates of the current best configuration of the VXDTF2. These results will be summarised and put into context with the previous version of the algorithm in Section 5.1. The following section, on the other hand, discusses a possible method to further improve these results.
4. Improved quality estimation based on a multivariate analysis method

The quality estimators that were introduced in Section 3.3 only incorporate the global coordinates of the SpacePoints and in case of the CircleFit and the HelixFit also the uncertainties of these. But there is additional information provided by the detector systems and furthermore, the momentum estimation provided by the quality estimators is not utilised in the candidate selection. All this information could be easily incorporated by using a multivariate analysis (MVA) method. An extensive introduction to these methods can be found in the Book[18]. In this chapter a first take on these methods is evaluated, to test whether the additional information can improve the quality estimation of the track candidates.

In addition to utilising this information, the MVA method can also be used to transform the output distribution of the quality estimator. Hence the use of an MVA method has the possibility to tackle two of the issues present in the current quality estimation:

- imperfect ranking,
- unsuitable QI distribution for the Hopfield algorithm.

For this first approach a simple FastBDT[19] is used, with the default configuration present in the BASF2 framework, as it was chosen by the author of the FastBDT:

- 200 trees
- 3 levels
- 0.1 shrinkage

These hyper parameters have not been tuned for this study. The output of the FastBDT is constructed in a way that its distribution represents a probability, based on the training data set. If the QI distribution is binned, each bin has a signal to background ratio equal to a value within its QI range.

4.1 Input features

The variables that can be turned into features passed to the MVA method can be separated into track candidate variables and cluster variables.
Track candidate variables

These variables describe properties of the whole track candidate. Except for the number of SpacePoints, they are all dependent on the chosen quality estimator.

**NSpacePoints** Number of SpacePoints in this track candidate.

**Chi2** \( \chi^2 \) provided by the chosen quality estimator.

**QI** \( \chi^2 \) probability based on Chi2 and NSpacePoints.

**\( p_T / \vec{p} \)** Momentum estimation provided by the chosen quality estimator. The three dimensional momentum estimation is only available for the TripletFit and the HelixFit; the CircleFit can only provide a transverse momentum estimation.

Cluster variables

These variables describe properties of the individual clusters assigned to the track candidate; at least six clusters per candidate.

**Charge** the total amount of energy deposited in this cluster in units of electron charge,

**MaxCharge** the largest amount of energy deposited in a single strip of the cluster,

**Size** the number of strips combined in this cluster,

**ClsTime** the time estimate relative to the readout window,

**ClsTimeSigma** the estimated uncertainty of the time estimation.

As already mentioned in the evaluation sample description on Page 27 and in Section 2.1.3, the time information is not properly simulated in the current state of the software and therefore cannot be included in the context of this study. However, as soon as it is available, it should provide useful information to classify whether a random background cluster is part of a track candidate.

It was chosen to calculate one additional variable from these: the ratio \( \text{Charge/Size} \). This variable encodes the expectation that more energy is deposited in a larger cluster than in a smaller one.

Feature selection

While some of the track candidate features are available independently of the used quality estimator: NSpacePoints, QI, Chi2 and \( p_T \), the estimation of the three dimensional momentum vector is only available for the TripletFit and the HelixFit. This vector can be used in different ways. Obviously, the \( x, y \) and \( z \) components of the vector can be used as features, but it is also possible to use the polar coordinates, \( |p|, p_\phi, p_\theta \). Since it is not perfectly clear which of these representations is favourable, both of them are used. The variable ranking of the FastBDT should be able to select the best set of features.
4.1 Input features

The cluster variables have to be treated specially, to be able to use them as features, because not every track candidate has the same amount of SpacePoints and therefore clusters. There are several strategies to treat such a variable input size:

**Combination** create features that describe the distribution of a cluster variable in the context of this track candidate,

**Default Value** set values of features to infinity if they are not available for this track candidate,

**None Value** set values of features to *Not a Number* (NaN), if they are not available for this track candidate.

The two latter options are distinguished, because on the one hand they have to be treated differently in the MVA implementation and on the other hand, they have different implications. The FastBDT supports both:

If the missing feature is set to an infinite value, it can be considered with a cut in the FastBDT. Therefore, the absence of a feature can be utilised.

On the other hand, if the missing feature is set to NaN, the value is ignored during the training and evaluation. If a cut of the FastBDT is applied to such a missing value, the current node is treated as a final leaf instead. In this way, the absence of a feature is ignored.

For the application at hand, both options are viable. While it is a valuable information if a candidate has less clusters, this information is already encoded in the variable NSpacePoints. Nevertheless, infinity values were chosen to represent a missing value.

Generally, if the variables are not combined to a track variable, it has to be chosen how many variables are considered. As an example the variables of eight clusters can be chosen to be included. If there are more clusters, the variables of those are discarded, if there are less, the infinity value is inserted for the variables.

Several of the following approaches have been tested:

1. combining all cluster variables and use the mean, standard deviation, min and max of these,
2. allow eight clusters and use infinity values for these features, if not available,
3. calculate the mean for the clusters related to a SpacePoint and then consider four SpacePoints,
4. calculate the mean for the clusters related to a SpacePoint and then consider the four SpacePoints with the highest Charge to Size ratio.

From these, the first option turned out to perform the best, which might be due to the simple MVA model. The increased amount of features present in the other options has to be combined in a useful way, which is harder to train, compared to the smaller amount of precombined values. Also, while the third and fourth options introduce information by defining relations between the clusters, also some information is lost by utilising only the mean of the two values.
4 Improved quality estimation based on a multivariate analysis method

Figure 4.1: Example features

In this figure two examples of track candidate features calculated from the cluster variables are shown. The signal and background distributions are normalised independently to be able to clearly visualise both, even though there is a lot more background than signal.

Especially the minimum SeedCharge and the standard deviation of the Charge of a track candidate allow for a good separation of track candidates that incorporate background clusters and clean track candidates. The distributions of these features are shown in 4.1a and 4.1b, as an example for the combined cluster variables.

4.2 Training goal

The training goal is based on the quality estimation of the MCQualityEstimator described in Section 3.3.4. However, all results unequal to zero were replaced by the value one, because the FastBDT only supports a binary training variable. This binary variable therefore encodes whether a track candidate has a 100% hit purity or not. Therefore, track candidates of different length are treated the same. But if they refer to the same Monte Carlo track, the longest track candidate should be preferred from a track finding point of view.

However, the path finding described in Section 3.2.3, is configured to select the longest possible paths for each seed node. Therefore, although there may be several alternative paths, most of them have the same length. In this scenario the chosen training goal should work perfectly fine.

4.3 Training

For the training of the MVA method, two additional samples, similar to the Y4S+BKG sample, have been simulated. One training sample containing 100 000 events and one evaluation sample containing 10 000 events. The features and the training target have been extracted for each track candidate produced by the candidate creation with the MuonMap. Therefore the training data represents the evaluation data as precisely as possible.
The MVA method has been trained both with and without the cluster features, in order to be able to differentiate between the benefits of the cluster features and the benefits of the MVA method based on the track candidate features. Because of the different track variable distributions, both of these configurations were trained for each quality estimator independently. However, for a more complex MVA approach, one could pretrain on the variables that are independent of the quality estimator and then finalise by including the variables that are not.

During the training half of the training data set is used for the actual training and half is used for evaluating its progress. The selection of the final cluster features was based on the evaluation sample.

### 4.4 Evaluation

The FastBDT was evaluated on the Y4S+BKG sample for each quality estimator, once without the cluster features and once including them.

As a reference, the results without the application of any MVA method are given in Table 4.2a. In comparison to the results based on the FastBDT without the cluster variables, Table 4.2b, improvements in fake rate and finding efficiency are visible for all quality estimators. Although there are fluctuations in the clone rate, these are negligible.

Especially the HelixFit performs better than without the FastBDT and even overcomes the CircleFit, as it was expected in the first place. The FastBDT seems to be able to correct for the error made in the calculation of the QI of the HelixFit. Alongside the improvements of the fake rate and finding efficiency, the HelixFit is the only quality estimator that also shows a significant improvement in hit efficiency and purity. This is probably due to the bigger improvement of the QI, compared to the other quality estimators. Still, the TripletFit performs the best of the three quality estimators.

If the cluster features are included, shown in Table 4.2c, the results for all quality estimators improve even further. Compared to Table 4.2a, even the hit purity is now increased significantly for all three quality estimators. The CircleFit improves the most, bringing the three quality estimators closer to each other. Nevertheless, the TripletFit remains the best, performing almost as well the MCQualityEstimator used to determine the training target. This is compared in Table 4.3. The initial difference in finding efficiency between the vanilla TripletFit and the MCQualityEstimator of 0.9% is reduced to merely 0.18%, a difference, that is almost indistinguishable within the statistical uncertainty on the used evaluation sample.

Only in terms of reducing the fake rate, the MCQualityEstimator shows a clear advantage: While all fakes can be removed in the case of MCQualityEstimator, due to its absolute knowledge, this is not possible for the TripletFit, even if the FastBDT is used. But, compared to the the vanilla TripletFit, a cut on the QI after the candidate selection allows for a significantly better fake rate reduction when the FastBDT is used. This is shown in Figure 4.2: Without a significant loss in finding efficiency, the fake rate can be halved. This allows for a cut even before the final fit of the track finding, the DAF.

At last, as stated in the introduction to this topic, Section 4, the FastBDT is constructed in a way that the QI represents a probability. This change of the QI distribution might
Table 4.1: FastBDT results – Greedy
In this table the Greedy algorithm was applied based on the QIs produced by a FastBDT, trained on the different quality estimators. As a reference, the first table contains the vanilla result without the application of any MVA method.

(a) Vanilla – Greedy

<table>
<thead>
<tr>
<th>Quality Estimator</th>
<th>CircleFit</th>
<th>HelixFit</th>
<th>TripletFit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.41 ± 0.02</td>
<td>0.44 ± 0.02</td>
<td>0.43 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>21.73 ± 0.12</td>
<td>24.72 ± 0.13</td>
<td>20.62 ± 0.12</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>91.47 ± 0.09</td>
<td>87.51 ± 0.11</td>
<td>93.58 ± 0.08</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>95.24 ± 0.04</td>
<td>91.53 ± 0.05</td>
<td>96.36 ± 0.04</td>
</tr>
<tr>
<td>Hit purity</td>
<td>97.66 ± 0.02</td>
<td>96.11 ± 0.03</td>
<td>98.85 ± 0.01</td>
</tr>
</tbody>
</table>

(b) FastBDT – without cluster features – Greedy

<table>
<thead>
<tr>
<th>Quality Estimator</th>
<th>CircleFit</th>
<th>HelixFit</th>
<th>TripletFit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.40 ± 0.02</td>
<td>0.42 ± 0.02</td>
<td>0.43 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>21.61 ± 0.12</td>
<td>21.29 ± 0.12</td>
<td>20.37 ± 0.12</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>91.73 ± 0.09</td>
<td>92.26 ± 0.08</td>
<td>93.91 ± 0.08</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>95.27 ± 0.04</td>
<td>95.18 ± 0.04</td>
<td>95.93 ± 0.04</td>
</tr>
<tr>
<td>Hit purity</td>
<td>97.68 ± 0.02</td>
<td>97.78 ± 0.02</td>
<td>98.86 ± 0.01</td>
</tr>
</tbody>
</table>

(c) FastBDT – including cluster features – Greedy

<table>
<thead>
<tr>
<th>Quality Estimator</th>
<th>CircleFit</th>
<th>HelixFit</th>
<th>TripletFit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.44 ± 0.02</td>
<td>0.43 ± 0.02</td>
<td>0.44 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>20.78 ± 0.12</td>
<td>20.49 ± 0.12</td>
<td>20.04 ± 0.12</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>93.04 ± 0.08</td>
<td>93.35 ± 0.08</td>
<td>94.30 ± 0.07</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>95.56 ± 0.04</td>
<td>95.73 ± 0.04</td>
<td>96.05 ± 0.04</td>
</tr>
<tr>
<td>Hit purity</td>
<td>98.47 ± 0.02</td>
<td>98.63 ± 0.02</td>
<td>99.23 ± 0.01</td>
</tr>
</tbody>
</table>

improve the results of the Hopfield candidate selection and indeed, as can be seen in Table 4.5b, all quality estimators show improved results compared to their counterpart without the MVA method, Table 4.5a. Interestingly, the ranking of the quality estimators changed between these two tables; with the MVA method, it is in the same order as for the Greedy algorithm. This shows, that the FastBDT indeed changed the absolute values of the QI in a way that affects the performance of the Hopfield. The differences between the three quality estimators can now be explained by the different accuracy of the estimations, rather than the different absolute values.

However, in comparison to the results obtained with the Greedy algorithm, the Hopfield algorithm performs worse for every quality estimator, if the FastBDT is used. Therefore, although the distribution of the QI is comparable for the different quality estimators, it seems to be not ideal for the Hopfield algorithm.
Table 4.3: FastBDT improvements
In this table the FastBDT including cluster features and based on the TripletFit, is compared against the vanilla TripletFit and the MCQualityEstimator that was used to determine the training target of the FastBDT. In all cases the Greedy candidate selection is applied.

<table>
<thead>
<tr>
<th></th>
<th>FastBDT</th>
<th>MCQE</th>
<th>TripletFit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.44 ± 0.02</td>
<td>0.41 ± 0.02</td>
<td>0.43 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>20.04 ± 0.12</td>
<td>19.81 ± 0.12</td>
<td>20.62 ± 0.12</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>94.30 ± 0.07</td>
<td>94.48 ± 0.07</td>
<td>93.58 ± 0.08</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>96.05 ± 0.04</td>
<td>97.21 ± 0.03</td>
<td>96.36 ± 0.04</td>
</tr>
<tr>
<td>Hit purity</td>
<td>99.23 ± 0.01</td>
<td>99.76 ± 0.01</td>
<td>98.85 ± 0.01</td>
</tr>
</tbody>
</table>

In Table 4.2c the FastBDT based on the TripletFit, the best setup for both the Hopfield and the Greedy algorithms, is used to compare them. While the Hopfield algorithm features a lower fake rate with also a lower finding efficiency as trade-off, an even better trade-off can be obtained by reasonable cuts on the QI after the application of the Greedy algorithm. The variation of this trade off in dependency of the cut value was already shown in Figure 4.2b. In the last two columns of this table, two cuts are shown, to ease the comparison. Obviously, both of these cuts result in better figures of merit than the Hopfield algorithm.

Although the study was kept simple, the results show improvements of the results for both the FastBDT trained on the track candidate features and the FastBDT including the additional cluster features. The largest part of the possible efficiency gain of about 1% in case of the TripletFit is covered by the FastBDT. The big advantage however, is that the fake rate can be halved with only minor finding efficiency losses. A more elaborated study might yield even better results; for example based on a different architecture of the MVA method or different features.

Based on these results, the configuration that turned out to be the best is used in the context of the improved candidate selection. The results are presented in Table 4.7. The MVA method also improves the figures of merit in this context, except for the hit efficiency.

While a slight drop in hit efficiency is to be expected, because some of the SpacePoints of the track candidates are removed, such a drastic drop is not observed when either the vanilla TripletFit or the MCQualityEstimator is used, as it was already shown in Table 3.11. This is likely to be caused by the problematic training goal discussed in Section 4.2. The training target works for the first step of the improved candidate selection, the best candidate selection of each track family, because there are usually no alternative candidates of different lengths. But in the second step, shorter alternatives are explicitly added to the set of track candidates. The only way to differentiate between these alternatives is their purity, but since the best candidate selection in the first step removes most of the track candidates with a purity below 100%, most of the shorter alternatives feature also a purity of 100%. In this scenario the MVA method cannot differentiate between the alternatives effectively, leading to a drop in hit efficiency, because there are more SpacePoints removed than necessary.
Figure 4.2: Quality indicator cut after candidate selection

In this figure, for different QI cuts, it is plotted how the finding efficiency and the fake rate are changing relative to their value without any cut; the absolute values without cut are stated in the legend. As a reference, the Figure 3.10 based on the vanilla TripletFit is shown again on the left. In comparison, the figure on the right is based on the FastBDT including the cluster features and on the vanilla TripletFit. In both cases the quality estimators were used to determine a QI for every track candidate and for the best 1000 of them Monte Carlo matching was performed.

This shows, that in further studies, either the training goal should be modified, or the bias towards longer track candidates has to be introduced by other means.
### 4.4 Evaluation

**Table 4.4: FastBDT results – Hopfield**

In this table the Hopfield algorithm was applied based on the QIs produced by a FastBDT, trained on the different quality estimators. As a reference, the first part contains the result without the application of any MVA method.

(a) **Vanilla – Hopfield**

<table>
<thead>
<tr>
<th></th>
<th>CircleFit</th>
<th>HelixFit</th>
<th>TripletFit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.42 ± 0.02</td>
<td>0.55 ± 0.03</td>
<td>0.47 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>19.15 ± 0.12</td>
<td>28.10 ± 0.13</td>
<td>25.40 ± 0.13</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>91.69 ± 0.09</td>
<td>83.88 ± 0.12</td>
<td>89.44 ± 0.10</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>94.95 ± 0.04</td>
<td>88.70 ± 0.06</td>
<td>91.45 ± 0.05</td>
</tr>
<tr>
<td>Hit purity</td>
<td>97.66 ± 0.02</td>
<td>94.42 ± 0.04</td>
<td>95.58 ± 0.03</td>
</tr>
</tbody>
</table>

(b) **FastBDT – including cluster features – Hopfield**

<table>
<thead>
<tr>
<th></th>
<th>CircleFit</th>
<th>HelixFit</th>
<th>TripletFit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.41 ± 0.02</td>
<td>0.43 ± 0.02</td>
<td>0.43 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>15.30 ± 0.11</td>
<td>14.68 ± 0.11</td>
<td>14.07 ± 0.11</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>92.04 ± 0.09</td>
<td>92.46 ± 0.08</td>
<td>92.91 ± 0.08</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>95.45 ± 0.04</td>
<td>95.62 ± 0.04</td>
<td>95.91 ± 0.04</td>
</tr>
<tr>
<td>Hit purity</td>
<td>98.44 ± 0.02</td>
<td>98.62 ± 0.02</td>
<td>99.14 ± 0.01</td>
</tr>
</tbody>
</table>

**Table 4.6: Hopfield vs Greedy**

This table is based on the FastBDT that includes the cluster features and was trained on the QIs obtained from the TripletFit. Due to how the cuts in the last two columns have been applied, the hit efficiency could not be calculated, but no significant changes are to be expected.

<table>
<thead>
<tr>
<th></th>
<th>Hopfield</th>
<th>Greedy (QI &gt; 0.0001)</th>
<th>Greedy (QI &gt; 0.01)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.43 ± 0.02</td>
<td>0.41 ± 0.02</td>
<td>0.34 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>14.07 ± 0.11</td>
<td>10.13 ± 0.09</td>
<td>5.12 ± 0.07</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>92.91 ± 0.08</td>
<td>94.03 ± 0.08</td>
<td>92.57 ± 0.08</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>95.91 ± 0.04</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Hit purity</td>
<td>99.14 ± 0.01</td>
<td>99.29 ± 0.01</td>
<td>99.42 ± 0.01</td>
</tr>
</tbody>
</table>

**Table 4.7: FastBDT – improved candidate selection**

In this table the improved candidate selection was applied based on the QIs produced by a FastBDT including the cluster features, trained on the TripletFit. As a reference the results based on the vanilla TripletFit are shown in the column denoted Improved – Vanilla.

<table>
<thead>
<tr>
<th></th>
<th>Improved – MVA</th>
<th>Improved – Vanilla</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.43 ± 0.02</td>
<td>0.40 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>19.56 ± 0.12</td>
<td>20.03 ± 0.12</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>94.98 ± 0.07</td>
<td>94.01 ± 0.08</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>90.03 ± 0.05</td>
<td>95.36 ± 0.04</td>
</tr>
<tr>
<td>Hit purity</td>
<td>99.39 ± 0.01</td>
<td>98.95 ± 0.01</td>
</tr>
</tbody>
</table>
4.5 Feasibility discussion

Even though the additional MVA method shows significant improvements for the VXDTF2 on simulated data, one has to make sure that this holds true for data recorded by the experiment and that it is in agreement with the execution time limitations for the VXDTF2.

The latter point does not pose a major problem. The additional execution time for the MVA method turned out to be in the order of ms, as it will be shown in Table 5.1. Although this is significantly slower than the quality estimator alone, the execution time limitations are one order of magnitude larger. If nothing else, the MVA method could be activated only during the offline reconstruction and not during the processing on the HLT. Since it also does not require large amounts of memory, such an MVA method seems to be a good extension of the VXDTF2 algorithm.

The first point however, is harder to handle. Even though the VXDTF2 already incorporates a Monte Carlo trained component, the SectorMap, this is a slightly different situation. The SectorMap only learns a rough representation of a realistic track and which sectors are befriended with each other. These characteristics are well understood and no significant deviation is to be expected on recorded data. The MVA method, however, provides a more detailed differentiation between signal and background, based on variables that are harder to control. Especially the cluster variables might pose a problem. If the detector in reality turns out to report slightly biased values compared to the simulation, something like a cut on the minimum or maximum charge of clusters of a track might only be reasonable on simulated data. Even worse, the bias might change over time, because of degradation of the detector.

Either these detector simulation specific variables have to be dropped, reducing the improvements to the ones listed in Table 4.2b, or a strategy to calibrate the algorithm on recorded data has to be developed.

The VXDTF2 could be used with an MVA method that does not include the cluster features during a data taking period with low luminosity. In this scenario the background rate is strongly reduced and the produced track candidates have a very high probability to be correct. These track candidates could be used as signal track candidates for the training of the MVA method. The discarded candidates that are not subsets of the final candidates, either include some background SpacePoints or merge different trajectories together. Those track candidates can be used as background for the training. If the amount of background candidates gathered in this way is too low, time periods without an Υ(4S) event can be analysed to gain more.

Such a training sample is not as clean as one produced with Monte Carlo information, but can still be good enough to improve the quality estimation. However, the only useful validation of the trained MVA method would be an extrapolation from the CDC. But this cannot cover low momentum tracks. Therefore, this approach should only be applied if a dedicated high momentum pass of the VXDTF2 is considered.

Another approach could be to use the VXDTF2 without the cluster variables, but analyse the recorded values for them. Differences between this data and the one obtained from Monte Carlo can then be corrected before the MVA method is applied. In this way, a general bias
in the values of the variables can be corrected. If this approach is repeated periodically it has the additional advantage, to be able to detect degradation of the detector and take it into account.

Independent of the chosen approach, it may be problematic to construct the output in such a way that it represents a probability. If the signal to background ratio of the training data is not the same as the evaluation one, this transformation does not hold true anymore. Any part of the VXDTF2 that strongly depends on the output distribution of the quality estimation, the Hopfield algorithm for example, might perform worse because of that.

In conclusion, the simple MVA method studied in this thesis improves the quality estimation with both cluster variables included or not and does not significantly increase the execution time of the VXDTF2. If the cluster variables are not included, the Monte Carlo bias should be negligible and approaches similar to the ones discussed above could also make the inclusion of the cluster variables feasible.
5. Overview and conclusion

In this section an overview over the current state of SVD standalone track finding with the VXDTF2 is given, both, in terms of execution time and in terms of the figures of merit defined in Section 2.4. The VXDTF2 is compared against its predecessor, the VXDTF, and brought into context with the CDC track finding algorithm. Subsequently, a discussion of possible further improvements as well as a conclusion of this thesis is given.

5.1 Performance overview of the VXDTF2

All results presented for the VXDTF2 in this section are based on the same configuration and the Y4S+BKG sample, described on page 27. The candidates are created based on the MuonMap as described on page 31 and the improved candidate selection, discussed in Section 3.4.6, based on the TripletFit, introduced in Section 3.3.3, was applied. In case of the columns denoted with MVA, additionally the MVA method discussed in Section 4 is used. The latter setup also includes a cut on the quality indicator, as it was discussed in Section 4.4.

In order to determine the execution time of the individual stages of the VXDTF2, the algorithm was run on an idling machine equipped with an Intel® Core™ i7-6700 @ 3.40 GHz. The framework records the execution time of each module per event and calculates the mean execution time for each. For the Y4S+BKG sample the complete VXDTF2 took on average 14.5 ms per event, which is twice as fast as the VXDTF and safely within the targeted execution time in the order of magnitude of 10 ms. Even though the used machine is different from the target machine defining the execution time limitations, the HLT, this should be in agreement.

The execution time can be further broken down as shown in the column denoted VXDTF2 in Table 5.1. This shows, that the most time is spent during the network creation, the stage that is responsible for prefiltering the possible combinations of SpacePoints. Therefore, improvements in this stage have the biggest potential to further reduce the execution time, if necessary. The comparison between the columns VXDTF2 and MVA provides an indication of the reproducibility of the measurement: the MVA method influences only the quality estimation and indirectly the candidate selection. The values obtained for the Candidate Creation should be identical, since nothing was changed in this regard. Without a more sophisticated setup, those fluctuations cannot be prevented.
Table 5.1: Average execution time per event on the Y48+BKG sample
Rough estimate on an idling machine equipped with an Intel® Core™ i7-6700 @ 3.40 GHz based on a test run.

<table>
<thead>
<tr>
<th></th>
<th>VXDTF2</th>
<th>MVA</th>
<th>VXDTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>ms</td>
<td>ms</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>Candidate Creation</td>
<td>Sec 3.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpacePoint Creation</td>
<td>1.7</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>Network Creation</td>
<td>8.2</td>
<td>8.1</td>
<td></td>
</tr>
<tr>
<td>Path Finding</td>
<td>2.2</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>Quality Estimation</td>
<td>Sec 3.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>4.8</td>
<td></td>
</tr>
<tr>
<td>Candidate Selection</td>
<td>Sec 3.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.8</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>14.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32.7</td>
</tr>
</tbody>
</table>

Table 5.2: VXDTF2 figures of merit

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>MVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.40 ± 0.02</td>
<td>0.39 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>20.03 ± 0.12</td>
<td>10.06 ± 0.09</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>94.01 ± 0.08</td>
<td>94.80 ± 0.07</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>95.36 ± 0.04</td>
<td>90.09 ± 0.05</td>
</tr>
<tr>
<td>Hit purity</td>
<td>98.95 ± 0.01</td>
<td>99.44 ± 0.01</td>
</tr>
</tbody>
</table>

The current results, in terms of the figures of merit defined in Section 2.4, are presented in the column denoted Default of Table 5.2. The SVD standalone track finding with the VXDTF2 features a high finding efficiency with both hit efficiency and hit purity being over 95%. While there are almost no clones produced, the fake rate is quite high. However, with an additional FastBDT in the quality estimation, this can be improved drastically, as the column MVA shows. The fake rate can be halved, while the remaining quantities are improved even further. The only drawback is an hit efficiency loss of 5%, but it should be possible to prevent most of it based on a modified training goal, as discussed in Section 4.4.

Also, it has to be taken into account that the current amount of background clusters is considered to be an overestimation by a factor up to 5, because cluster timing information is still missing in the BASF2 framework, as it was explained in the paragraph Evaluation samples of Section 3.2.5.

Finally, one can imagine that the fake rate can be reduced further, after the combination of the VXD and CDC track finding and based on the results of the DAF (described in Section 3.3). But even then, fake tracks cannot be ruled out, because it is possible that the background effects produce a charged particle originating from the interaction point. Such a particle cannot be differentiated from a signal particle.

Even without further improvements the VXDTF2 performs significantly better than the alternatives:

As stated in Section 2.3, the alternative to VXD standalone track finding would be an ex-
5.1 Performance overview of the VXDTF2

These figures cannot be compared against each other:

In Figure (a), the finding efficiency based on all track finding detector systems is evaluated once with and once without the VXDTF2. The case without represents a scenario, where the VXD detector is only utilised in the context of an extrapolation from the CDC.

In Figure (b), the finding efficiency of the VXDTF2 is compared to its predecessor VXDTF based on SVD standalone track finding.

However, the combined setup requires merging of the tracks reconstructed by the individual algorithms. Currently this is done with a simple distance measurement between the points at which the track candidates pass through the inner CDC border. If this merging fails to merge two tracks corresponding to a single particle, one of the tracks is counted as clone. If two tracks are merged, although they do not describe the same particle, they either reduce the hit efficiency and purity or count as fake. More sophisticated algorithms for this merging procedure are currently in development.

Since the implemented merging is not perfect, only the influence on the finding efficiency is presented, because it is at worst an underestimation. Figure 5.1a shows the large influence on the finding efficiency of primary particles, if the SVD standalone track finding is added to CDC track finding. The VXDTF2 not only provides additional SVD hits, but also successfully reconstructs tracks that could not be reconstructed with CDC track finding alone. This is not only true for tracks with low momenta, but also for tracks above 1 GeV, which might be due to tracks passing only a short section of the CDC at its outer boundaries. Such a short track can be hard to reconstruct, based on the CDC information alone.

While an improvement was also possible with the VXDTF, Table 5.3 shows that the VXDTF2 also features a significant advantage over its predecessor. The finding efficiency was increased by almost 10% with only a minor increase in fake rate. Moreover, column VXDTF2 (QI > 0.8) proves that the efficiency gain is not a strict trade-off with the slight increase in
Table 5.3: VXDT versus VXDTF2
The hit efficiency in the last column could not be calculated, due to way the cut is applied.

<table>
<thead>
<tr>
<th></th>
<th>VXDTF1</th>
<th>VXDTF2</th>
<th>VXDTF2 (QI &gt; 0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.41 ± 0.02</td>
<td>0.40 ± 0.02</td>
<td>0.38 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>18.07 ± 0.12</td>
<td>20.03 ± 0.12</td>
<td>17.53 ± 0.11</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>84.04 ± 0.12</td>
<td>94.01 ± 0.08</td>
<td>93.33 ± 0.08</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>88.26 ± 0.06</td>
<td>95.36 ± 0.04</td>
<td>-</td>
</tr>
<tr>
<td>Hit purity</td>
<td>95.94 ± 0.03</td>
<td>98.95 ± 0.01</td>
<td>99.00 ± 0.01</td>
</tr>
</tbody>
</table>

Table 5.4: Secondary particles
In this table the VXDTF2 is once evaluated with secondary particles included and once without.

<table>
<thead>
<tr>
<th></th>
<th>Primaries+Secondaries</th>
<th>Primaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone rate</td>
<td>0.50 ± 0.02</td>
<td>0.40 ± 0.02</td>
</tr>
<tr>
<td>Fake rate</td>
<td>17.09 ± 0.11</td>
<td>20.03 ± 0.12</td>
</tr>
<tr>
<td>Finding efficiency</td>
<td>90.17 ± 0.09</td>
<td>94.01 ± 0.08</td>
</tr>
<tr>
<td>Hit efficiency</td>
<td>94.98 ± 0.04</td>
<td>95.36 ± 0.04</td>
</tr>
<tr>
<td>Hit purity</td>
<td>98.85 ± 0.01</td>
<td>98.95 ± 0.01</td>
</tr>
</tbody>
</table>

fake rate: with a reasonable cut on the quality index (QI), the fake rate is lower than the one produced by the VXDTF, while still most of the finding efficiency gain can be retained. Additional to those benefits, also the hit efficiency and purity of the track candidates are significantly higher compared to the first version. Figure 5.1b shows the finding efficiency of the VXDTF compared to the VXDTF2; it is not a specific region that was improved by the VXDTF2, but the whole transverse momentum spectrum.

Finally, it is stated in Section 2.4 that the development of the VXDTF2 is focused on the reconstruction of primary particles. While this is true, it may still happen that trajectories of secondary particles are reconstructed. But since these particles are not considered trackable in the definition described in Section 2.4, such a track candidate is counted as fake.

The column Primaries+Secondaries in Table 5.4 shows the results of the VXDTF2, if secondary particles are also considered as trackable. On the one hand 3% of the fake rate are now counted as fake, due to the scenario described above. On the other hand, not all secondary particles are found, and therefore the finding efficiency is reduced by 4%. From these differences it can be estimated that about 43% of the secondary particles have been successfully reconstructed; the actual result is presented in Figure 5.2. Overall, the influence is relatively small because of the low amount of secondaries compared to primaries. This is at least partially caused by the trackable requirement to produce enough hits to have at least five degrees of freedom: Secondaries produced in layers three and four of the SVD simply have a low probability to pass this requirement in the SVD alone.
5.2 Outlook

There are still some topics that need to be addressed in the context of the further \vXDTF2 development. Various suggestions where to start are listed in this section.

Figure 5.3 compares the finding efficiency achieved in the candidate creation with the final finding efficiency of the complete \vXDTF2. Only a small part of the finding efficiency is lost during the candidate selection; for most of the missing tracks no candidates were created at all. Therefore, the biggest influence in terms of finding efficiency can be achieved by further studies on the SectorMap.

As stated in Section 3.2, there are a number of open topics in the context of its training procedure and evaluation:

Training amount The amount of required training data is hard to choose correctly, because the training of the SectorMap does not seem to converge. Even if it converges at some point, there would be so many fake candidates allowed that the SectorMap does not fulfil its purpose anymore. It is currently under investigation if this can be solved by rejecting trajectories of particles that suffered from strong multiple scattering. But up to now, it is unclear if this is sufficient; probably further studies will be required.

Filter selection It was not evaluated which kind of filters are effective or whether some of them are redundant. As an example it might even be beneficial to skip the 2-SpacePoint filters completely and only rely on friendship relations and 3-SpacePoint filters.
Figure 5.3: Finding efficiency discrepancies
In this figure, three different configurations of the VXDTF2 are evaluated. One without any candidate selection, one with default candidate selection and one with the MVA quality estimator.

Training bias
It is unclear, what kind of bias is introduced by the chosen training sample, because it could not be verified thoroughly on recorded data up to now. During first studies on a test beam one example for such a bias was discovered: if the interaction point in the training sample is not varied, the SectorMap loses a lot of its efficiency for a beam setup featuring a different interaction point. In such a scenario the SectorMap would have to be retrained frequently.

Memory access
The memory management of the network creation is poorly optimised. As shown in Table 5.1, this part of the VXDTF2 is responsible for over 55% of the total execution time. A first analysis showed, that most of the time of the network creation is spent in memory operations instead of the different filters of the SectorMap. Only the slowest parts of the filters are relevant at all and those are trigonometric functions, which could be faster and still accurate enough even if reference tables or other approximations are utilised.

These topics influence the maximum finding efficiency that can be achieved by the VXDTF2 and how many fake track candidates are created at all. In order to reduce the fake rate, the quality estimators have to be improved, since the SectorMap cannot prevent the creation of those fake track candidates, as it was discussed in Section 3.2. This thesis covered only a first look into MVA methods and there is room for more improvements:

1. As already discussed in Section 4.2, the chosen training goal is suboptimal.
2. A small neuronal network could be tested as an alternative to the FastBDT that was
used in this thesis.

3. The input features have not been studied extensively and some, such as the timing information of the SVD clusters, are not implemented at all.

4. A large fraction of the execution time of the MVA method is spent to prepare the input features. Therefore, better handling of the variables should improve the execution time.

However, before doing so, a decision has to be made on how to ensure that this method will also perform well on data recorded by the experiment.

Furthermore, even though the new implementation is significantly faster than the old one, some of the algorithms used in the VXDTF2 are still implemented in a suboptimal way in terms of execution time and memory requirements. Since there are no major changes of the logic of the VXDTF2 expected, it seems to be a good time to perform a dedicated study to identify bottlenecks of the algorithm.

Finally, once these topics are covered, special treatment of curling and secondary particles could be added.

5.3 Conclusion

In the context of this thesis, the development of the VXDTF2, that was started as a part of the thesis covering the VXDTF [3], was continued. It is now in a stable working condition for the first time and will soon replace its predecessor in the reconstruction chain of the BASF2 framework, because it outperforms the VXDTF both in terms of the execution time and in track finding performance:

As it was shown in the performance overview, Section 5.1, the new version of the algorithm is twice as fast as the previous one and improves the finding efficiency from 84\% to 94\%, reducing almost two thirds of the inefficiency. Not only is this achieved without an increase in either fake or clone rate, but the VXDTF2 also features a significantly higher hit efficiency and hit purity.

Furthermore, a feasibility study on improving the quality estimation with MVA methods was performed, in order to prepare the way for further improvements. It was shown that especially the fake rate can benefit from that, but also the hit purity and the finding efficiency. Finally, sections of the algorithm that should be improved further were identified.
Bibliography


Danksagung

Zunächst möchte ich mich bei PD Andreas Meyer und Professor Günter Quast bedanken, dass sie die Betreuung und Bewertung dieser Arbeit übernommen haben. Besonderer Dank geht an Dr. Martin Heck, der diese Arbeit ermöglicht hat und aufgrund seiner sowohl physikalischen als auch technischen Expertise konstruktive Kommentare zu angestrebten Lösungen liefern konnte.


Schlussendlich geht mein Dank auch an meine Familie, die mich mein gesamtes Studium über unterstützt hat und an meine Freundin, die mich in stressigen Zeiten vor zusätzlicher Arbeit bewahrt hat.
Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, 25.09.2017

.........................................

(Jonas Wagner)