Master's Thesis

Upgrade of Belle II's Neural Network Trigger by Track Finding in Three-Dimensional Hough Space

Submitted by
Simon Hiesl

Supervised by
Prof. Dr. Christian Kiesling

Ludwig-Maximilians-University Munich
Faculty of Physics

Munich, 12th of April 2024

Masterarbeit

Upgrade des neuronalen Netzwerk Triggers von Belle II durch Spurfindung im dreidimensionalen Hough-Raum

Vorgelegt von
Simon Hiesl

unter Betreuung von
Prof. Dr. Christian Kiesling

Ludwig-Maximilians-Universität München
Fakultät für Physik

München, den 12.04.2024

Abstract

The target luminosity of the Belle II experiment, located at the SuperKEKB electron-positron collider in Tsukuba, Japan, is $6 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ [1]. For this world-record luminosity, high levels of beam background are expected. Thus, an efficient and robust trigger system at the hardware level, capable of selecting annihilation events from the interaction point ("$z = 0$"), is indispensable. For this purpose, a novel track trigger using the wire information of the central drift chamber predicts the z-vertex utilizing a neural network [2]. While this trigger is very successful in suppressing background, recent high luminosity physics runs produced many fake tracks, increasing the trigger rate close to the maximum limit [3]. To keep the L1 trigger rate below the design value of 30 kHz, an upgrade of the Neuro Trigger is proposed in this work.

In this thesis, the preselection algorithm for the neural network, called the 3DFinder [4], which creates three-dimensional track candidates using a three-dimensional Hough transformation, is extensively analyzed and upgraded. For the analysis, both simulated Monte Carlo data and real data from the latest Belle II runs are utilized. The upgrade includes a new clustering algorithm and new parameters to reduce the number of fake tracks. The proposed new algorithms for the 3DFinder allow for an implementation on the trigger hardware while achieving high efficiency for single IP tracks and a low fake rate.

Contents

Chapter 1

Introduction

The Standard Model (SM) of particle physics is by far the most successful theory, describing the interaction of fundamental particles with remarkable precision. However, the SM is incomplete as it fails to provide answers to many fundamental questions. For example, the observed matter-antimatter asymmetry in our universe is not explained. The CP-violation of the SM is orders of magnitude too small to account for this asymmetry [1]. The motivation of the Belle II experiment at the SuperKEKB asymmetric-energy electron-positron collider in Tsukuba, Japan, is to challenge the SM and discover New Physics (NP) in order to provide answers to many fundamental questions of nature [5, 1].

The SuperKEKB collider, which was upgraded from the KEKB collider, provides collisions of e^+e^- pairs at the $\Upsilon(4S)$ resonance. With a target luminosity of $6 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, the number of physics processes to be recorded at the Belle II detector is larger by a factor of 30 compared to the predecessor Belle [6]. While this allows for large statistics for physics events, the world record luminosity also introduces many challenges to data acquisition. As it is neither possible nor interesting to record every event, a trigger system is necessary for the Belle II detector. Due to the high luminosity, very high backgrounds are anticipated. This predominately includes machine background leading to displaced tracks along the beam ("z") axis [7]. Hence, the trigger system needs to make a decision about whether the observed event originates from the intended interaction point (IP) of the detector. As the Belle II detector can buffer the incoming detector data for at most 5 µs, the available decision time for event recording is very limited. For this purpose, a pipe-lined deadtime-free hardware trigger, the "Level 1" (L1) trigger, is used [8]. The sense wire hits in the Central Drift Chamber (CDC) are sampled in so-called track segments, which are used to determine track candidates. Currently, a two-dimensional Hough transformation is executed to find two-dimensional track candidates [9]. However, this allows for track candidates to be displaced along the whole beam pipe, as no z-information is available. To access three-dimensional information, stereo track segments are selected for each candidate. Utilizing the combined information, a neural network is used in the Neuro Trigger module to predict the z-vertex origin and the polar emission angle θ for each track candidate [3]. While this Neuro Trigger is very successful at reducing background, two big problems are currently observed. With an incorrect vertex prediction of displaced tracks, the trigger rate increases with background tracks. Furthermore, fake tracks resulting from random background track segments in the CDC are increasing the trigger rate even further [3]. As these problems will get increasingly worse when approaching the luminosity target, the track trigger has to be upgraded. For this purpose, a three-dimensional Hough transformation was proposed [4]. Using this algorithm, three-dimensional track candidates can be created by utilizing the complete CDC information, which includes the stereo track segments. This method automatically suppresses displaced tracks as an IP assumption is built into this track model.

However, multiple problems in the original implementation of the algorithm do not allow it to be used on the L1 trigger.

The goal of this thesis is to propose a new algorithm in order to deal with the expected backgrounds, make it suitable for the L1 trigger, and make an implementation on field-programmable gate array (FPGA) boards possible. In Chap. 2, a short theoretical motivation for the Belle II experiment is given. This provides insight into the types of events for which the trigger system must be efficient. In Chap. 3, the SuperKEKB collider and the Belle II detector are described. Especially the CDC, the only subdetector used for the track trigger, is explained in detail. In Chap. 4, the current trigger system of Belle II is introduced. This includes the expected event types, the track segment finder, the two-dimensional Hough track finder (2DFinder), and the Neuro Trigger. The problems of the current trigger will be explained in detail, motivating the three-dimensional Hough track finder (3DFinder). In Chap. 5, the original 3DFinder introduced in [4] is thoroughly analyzed. This includes an explanation of the algorithm in the first section and Monte Carlo studies of simulated data with and without background. In Chap. 6, the Hough space is analyzed. For this purpose, a detailed statistical cluster analysis is conducted. With this information, the average clusters of simulated tracks can be determined, motivating a new clustering algorithm. Furthermore, the difference between real clusters and fake clusters can be analyzed. Chap. 7 introduces a new clustering algorithm suitable for the hardware implementation, utilizing the results of the previous chapter. This new algorithm is analyzed and updated to suppress fake tracks by introducing new cut parameters. In Chap. 8, this new algorithm is tested on real data, revealing, however, an efficiency problem. This problem is solved by introducing a new hit-to-cluster association algorithm for the 3DFinder. Moreover, a cut on the ADC count of the sense wires and a newly trained neural network are investigated. Finally, Chap. 9 provides a summary of the achievements reached thus far along with an outlook towards new developments.

Chapter 2

Theoretical Motivation

The Standard Model (SM) of particle physics is one of the most successful theories in physics. It describes the interactions among elementary particles, mediated by the electromagnetic, strong, and weak forces, with remarkable precision. However, despite being the best-tested theory of nature, it fails to provide answers to many fundamental questions [1]. This includes the matter-antimatter asymmetry observed in our universe, which cannot be explained with the SM. Other problems are, for example, the observation of neutrino oscillations, the hierarchy problem in fermion masses, or the nature of dark matter. Furthermore, some theoretical predictions of the SM are in tension with the current experimental results. For example, the magnetic moment of the muon $(g-2)$ has a tension above 3σ [4].

With the Belle II experiment, exploration of New Physics (NP) beyond the SM is possible through measurements of flavor physics reactions at the precision frontier [5]. As an example of such measurements, the following section describes the CP-violation predicted by the SM for quark mixing and is based on [10].

2.1 The CKM Matrix

As matter is considerably more prevalent in our universe than antimatter, CP-violating interactions must exist [11]. If there were no CP-violation, for every process $i \to f$, the process $\bar{i} \to \bar{f}$ should occur with the same probability, leading to the annihilation of matter and antimatter. In the Lagrangian field theory of the SM, all couplings must be real for an unbroken CP-symmetry. However, the Belle experiment provided conclusive proof that the couplings in the quark mixing must be complex [12]. No choice of phase is possible to make these couplings real.

In the SM Lagrangian, the charged-current interaction term between the quarks is given as

$$\mathscr{L}_{cc} = \frac{g}{\sqrt{2}} \begin{pmatrix} \bar{u}_L & \bar{c}_L & \bar{t}_L \end{pmatrix} V_{\mathrm{CKM}} \gamma^\mu \begin{pmatrix} d_L \\ s_L \\ b_L \end{pmatrix} W_\mu^+ + h.c.\,, \tag{2.1}$$

where g is the coupling constant, $(u_L, d_L, \dots)$ the left-handed quark fields, W_μ the weak W-boson, and V_{CKM} the Cabibbo-Kobayashi-Maskawa[1] (CKM) matrix. The couplings of

[1] This was formulated by Kobayashi and Maskawa in 1973 (see [13]).

the quark mixing transitions are the matrix elements V_{ij} of the CKM matrix, defined as

$$V_{\text{CKM}} = \begin{pmatrix} V_{ud} & V_{us} & V_{ub} \\ V_{cd} & V_{cs} & V_{cb} \\ V_{td} & V_{ts} & V_{tb} \end{pmatrix} . \tag{2.2}$$

As this matrix is unitary, i.e., $V_{\text{CKM}} \cdot V_{\text{CKM}}^{\dagger} = \mathbb{1}$, and considering the freedom to redefine the phase of the quark fields, only four independent parameters exist. Hence, the CKM can be represented with the Euler angles $(\theta_{12}, \theta_{23}, \theta_{13})$ as

$$V_{\text{CKM}} = \begin{pmatrix} c_{12}c_{13} & s_{12}c_{13} & s_{13}e^{-i\delta} \\ -s_{12}c_{23} - c_{12}s_{23}s_{13}e^{i\delta} & c_{12}c_{23} - s_{12}s_{23}s_{13}e^{i\delta} & s_{23}c_{13} \\ s_{12}s_{23} - c_{12}c_{23}s_{13}e^{i\delta} & -c_{12}s_{23} - s_{12}c_{23}s_{13}e^{i\delta} & c_{23}c_{13} \end{pmatrix} , \tag{2.3}$$

where $s_{ij} = \sin(\theta_{ij})$, $c_{ij} = \cos(\theta_{ij})$, and δ is the complex phase responsible for CP-violation. As $\theta_{12} \gg \theta_{23} \gg \theta_{13}$, one can express the CKM matrix with the four Wolfenstein parameters (λ, A, ρ, η):

$$\lambda = s_{12}, \quad A = \frac{s_{23}}{\lambda^2}, \quad A\lambda^3(\rho - i\eta) = s_{13}e^{-i\delta} . \tag{2.4}$$

Hence, the CKM matrix can be expressed up to order λ^3 as

$$V_{\text{CKM}} = \begin{pmatrix} 1 - \lambda^2/2 & \lambda & A\lambda^3(\rho - i\eta) \\ -\lambda & 1 - \lambda^2/2 & A\lambda^2 \\ A\lambda^3(1 - \rho - i\eta) & -A\lambda^2 & 1 \end{pmatrix} + \mathcal{O}(\lambda^4) . \tag{2.5}$$

This is an expansion in $\lambda = |V_{us}|$, where unitarity is satisfied up to order λ^4. Using the unitarity condition, the relation

$$V_{ud}V_{ub}^* + V_{cd}V_{cb}^* + V_{td}V_{tb}^* = 0 \tag{2.6}$$

$$\implies -1 + \frac{V_{ud}V_{ub}^*}{-V_{cd}V_{cb}^*} + \frac{V_{td}V_{tb}^*}{-V_{cd}V_{cb}^*} = 0 \tag{2.7}$$

can be represented as a triangle in the complex plane, called the unitary triangle. The apex of this triangle can be defined by the Wolfenstein parameters as (ρ, η). This triangle is displayed in Fig. 2.1. Note that there are a total of six independent triangles, where each triangle has the same area. The three angles in Fig. 2.1 are defined as

$$\phi_1 \equiv \arg\left(\frac{V_{cd}V_{cb}^*}{-V_{td}V_{tb}^*}\right), \quad \phi_2 \equiv \arg\left(\frac{V_{td}V_{tb}^*}{-V_{ud}V_{ub}^*}\right), \quad \phi_3 \equiv \arg\left(\frac{V_{ud}V_{ub}^*}{-V_{cd}V_{cb}^*}\right) . \tag{2.8}$$

When this triangle is non-trivial, i.e., no angle is 0° or 180°, the triangle must be in the complex plane. Hence, the couplings must be complex, resulting in a non-zero phase δ and therefore in a CP-violation. The amount of CP-violation is proportional to the area of the triangle.

To measure the couplings, the decay of bottom quarks can be investigated. For this purpose, the $\Upsilon(4S)$ resonance is used at Belle II. With $m_{\Upsilon(4S)} = 10.58\,\text{GeV}/c^2$ the $\Upsilon(4S)$ meson is kinematically allowed to decay into two B mesons[2], i.e., $B^0\bar{B}^0$ or B^+B^-. Studying such B meson decays in the Belle experiment, the unitary triangle could be measured

[2] $B^0 = d\bar{b}$, $\bar{B}^0 = \bar{d}b$, $B^+ = u\bar{b}$, and $B^- = \bar{u}b$.

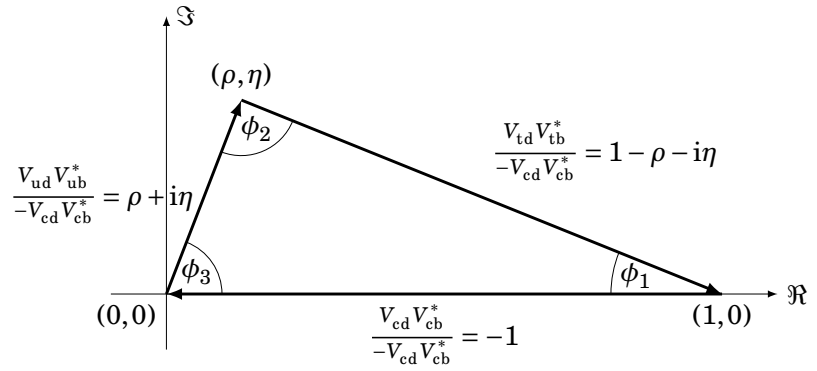


Figure 2.1: The unitary triangle in the complex plane defined by the quark mixing couplings [9].

through quark mixing, verifying the CKM matrix theory and therefore the SM [12]. However, the CP-violation observed from the CKM matrix is many orders of magnitude too small to explain the matter-antimatter asymmetry [1]. With the Belle II experiment, more precise measurements of this triangle to possibly find deviations from the SM are planned. For this purpose, considerably larger statistics and a more precise reconstruction are necessary. Any discrepancy of a perfect triangle must result in non-unitarity of the CKM matrix and therefore NP, which may describe a new source of CP-violation. Moreover, the diagonal hierarchy of the CKM matrix is not required by the SM and may indicate a flavor symmetry at higher energy scales [1].

On average, there are 10 tracks in a $\Upsilon(4S)$ event with an average transverse momentum of $p_\mathrm{T} = 500\,\mathrm{MeV}/c$ [4]. It is very important to record such high-multiplicity events at Belle II with high efficiency at the trigger system.

2.2 New Physics at Belle II

Not only the CP-violation of the CKM matrix can be investigated at Belle II. In the following, additional planned investigations of NP are listed, taken from [4] and [1]: Note that this is not an exhaustive list.

- Invisible B decays: One B meson, e.g., $B^0 \to \nu\nu$, could decay into an invisible final state. A reconstruction of the other B meson can make a full reconstruction possible, as the center of mass energy is known.
- τ physics: As the cross section for τ pair production at Belle II is comparable to that of B pair production, NP could be observed in τ decays. For example, lepton number violating decays like $\tau \to lll$ or $\tau \to \mu\gamma$ would clearly violate the SM.
- Dark photons: The production of $e^+e^- \to \gamma A'$, where A' is a dark photon that couples to the electric charge e, could be detected by missing energy or by a displaced vertex.
- Muon $g-2$: Currently, the experimental value of the magnetic moment of the muon has a tension above 3σ with the theoretical prediction. To investigate this discrepancy, the theoretical error must be minimized by experimentally measuring the hadronic vacuum polarization. This can be conducted at Belle II, as the initial

state radiation $e^+e^- \to \gamma e^+e^- \to \gamma q\bar{q}$ produces final states with few hadrons, which dominate the uncertainty in the theoretical calculations.

- New CP-violating phases in the quark sector: As mentioned in Sec. 2.1, the CP-violation of the CKM matrix is not sufficient to explain the matter-antimatter asymmetry in the universe. New sources may be found by studying, for example, time-dependent CP-violation in penguin transitions.
- Multiple Higgs Bosons: Extensions to the SM predict charged Higgs bosons that could be observed in τ lepton flavor transitions [1].

Especially the first three points cause events with a low track multiplicity [4]. Thus, the Belle II trigger system must not only efficiently handle the high multiplicity of B-pair productions as mentioned in Sec. 2.1. It is also very important to have high efficiency for triggering low multiplicity events with minimal track counts.

Chapter 3

SuperKEKB and the Belle II Detector

In order to challenge the Standard Model, large statistics of physics events are necessary. The number of physics events to be observed at Belle II,

$$N = \sigma \int L(t)\,\mathrm{d}t\,, \tag{3.1}$$

is the cross section σ of the physics interactions multiplied with the integrated luminosity [4]. The luminosity L, which is the interaction rate per unit cross section for colliding particles, has to be increased to reach the targeted event number N in a reasonable timeframe [6]. In this context, the objective is to reach a luminosity of $6 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, meaning the peak luminosity of the KEKB particle collider has to be multiplied by a factor of 30 [6]. To achieve this, the KEKB collider was upgraded to the SuperKEKB collider.

3.1 SuperKEKB

The SuperKEKB collider is reusing the same tunnel as KEKB [5]. SuperKEKB consists of an injection linear accelerator (LINAC) used to accelerate the electron (e^-) and the positron (e^+) beams. Those two beams are fed into the storage tunnel consisting of two rings [6]. As shown in Fig. 3.1, one of the two storage rings is called the low-energy ring (LER), where positrons with an energy of 4.0 GeV are stored. The second ring, called the high-energy ring (HER), is used to store electrons at 7.0 GeV [5].

In each storage ring, 2500 bunches of either electrons (HER) or positrons (LER) are traveling at nearly the speed of light [14]. At a certain crossing angle, both beams are brought together for collision, which happens at the so-called interaction point (IP). A Cartesian coordinate system is introduced, where the nominal collision point is defined as $(x, y, z) = (0, 0, 0)$. Therefore, a center-of-mass energy of 11 GeV is achieved, which matches the boosted $\Upsilon(4S)$ resonance at 10.58 GeV. The asymmetric beam energies cause a forward boost in the center-of-mass frame, which defines the positive z-direction of the coordinate system [5]. Note that the detector around the IP (called Belle II) is built asymmetrically in order to account for the more likely forward momentum of the produced particles. Since both rings have a circumference of approximately 3 km [5], a bunch crossing frequency of about 250 MHz is expected. Consequently, a bunch crossing happens every 4 ns.

The luminosity is given by [9]

$$L = \frac{N_+ N_- f_c}{4\pi\sigma_x\sigma_y} \cdot R_L\,, \tag{3.2}$$

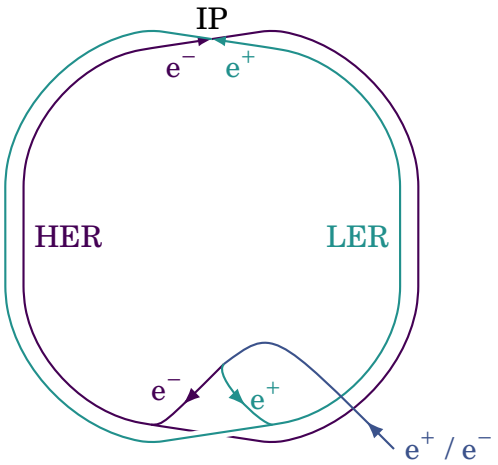


Figure 3.1: A schematic view of the SuperKEKB collider, consisting of a low-energy ring (LER) for positrons and a high-energy ring (HER) for electrons [9]. The particles are accelerated and injected by a linear accelerator (LINAC).

where $\sigma_{x/y}$ describes the Gaussian extensions of the beams, $N_{+/-}$ the number of particles in each beam, f_c the crossing frequency of the bunches, and R_L the reduction factor [9]. The horizontal and vertical Gaussian profiles are proportional to the beta function $\sigma_{x/y} \propto \sqrt{\beta_{x/y}}$, a function describing the transverse size of the beam [9, 4]. Hence, the luminosity can be increased by focusing the beams shortly before the IP with quadrupole magnets [4]. This squeezing of the vertical beta function, the so-called "Nano-Beam" scheme, is used at SuperKEKB with the intention of achieving a twentyfold extension reduction at the IP [5, 6]. To further enhance the luminosity, the beam currents are twice as large as those of KEKB [14], increasing $N_{+/-}$ and f_c.

3.2 The Belle II Detector

The Belle II detector is an upgrade of the Belle general purpose (4π) particle detector [5]. Its outline is displayed in Fig. 3.2 [15] where the most important subsystems for particle identification are labeled.

Starting from the interaction point (IP), the first detector system is called the vertex detector (VXD). This detector consists of the innermost pixel detector (PXD), a small semiconductor detector based on the DEPFET (depleted field effect transistor) technology [5]. With its 8×10^6 pixels, it can provide a very precise vertex position measurement of charged particles. This detector is also useful for the identification of low-momentum particles, like slow pions [16]. Surrounding the PXD, a silicon vertex detector (SVD) is used with an inner radius of 38 mm and an outer radius of 140 mm. The SVD consists of four layers of double-sided silicon strip detectors and, like the PXD, has a low material budget [5]. In combination with the PXD, the SVD is important to reconstruct K^0_S meson decays that happen inside the VXD [1].

The VXD is surrounded by Belle II's main tracking device, the central drift chamber (CDC). Since this detector is the only one used in this thesis, the CDC is dealt with in detail in Sec. 3.3.

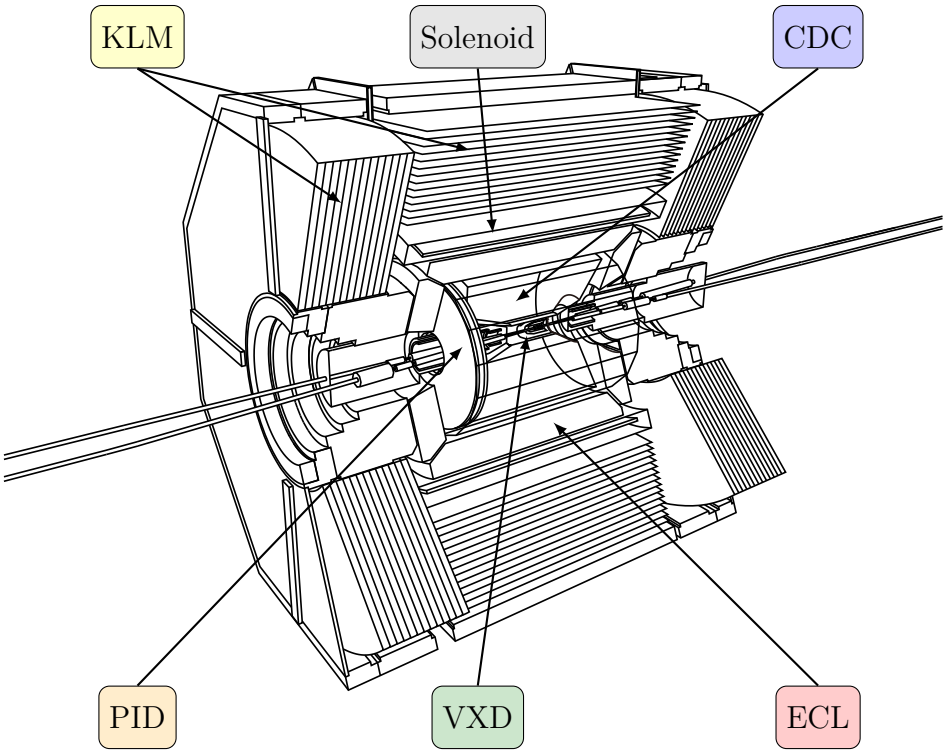


Figure 3.2: The Belle II detector outline illustrating the main detector subsystems [15].

The CDC is surrounded by a particle identification system (PID), consisting of two different types of detectors. In the forward endcap region of the detector, an aerogel ring-imaging Cherenkov detector (ARICH) is used, while the barrel region of the CDC is surrounded by a Time-Of-Propagation (TOP) counter [5]. Both detectors utilize the Cherenkov radiation that a charged particle emits in a medium when its velocity exceeds that of light traveling through that medium. Utilizing the Cherenkov angle of the emitted photons, the velocity of such a particle can be measured precisely. Furthermore, in combination with the momentum of a particle, the velocity can be used to differentiate between particles like kaons and pions [5].

The PID is followed by the electromagnetic calorimeters (ECL). Here scintillating CsI(Tl) crystals are used in which electromagnetic particles leave showers of e^+e^- pairs [5]. This allows for the identification of photons and electrons while providing a precise measurement of their energies. Hence, the ECL can easily identify Bhabha- and $\gamma\gamma$-scattering events [8].

The outermost part of the Belle II detector is a K_L and muon detector (KLM). Alternating iron plates and detector elements are used in order to detect long-lived particles that do not get stopped by the ECL [5].

3.3 The Central Drift Chamber of Belle II

The central drift chamber (CDC) is the main tracking device for charged particles in the Belle II detector. This chamber, with an inner radius of 160 mm and an outer radius of 1130 mm, is filled with a gas mixture of helium and ethane (He-C_2H_6) [5]. Around the

CDC, a super-conduction solenoid is producing a nearly homogeneous magnetic field in the z-direction of approximately 1.5T. Due to the Lorentz force $\vec{F}_{\mathrm{L}}$, charged particles will travel on helical trajectories perpendicular to the magnetic field. Since the material budget of the CDC is very small, the energy loss can be neglected, and hence, the transverse momentum p_{T} of a particle can be directly calculated from the radius $r_{2\mathrm{d}}$ with

$$|\vec{F}_{\mathrm{L}}| = q|\vec{v} \times \vec{B}| = \frac{q}{m} \cdot p_{\mathrm{T}} B \stackrel{!}{=} |\vec{F}_{\mathrm{c}}| = m \cdot \frac{v^2}{r_{2\mathrm{d}}} = \frac{p_{\mathrm{T}}^2}{m r_{2\mathrm{d}}} \implies p_{\mathrm{T}} = q r_{2\mathrm{d}} B \,. \tag{3.3}$$

Therefore, a charged particle can only leave the CDC if its radius is above 1130/2 mm. Using Eq. 3.3 we get the lowest possible transverse momentum for a charged particle to not curl back into the CDC as

$$p_{\mathrm{T}}^{\mathrm{min}} = r_{2\mathrm{d}}^{\mathrm{min}} q B = \frac{1130}{2}\,\mathrm{mm} \cdot e \cdot 1.5\mathrm{T} \approx 254\,\mathrm{MeV}/c \,. \tag{3.4}$$

In order to measure the trajectory of a charged particle, the CDC is divided into drift cells of 2 mm size with two types of wires: sense and field wires. The 42,240 field wires surround the 14,336 sense wires, creating a high-voltage electric field [4]. Charged particles traveling through the CDC will ionize the gas according to the Bethe-Bloch formula. Free electrons, released from the ionization of the gas mixture, will accelerate towards those sense wires, causing new ionization. Due to the strongly increasing electric field close to a sense wire, an avalanche effect takes place, creating a sizeable electric signal at the sense wire. Such a signal is considered a "hit" in the CDC. The pressure of the gas and the voltage are adjusted in such a way that the avalanche is confined to the closest sense wire with an approximately constant drift velocity. Therefore, the drift time t_{d} can be used as a distant measure between the track and the wire.

For the reconstruction of a particle's trajectory, three-dimensional positional information is required. The CDC makes this possible by providing two different spatial orientations of sense wires: axial and stereo wires. Axial wires are completely parallel to the z-axis, going from one end plate of the CDC to the other, while stereo wires are skewed with respect to the z-axis. Each wire layer in the CDC is either composed of axial wires or of stereo wires only. In Fig. 3.3 an axial wire layer and a stereo wire layer are schematically displayed. Multiple axial layers next to each other are combined to form an axial super layer, while multiple stereo layers are combined to form a stereo super layer. In total, there are five axial and four stereo super layers in the CDC. The innermost super layer (SL 0) is an axial super layer containing eight wire layers, while all other super layers (SL 1-8) contain six wire layers each. As can be seen in Fig. 3.4, the axial (gray) and stereo (black) super layers alternate, starting and ending with an axial super layer. Using the combinatorics of the wire hits in the axial and stereo super layers, a precise helical track reconstruction is possible. This yields, most importantly, the charge, momentum, and the vertex of a charged particle.

Since the CDC does not cover the complete space around the IP, not all charged tracks can be reconstructed by the CDC. In Fig. 3.5, the cross-section of the CDC along the beam axis is illustrated. The polar emission angle θ of a particle, where θ starts in the forward direction of the z-axis, determines the reconstruction capabilities of the CDC. The full CDC acceptance range is $\theta \in [35, 123]^\circ$, i.e., all super layers are hit by such a particle

(a) Axial Wires **(b)** Stereo Wires

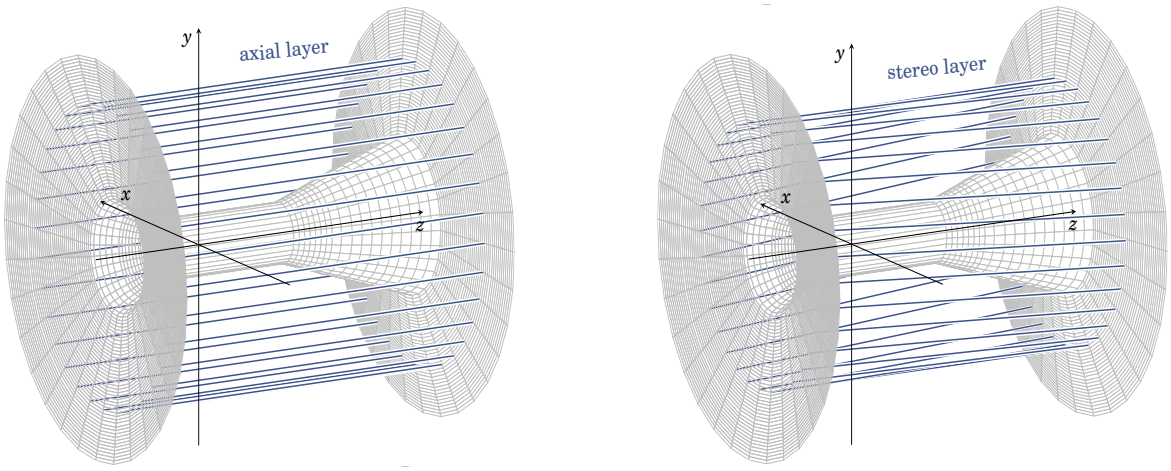


Figure 3.3: A schematic display of the two different types of CDC wire layers [9]. In (a), axial wires are displayed, and in (b), skewed stereo wires.

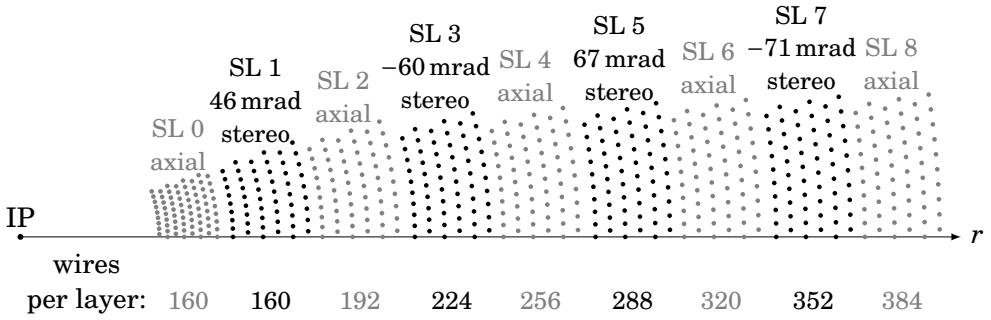


Figure 3.4: The transverse cross-section of the CDC illustrating the arrangement of the wire and super layers [9].

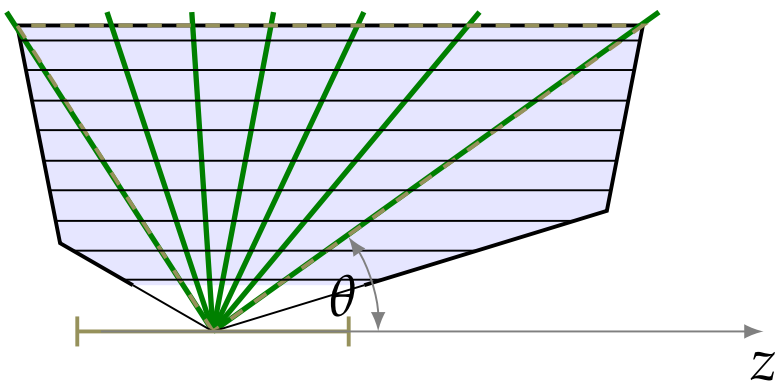


Figure 3.5: The longitudinal cross-section of the CDC illustrating the acceptance range and the super layers [4].

given a sufficient transverse momentum (p_{T}). This is illustrated by the green tracks in Fig. 3.5: The rightmost track has an emission angle of $\theta = 35°$, while the leftmost has an angle of $\theta = 123$. Hence, the shallower the emission angle of a particle is, the less likely it is to be found.

Chapter 4

The Current Trigger

4.1 Expected Event Types

In Tab. 4.1 the expected cross sections and trigger rates at the target luminosity of $6 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ are listed, where the Bhabha and $\gamma\gamma$ processes are already pre-scaled by a factor of 1/100 due to their large cross sections. Of those events, particularly $\Upsilon(4S)$

Table 4.1: The expected cross sections and L1 trigger rates at Belle II [5].

Physics process	**Cross section** [nb]	**Rate** [Hz]
$\Upsilon(4S) \to B\bar{B}$	1.2	960
Hadron production from continuum	2.8	2200
$\mu^+\mu^-$	0.8	640
$\tau^+\tau^-$	0.8	640
Bhabha ($\theta_{\mathrm{lab}} \geq 17°$)	44	$350^{(a)}$
$\gamma\gamma$ ($\theta_{\mathrm{lab}} \geq 17°$)	2.4	$19^{(a)}$
2γ processes ($\theta_{\mathrm{lab}} \geq 17°$, $p_{\mathrm{T}} \geq 0.1\,\mathrm{GeV}/c$)	~ 80	~ 15000
Total	~ 130	~ 20000

$^{(a)}$ The rate is pre-scaled by a factor of 1/100.

$\to B\bar{B}$ and $\tau^+\tau^-$ events are interesting physics events to be recorded with a high efficiency at Belle II (see Chap. 2). While the event types in Tab. 4.1 describe physical processes resulting from electron and positron collisions, events not caused by such a collision are observed in the detector as well. These events are classified as "background". Due to the increase in luminosity, considerably higher background rates are to be expected.

The following types of background are observed at Belle II [7]:

- **Touschek effect**: It is possible for two particles to scatter in the same bunch. Those small changes in the transverse momentum can make the particles deviate from the planned trajectory, causing them to hit the beam pipe after some time. This may lead to a shower of particles entering the active volume of the detector.
- **Beam-gas scattering**: Although there is an ultra-high vacuum in the beam pipe, a beam particle may scatter on left-over "rest-gas" nuclei. Thus, the particle's trajectory is changed, emitting Bremsstrahlung in addition.
- **Synchrotron radiation**: Due to the deflection of the beam particles, photons are emitted, which can ionize the gas in the CDC.

- **Luminosity background** [4]:
  - Bhabha-scattering allows for low momentum transfer of the beam particles, causing a collision with the beam pipe.
  - Radiative Bhabha-scattering mainly emits photons along the z-axis, leading to nuclear spallation products by interacting with the beam pipe or with elements of the magnetic beam guide system.
  - $\gamma\gamma$-scattering events can cause the creation of a low-momentum e^+e^- pair that can hit the inner tracking detectors.
- **Injection background**: Shortly after the injection of beam bunches to keep the total beam currents stable, betatron oscillations are observed, which can cause a loss of particles around the interaction region.
- **Electronic crosstalk**: Although not a physical background, some wires in the CDC may incorrectly get activated by signal leakage from neighboring wire hits.

Considering these types of background and their inevitable increase when the luminosity is raised, more and more particles will enter the Belle II detector from the outside. Hence, a trigger system capable of differentiating between background and physics events is essential.

4.2 The Belle II Trigger

As mentioned in Sec. 3.1, the electron and positron bunches cross each other in the interaction region every 4 ns, which is equivalent to a crossing frequency of 250 MHz. This makes it impossible to read out the complete detector information of Belle II for every bunch crossing due to bandwidth restrictions [3]. Furthermore, a lot of background, uninteresting QED events (like Bhabha or $\gamma\gamma$ processes) and non-annihilation events would be recorded at a significantly higher rate than the interesting physics processes. Hence, a so-called trigger system is necessary at Belle II in order to collect as many interesting events as possible while keeping the trigger rate, i.e., the decision rate for data recording, below 30 kHz. Comparing the interesting physics rates in Tab. 4.1 (e.g., $\Upsilon(4S) \to B\bar{B}$ or $\tau^+\tau^-$) with the crossing frequency of 250 MHz, it becomes evident that a trigger system is indispensable.

The Belle II trigger consists of a "Level 1" (L1) trigger that is implemented completely on hardware [8]. This trigger has to make a trigger decision within a time frame of at most 5 µs since the data acquisition system is not able to buffer the hit information for longer. Here, field-programmable gate array (FPGA) boards are used for a near-dead-time-free processing of the detector data [8].

After the L1 trigger decision, the event data is fully read out and is passed to the High Level Trigger (HLT). This trigger runs on CPUs using the Belle II analysis software framework (basf2) [17] to make a full event reconstruction while further reducing the amount of data to be recorded by disregarding uninteresting events [18]. With an L1 input rate of 30 kHz and an event size of 100 kB, the HLT has to process 3 GB of data every second[1] and reduce

[1] With a crossing frequency of 250 MHz and no trigger system, this would result in a data rate of approximately 25 GB per second.

it to less than 1 GB per second [18].

4.3 The L1 Trigger System

Only four parts of the detector have fast enough readout times to be used for the L1 trigger. This includes the CDC, the ECL, the KLM, and the TOP detectors [8]. While the KLM trigger mainly provides muon hit patterns and the TOP trigger provides timing information for an event, the CDC trigger and ECL trigger are the most important. The ECL trigger is used as an "energy trigger", capable of triggering events with high total energy deposition or multi-hadronic physics while suppressing the characteristic Bhabha scattering events [8].

However, this energy trigger is not enough when considering low multiplicity final states, such as τ-pair production and the growing beam backgrounds. With the objective of increasing the signal-to-background ratio as much as possible, a track reconstruction at the L1 trigger is indispensable. For this purpose, the CDC is capable of providing, in principle, the necessary three-dimensional track information on charged particles at the L1 trigger. This includes the particle's momentum, charge, curvature radius, z-vertex origin, and ϕ and θ angles. Using such a trigger at L1 allows for a veto of tracks $|z| \gg 0$, i.e., tracks not originating from the IP [3]. Note that the VXD is not used at the L1 trigger. When an L1 trigger decision is given, the data of the PXD detector is stored for offline reconstruction only, while the SVD can be used in the HLT for a more precise vertex reconstruction [4].

In Fig. 4.1, a simplified version of the current L1 trigger pipeline is depicted. The readout

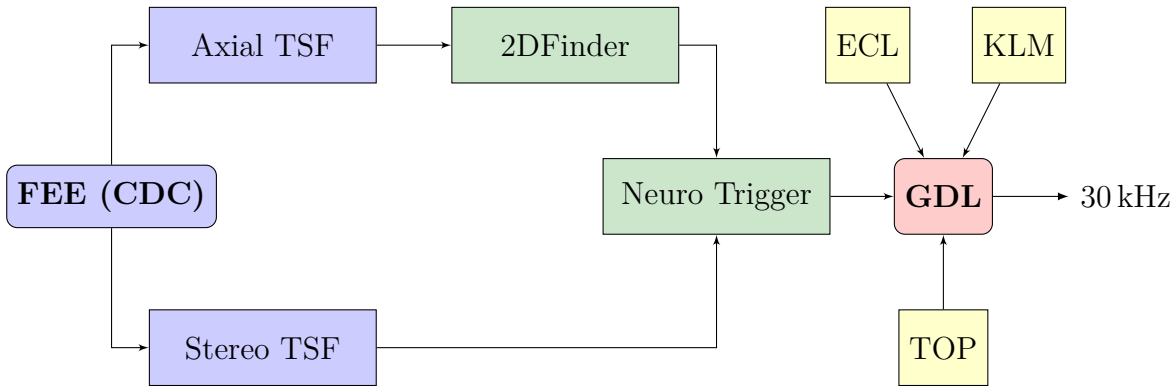


Figure 4.1: The current L1 CDC trigger system. Here only the neural network with 2DFinder input is used.

of the front-end electronics (FEE) of the CDC provides CDC wire hit information for two types of track segment finders (TSF).

Each super layer has its own TSF module implemented on an FPGA board (9 in total) [19]. An axial TSF searches for (axial) track segments (TS) in an axial super layer, while the stereo TSF searches in a stereo super layer (see Sec. 4.4). With a clock frequency of 62.5 MHz, the CDC wire hits are converted to track segments. Thus, every 16 ns, new track segments could be provided for the following modules:

The axial track segments are passed to the 2DFinder module (see Sec. 4.5). This module provides a two-dimensional track candidate with its corresponding axial track segments to the Neuro Trigger module. Using a 2DFinder track and the stereo track segments within the same ϕ domain of an event, the Neuro Trigger selects the corresponding stereo track segments belonging to the track. Afterwards, a feedforward neural network is used for track-finding at the L1 trigger. Using both axial and stereo track segments and the 2D-Finder track candidates, this network estimates the z and θ values of the tracks. Hence, the Neuro Trigger is capable of rejecting tracks not originating from the interaction region, i.e., $|z| > 15\,\text{cm}$, which greatly reduces the background (see Sec. 4.6). Given that the 2D-Finder runs on a different FPGA board than the Neuro Trigger, the latency budget of the Neuro Trigger is just 300 ns, which makes it impossible to implement deep learning neural networks [3]. Note that traditional track fitting, e.g., by using the least squares method like in a full offline reconstruction, is not feasible at the L1 trigger since the execution time is too long and non-deterministic.

In combination with the trigger information of the other three detectors, the event information is passed to the general decision logic (GDL). Under consideration of all inputs, the GDL makes the final trigger decision, resulting in the transmission of the event data to the HLT. Due to the limited buffer time of the Belle II detector, the complete latency of the L1 trigger pipeline must be below 5 µs.

The requirements of the CDC L1 trigger are as follows:

- High efficiency for (single) IP tracks.
- Rejection of background tracks that do not originate from the IP.
- Robustness under high backgrounds resulting from the increased luminosity (low number of fake tracks).
- Implementation on FPGA boards has to be possible while strictly adhering to the latency requirement.

In this thesis, a new method for the preprocessing of track inputs to the neural network is described, ensuring these requirements in view of the rising luminosities and expected backgrounds.

4.4 The Track Segment Finder

As a data compression and noise reduction step at the L1 CDC trigger, the CDC wire hits are compressed to track segments [4]. Here, a track segment finder (TSF), implemented on an FPGA board for each super layer in the CDC, is used [19].

A track segment consists of the hit pattern of 5 consecutive wire layers in each super layer. In Fig. 4.2, the drift cells of the CDC are displayed as squares. The sense wire is in the center of such a square, while the field wires surround it on the boundaries [4]. Note that the drift cells in super layer 0 measure approximately $7 \times 7\,\text{mm}$, whereas those in super layers 1–8 are more than twice as large, measuring $15 \times 15\,\text{mm}$ [4]. The innermost super layer, an axial layer with 8 wire layers, has a different track segment shape (“pyramid-shape”) than the other 8 super layers (“hourglass-shape”). As can be seen in Fig. 4.2 (a),

the first three wire layers are ignored due to the high background levels, while the fourth layer is called the priority layer [9]. If a hit is registered in this layer, the corresponding

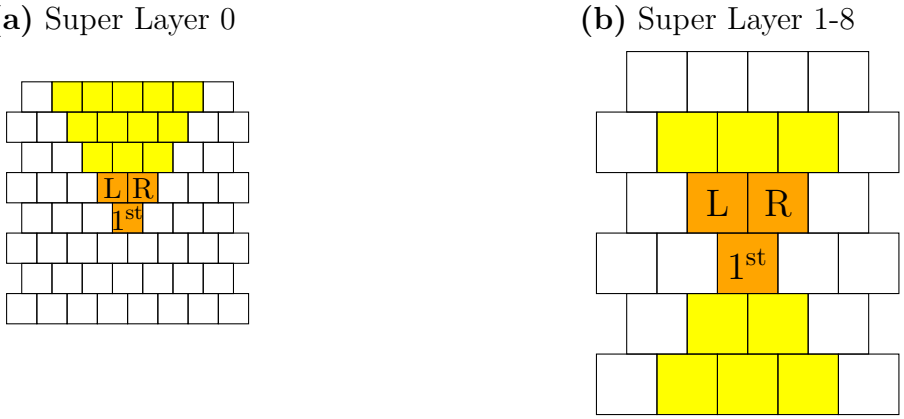


Figure 4.2: The track segment shapes of the different super layers [9]. The orange cells are the priority cells, while the yellow cells are checked when a priority wire is hit.

wire is considered to be the first priority wire, providing the distance between the beam line and the track segment and the timing information (priority timing) [19]. When no hit is present in this wire layer, one of the two secondary priority wires, "L" (left) and "R" (right), is used. The pyramid shape of the yellow cells is checked, and a track segment is created if at least one of the 3 priority wires and a total of 4 layers are hit in a certain timeframe [9]. The same applies to super layers 1–8 with the exception that the track segments are hourglass-shaped. Here, the outermost of the 6 wire layers is ignored, while the third layer is the priority layer. As in super layer 0, the yellow cells are checked, and at least 4 of the 5 layers have to be compatible with patterns from passing tracks. These patterns are predetermined in so-called Lookup Tables (LUT) [19].

In Fig. 4.3, two active track segments are displayed, where the red cells denote the hit wires and the blue cells are inactive. The patterns of the wires marked in red are used to determine whether the priority wire has been passed on the left- or right-hand side by the track [9].

In conclusion, the TSF provides the priority wire position, the drift time t_d of the priority wire, the left/right information, and the TS-ID [4]. Due to the approximately constant drift time in the CDC, t_d can be used in combination with the left/right information as a distance measure of the track to the priority wire. There are a total of 2336 possible track segments in the CDC [4].

In each clock cycle, each TSF can pass up to 10 track segments to the next trigger modules [4]. A charged track traversing the entire CDC can generate a total of 9 track segments, which enables considerably faster trajectory reconstruction compared to using every wire hit. In Fig. 4.4, an exemplary event is displayed in the cross section of the CDC, where all active wires and track segments are displayed. When considering the orange circles caused by the background, it is clear that the TSF reduces noise by only considering the allowed hit patterns. Furthermore, the track segments compress the amount of data the trigger modules have to process. It is also important to note that while the axial track segments (the super layers with gray-colored wires in Fig. 4.4) are on the two-dimensional trajectory

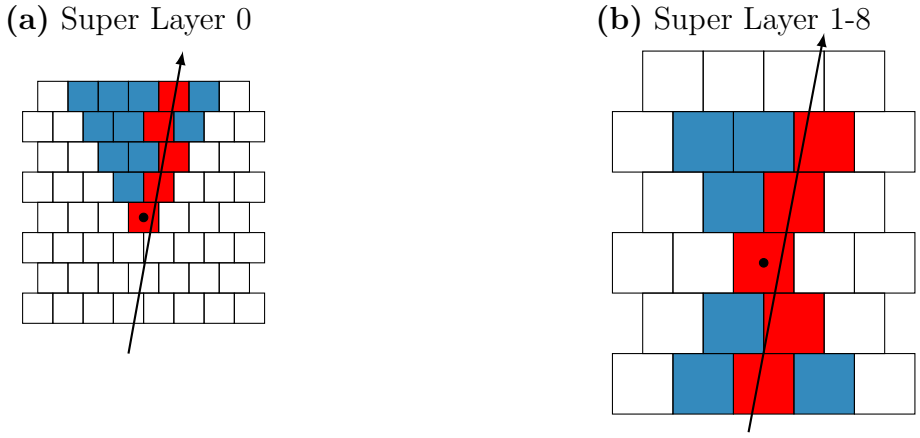


Figure 4.3: Two different track segment hits, where the red cells indicate a hit while the blue cells are inactive [4].

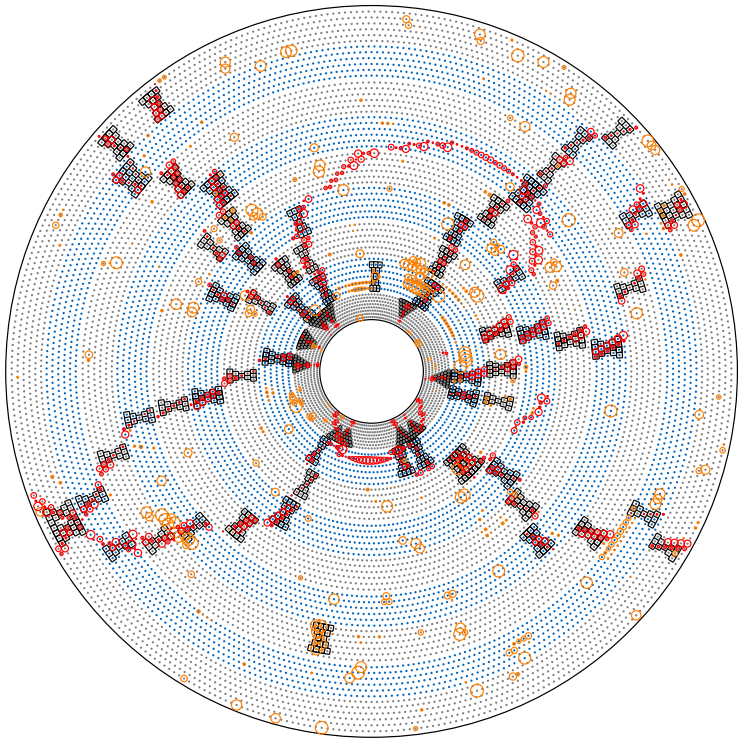


Figure 4.4: The cross section of the CDC shows an event with all the active track segments in some background [4]. All physics wire hits are marked with a red circle, while all background hits are marked with orange circles.

of a charged particle, the stereo track segments (the super layers with blue wires in Fig. 4.4) are displaced either to the left or the right of this trajectory. This is caused by the skewed angle of the stereo wires. When a hit is registered on the end plates, the hit might have happened anywhere along the z-axis. Thus, three-dimensional information can be extracted from those track segments.

It is also important to mention that a large crossing angle will not cause a track segment to be triggered, which is especially the case with low transverse momentum tracks. This can be observed in the upmost track in Fig. 4.4.

4.5 The 2DFinder

The difficult task now, given a set of track segments, is the selection of the corresponding track segments belonging to an actual unknown track. It is important that neither track segments of other tracks nor background track segments are assigned to such a track. For this purpose, a Hough transformation is used [9].

Using only the set of axial track segments from the TSF, the 2DFinder returns a two-dimensional track candidate. This arises from the fact that the axial wires are completely parallel to the z-axis, thus only providing two-dimensional information. Hence, a classical Hough transformation is employed in order to reconstruct a two-dimensional track.

The Hough transformation transforms a point from geometrical space into a curve in parameter space [4]. This parameter space is also named the Hough space. In the 2DFinder case, the point in geometrical space is an axial track segment hit in the detector. Such a hit i can be described by the coordinates of the priority wire on the CDC end plates (x_i, y_i). Now two assumptions are made. On the one hand, the track originates from the IP, i.e., $(x, y) = (0, 0)$. This is important since we only want to find physics events that necessarily originate from the IP. On the other hand, we assume that the particle's trajectory in two dimensions is circular. This is valid since the magnetic field in the CDC is nearly homogeneous and pointing into, the forward direction (positive z-direction). Furthermore, the material budget in the CDC is very low, so energy loss can be neglected [4]. Thus, the particle's trajectory can be completely described by the center of a circle [4]

$$\vec{m} = \begin{pmatrix} m_x \\ m_y \end{pmatrix} = q \cdot r_{2\mathrm{d}} \begin{pmatrix} \sin(\phi_0) \\ -\cos(\phi_0) \end{pmatrix} , \tag{4.1}$$

where q is the particle's charge, $r_{2\mathrm{d}}$ the radius of the circle, and by $\phi_0 \in [0, 2\pi)$ a parameter describing the emission angle from the IP. Note that due to the Lorentz force, $r_{2\mathrm{d}}$ is proportional to the transverse momentum p_T (see Eq. 3.3). Every hit (x_i, y_i) in the CDC must now satisfy the circle equation [4]

$$(x_i - m_x)^2 + (y_i - m_y)^2 = m_x^2 + m_y^2 . \tag{4.2}$$

Combining Eq. 4.1 and Eq. 4.2 with the definition of $\omega \equiv q r_{2\mathrm{d}}^{-1}$ yields [4]

$$\omega = \frac{2 \cdot x_i}{x_i^2 + y_i^2} \sin(\phi_0) - \frac{2 \cdot y_i}{x_i^2 + y_i^2} \cos(\phi_0) \equiv x' \sin(\phi_0) - y' \cos(\phi_0) . \tag{4.3}$$

Consequently, the hit (x_i, y_i) can be transformed into a curve in a (ω, ϕ_0) space. This curve is called a Hough curve. Note that, vice versa, any point in the parameter space corresponds to a curve in geometrical space (back-transformation). In Fig. 4.5, a hit, the black dot in subfigure (a), gets transformed into a curve in parameter space (b). The

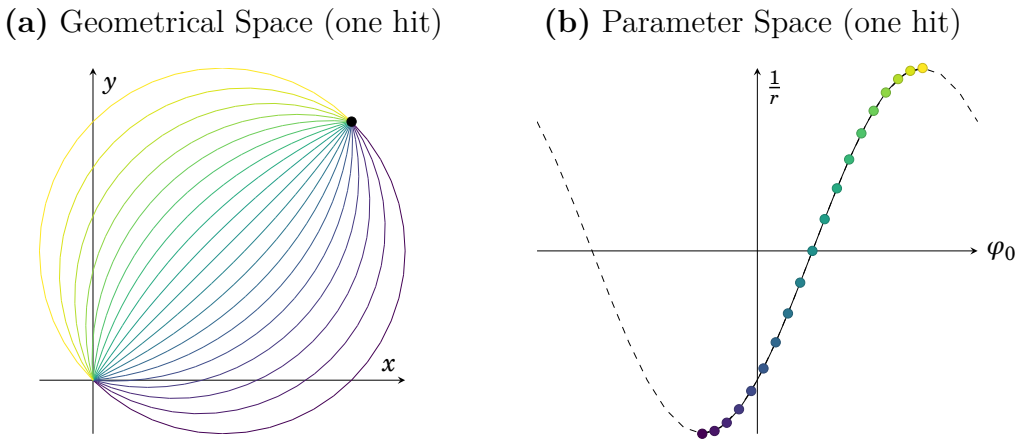


Figure 4.5: The two-dimensional Hough transformation of a single hit in geometrical space (black dot) into a single curve in parameter space [9].

different colored trajectories in plot (a) show a sample of possible tracks from the IP to the hit. These differ in the radius, i.e., the transverse momentum, and in the charge. Since $\omega = qr^{-1}$, a nearly straight line from the IP to the hit would have a very large momentum, i.e., a very large radius. Hence, $r^{-1} \approx 0$ and the corresponding curve parameter ω would be zero. Note that positively charged tracks are defined as being above the ϕ_0 axis, while negatively charged ones are below the ϕ_0 axis.

The set of hits associated with a track will create a set of curves in the parameter space. Since all of those hits are on a circular trajectory, the parameter curves will intersect at a point (ω, ϕ_0), yielding the track parameters. Accordingly, a set of random hits will not intersect at a single point and will therefore not result in a valid track.

In the case of the 2DFinder, the hits in geometrical space are selected from the priority wires in the track segments of the 5 axial super layers. A perfect track would have an intersection containing a track segment from super layers 0, 2, 4, 6, and 8. While the Hough transformation is analytical, the implementation on the trigger hardware is done through a binning of the parameter space. This is necessary for two reasons. On the one hand, the intersection point will never be perfect when using hits from the detector since the exact position from the track to the wire is unknown due to uncertainties in the digitized drift time [9]. On the other hand, it is much simpler for the hardware to calculate the intersection points when the space is binned. This is done by assigning every possible track segment a predetermined curve stored in a LUT. When a track segment is active, the 2DFinder adds the corresponding bin weights (integers) into the Hough space. By repeating this process for a fixed number of clock cycles, the peak weight in the Hough space can be determined for this time frame. When the peak weight is large enough, a track is considered to be found, the associated track parameters (ω, ϕ_0) are stored, and the corresponding track segments contributing to this maximum are read out. In Fig. 4.6,

an example of the intersecting Hough curves from a two-dimensional track candidate is displayed. The coloring of the cells indicates the corresponding cell weight.

(a) Hough Space Intersection **(b)** Weight Contributions

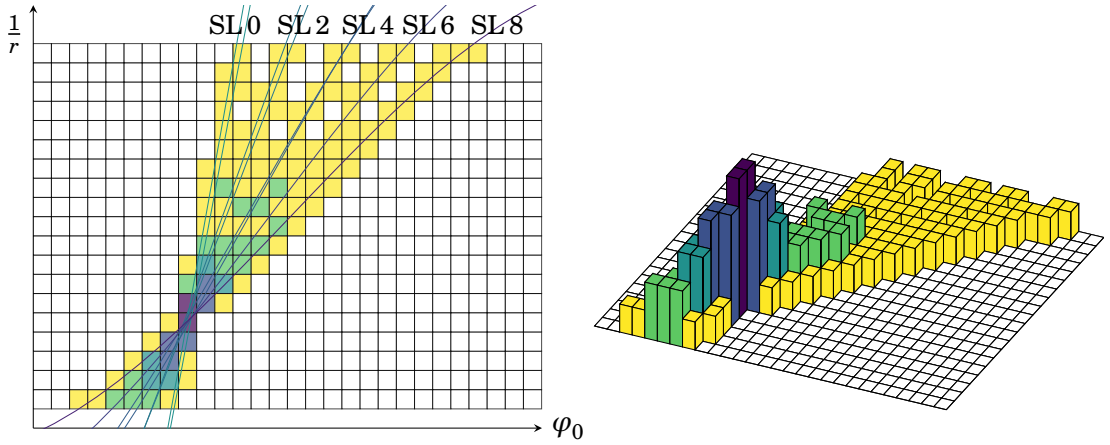


Figure 4.6: Example of intersecting Hough curves of a 2DFinder track in two-dimensional Hough space [9].

When using only the 2DFinder as a track trigger, no z-information about the tracks is available. Hence, a lot of background tracks that do not originate from the IP get triggered. As an example from the data taken early in the year 2020 ("Experiment 16"), more than 80% of tracks were from the outside, as can be seen in Fig. 4.7 [3]. Given that, the L1

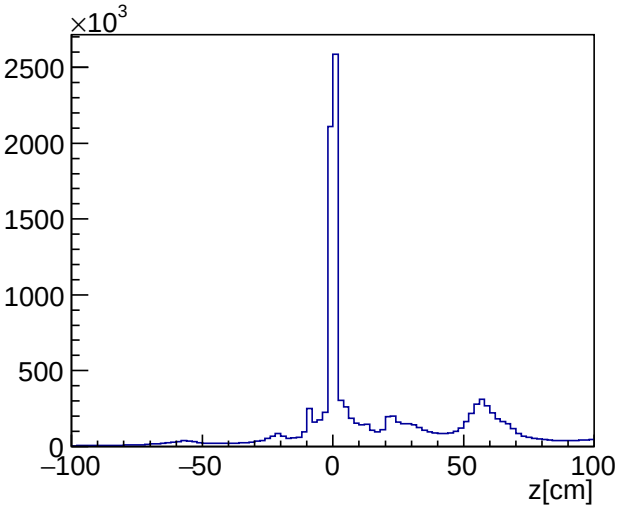


Figure 4.7: The reconstructed z-values of the triggered tracks using the 2DFinder in experiment 16 [3].

trigger has to make a vertex prediction in order to only trigger tracks originating from the IP. This should result in considerably better signal-to-background ratios, which gets more and more important since the background rates are expected to rise due to the increasing luminosity. For this purpose, the Neuro Trigger module is introduced.

4.6 The Neuro Trigger

As the execution time of classical track-finding (e.g., using the least squares method) is of non-deterministic length and not within the latency budget of the L1 trigger, a faster method for track reconstruction is necessary. Thus, the decision to employ neural networks was made. Neural networks provide a very fast and deterministic execution time while delivering accurate predictions given a sufficient architecture and training data. The following description of the principles of this neural trigger is explained in detail in reference [4].

The Neuro Trigger receives the 2DFinder tracks, i.e., the (ω, ϕ_0) parameters with the corresponding axial track segments, and the set of stereo track segments in this event from the stereo TSF. Using the stereo track segments in combination with the 2DFinder track makes it now possible to retrieve three-dimensional information. With the axial track segments, the Neuro Trigger selects the stereo track segments such that they are most likely to be on the particle's trajectory. In each of the 4 stereo super layers, at most one track segment can be selected. This is done by defining a region $[\phi_{\text{min}}(\text{SL}), \phi_{\text{max}}(\text{SL})]$ around the ϕ_0 given by the 2DFinder track for each super layer [9]. If a stereo track segment is in this region and was active in the corresponding time interval, it is considered a hit candidate [9]. Of those candidates, the one with a known left/right state and the shortest drift time is selected [9].

Given the axial and stereo track segments with the 2DFinder track parameters, a feedforward neural network is used to make a z and θ prediction, where $\theta \in [0, \pi]$ is the polar angle starting from the positive z-direction. The standard neural network already implemented in hardware is displayed in Fig. 4.8 [3]. It has $27 = 9 \times 3$ input nodes, 81 hidden nodes,

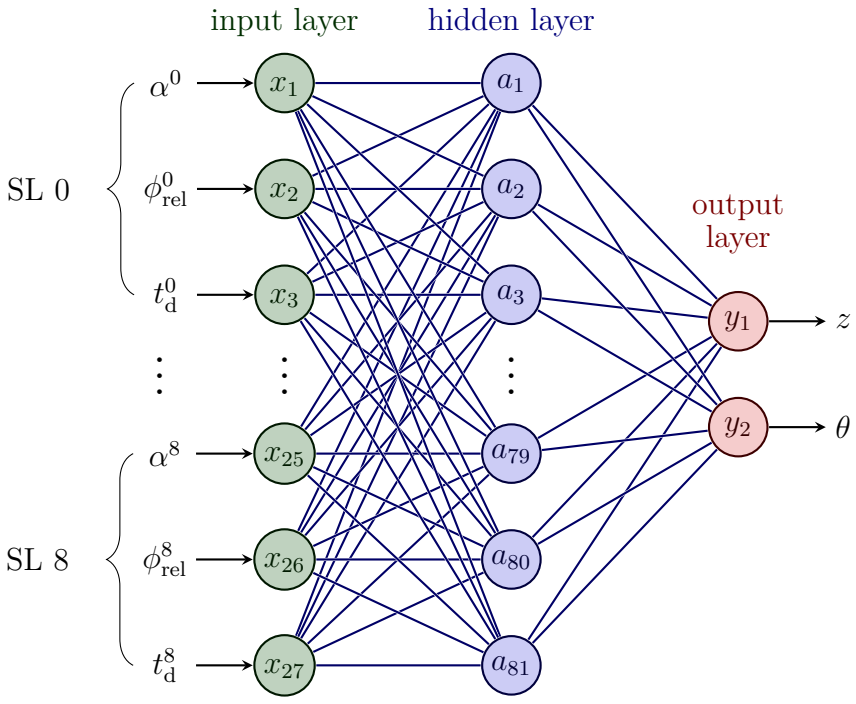


Figure 4.8: The standard neural network. There is an input layer of 27 nodes, one hidden layer with 81 nodes, and one output layer with 2 nodes [3].

and 2 output nodes for z and θ. Each super layer i contributes at most one track segment. Every track segment has 3 input nodes: $(\alpha^i, \phi^i_{\text{rel}}, t^i_{\text{d}})$, where α is the crossing angle, ϕ_{rel} the relative azimuthal angle, and t_{d} the drift time of the priority wire. In Fig. 4.9, the geometrical interpretation of the three input variables is given. The arrow indicates a

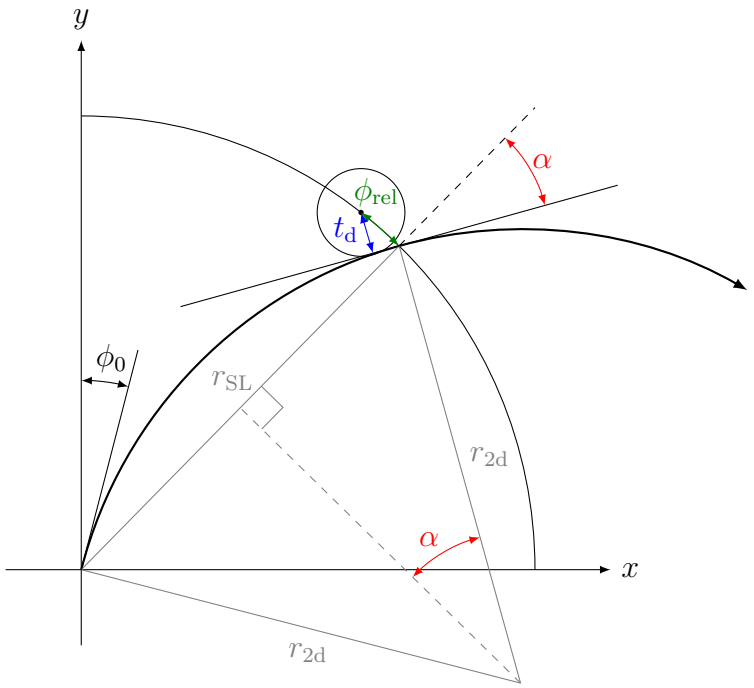


Figure 4.9: The three input variables per super layer for the current neural network [4].

2DFinder track with a radius of $r_{2\text{d}}$ in the geometrical (x, y) plane. The quarter circle is a priority wire layer with a constant[2] distance of r_{SL} from the IP, where the hit priority wire is displayed as a black dot.

As can be seen in Fig. 4.9, the crossing angle can be calculated with

$$\alpha = \arcsin\left(\frac{1}{2}\frac{r_{\text{SL}}}{r_{2\text{d}}}\right). \tag{4.4}$$

Hence, α can be in the range $\alpha \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ [4]. The ϕ_{rel} angle is the difference between the angle of the priority wire ϕ_{wire} and the intersection angle ϕ_{ref} of the 2DFinder track with the priority wire layer given as [4]

$$\phi_{\text{ref}} = n_{\text{wire}} \cdot \left(\frac{\phi_0 - \alpha}{2\pi}\right), \tag{4.5}$$

where n_{wire} is the number of wires in the layer of the priority wire. The drift time t_{d} gets a sign according to the left/right information of the track segment. Those three input parameters are mapped to the interval $[-1, 1]$ and passed to the 27 neural network input nodes. It is important to note here that the current Neuro Trigger uses 5 different neural networks, called experts, of which one gets selected for the corresponding track. One

[2]For a stereo wire layer, the skewed angles are small enough to consider r_{SL} as a constant [4].

network is used in the case where no stereo track segment is missing, while the other 4 networks are used when one of the 4 stereo layers is missing.

Fig. 4.10 shows the z-prediction of the standard neural network on reconstructed IP tracks of the last 50 runs[3] in experiment 26 (May to June 2022) with an exemplary cut of $\pm 15\,\mathrm{cm}$ on the z-prediction. In Fig. 4.11, the z-resolution, i.e., the difference between the neural

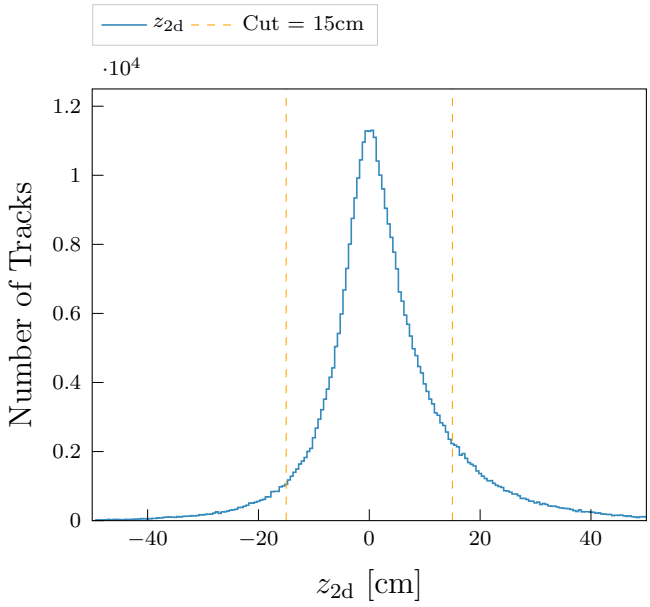


Figure 4.10: The z-distribution of the standard neural network with 2DFinder input. The tracks are related to the reconstructed IP tracks of the last 50 runs of experiment 26, and the $|z|$-cut of 15 cm is shown.

network prediction and the offline reconstruction, is depicted for two different experiments. While the neural network in subfigure (a) has been trained with the FANN library of basf2, subfigure (b) displays the z-resolution for a retrained network utilizing the PyTorch [20] library. The Δz-distributions are phenomenologically fitted with a double Gaussian.

4.6.1 The Single Track Trigger

Using this Neuro Trigger module, a minimum bias trigger, known as the single track trigger (STT), was implemented [3]. The minimum requirement for a positive trigger decision was a single neural network track with a $|z| < 15\,\mathrm{cm}$ prediction and a transverse momentum of $p_\mathrm{T} > 0.7\,\mathrm{GeV}/c$. The cuts on the momentum were introduced in order to remove low-momentum background. In the data-taking period from March 2021 until the long shutdown (LS1) in June 2022, the single track trigger was active. The efficiency of the STT significantly outperformed the former two-track triggers, as can be seen in Fig. 4.12 [3]. Despite the exceptionally high background during this experiment, the resolution of the neural networks did not suffer for IP tracks, although the networks were trained using real data with a much lower background. But the trigger rate of the STT increased from

[3]Note that this data has already gone through the L1 trigger during the data taking, which includes the Neuro Trigger, i.e., the data is not unbiased.

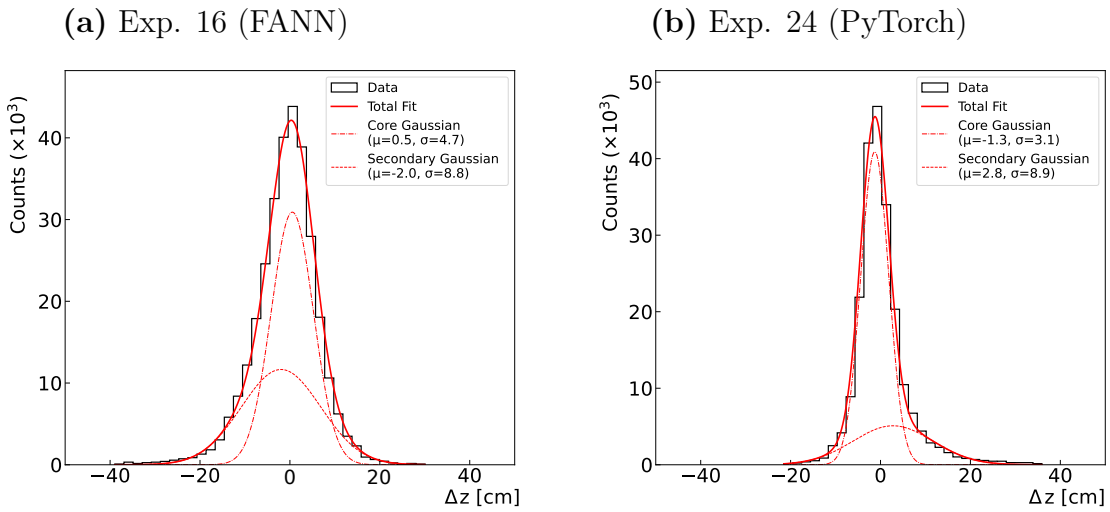


Figure 4.11: The z-resolution of the Neuro Trigger for two different experiments [3]. Note that two different training libraries have been used in the two subfigures.

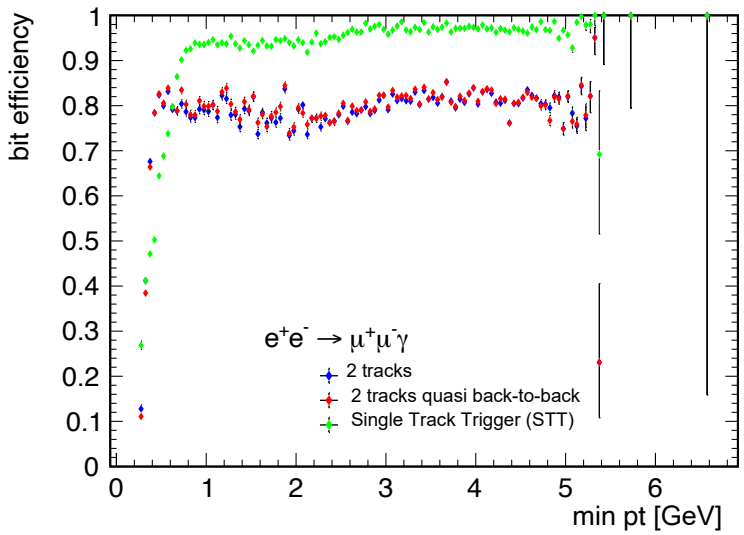


Figure 4.12: The efficiency of the STT compared with the two-track triggers for μ-pair production [3].

the usual 20% to 50% of the total trigger budget [3]. As can be seen in Fig. 4.13, the z-resolution decreases with large $|z|$, causing a so-called "feed-down" effect. A band in the

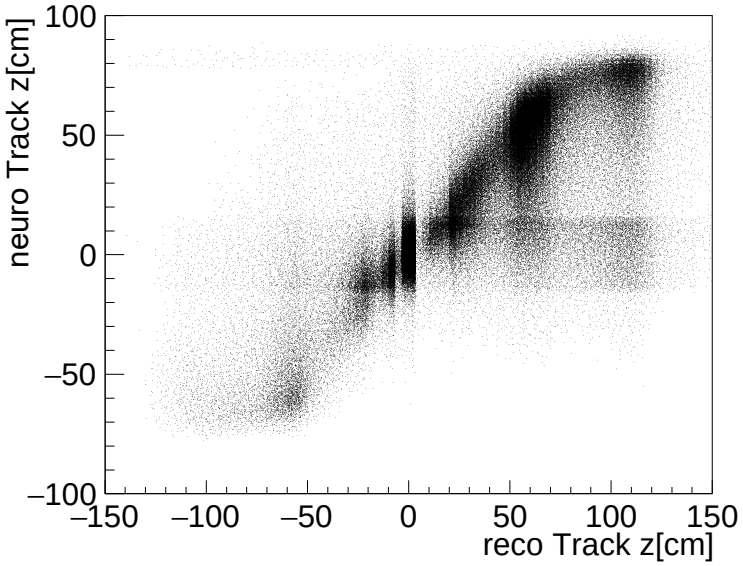


Figure 4.13: The correlation between the reconstructed z values triggered by the single track trigger (STT) and the corresponding neural network prediction in experiment 26 [3]. The observed band in $|z| < 15\,\text{cm}$ is the "feed-down" effect, causing an incorrect trigger decision.

$z \in [-15, 15]\,\text{cm}$ region is visible in this correlation plot, where the reconstructed z values are plotted against the corresponding neural network predictions of the STT. This causes a trigger signal, despite the origin of tracks being predominately from the $z \in \pm[50, 100]\,\text{cm}$ region [3]. This is mostly due to the fact that most of the training data for the neural network originates from the IP, while displaced tracks and high background levels are significantly underrepresented.

Moreover, a lot of fake neural tracks are created by the large amount of fake 2DFinder tracks, as can be seen in an exemplary event in Fig. 4.14. All the orange tracks are neurotracks created by 2DFinder input, while not a single reconstructed track was found in this event. The occurrence of fake tracks stems from the extensive background, which produces numerous active track segments in the CDC. This results in the generation of multiple 2DFinder tracks through accidental combinations, despite the absence of any physics signal in the event. Furthermore, the Neuro Trigger only has a latency budget of 300 ns, which only allows for a neural network with one hidden layer. As new research strongly suggests [21], deep learning architectures with 3 or 4 hidden layers and extended input are expected to have a much better resolution and therefore a better efficiency and rejection rate (see Chap. 8). In order to implement those architectures, a larger latency budget for the Neuro Trigger is required.

Although the Neuro Trigger has been running since January 2021 with remarkable success, the problems of the deteriorating z-resolution for off-IP tracks, the "feed-down" effect, and high fake rates have to be addressed, extrapolating to a desired further increase in luminosity [3].

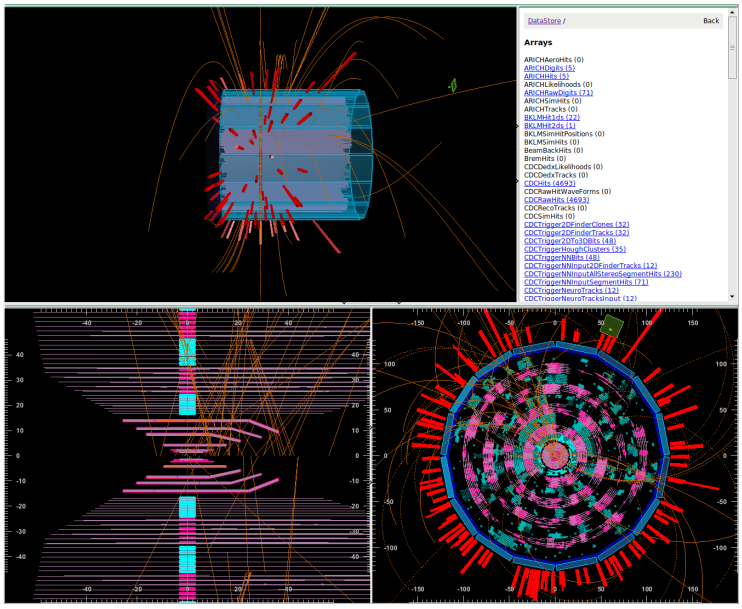


Figure 4.14: The event display of event 4 in run 33 of experiment 26. The orange tracks are 2DFinder and neural network tracks. Since there is not a single reconstructed track in the event, all tracks must be fake tracks.

Chapter 5

Analysis of the Original 3DFinder

5.1 The 3DFinder

One promising way to improve the performance of the neural trigger under high background conditions is to extend the Hough transformation used in the 2DFinder to three dimensions[1] [4]. This is done by not only using the axial track segments but also including the stereo hits in the track-finding step, which has multiple advantages. Firstly, the stereo track segments are now selected according to a physical model, where the hit must be in more precise agreement with a track hypothesis. Hence, fewer fake tracks and a better resolution are to be expected. Secondly, while in the two-dimensional case only a vertex hypothesis in $(x_0, y_0) = (0, 0)$ was made, this can now be extended to $(x_0, y_0, z_0) = (0, 0, 0)$. Therefore, tracks with a large displacement with respect to the beam axis ($|z| \gtrsim 50\,\text{cm}$) will naturally be suppressed, which will significantly reduce the background. Lastly, the 3DFinder is expected to increase efficiency because missing axial track segments can be compensated by stereo track segments. As a result, deficiencies in the drift chamber should have a smaller impact on the track-finding efficiency. Furthermore, shallow tracks ($\theta \in [19, 35]^\circ \cup [123, 140]^\circ$) and low momentum tracks ($p_\text{T} \in [0.25, 0.35]\,\text{GeV}/c$) with missing outer track segments are more likely to be found.

To achieve this, a new Hough parameter, the polar angle θ, is introduced [4]. While the two-dimensional Hough transformation of the 2DFinder contains a circular track hypothesis in the transverse plane, the 3DFinder will now make a helical track hypothesis in three dimensions with the IP as the origin for every track. Such a candidate track is completely described by the three parameters ω, ϕ, and θ.

5.1.1 Construction of the Hough Space

In the two-dimensional case, the two spacial axes were ω with 34 bins and ϕ with 160 bins, while in the three-dimensional Hough space, this binning is extended, for improved parameters of the track candidates, to 40 bins in ω and 384 in ϕ [4]. For the new parameter θ, 9 bins are chosen in this thesis, which allows for the three-dimensional Hough space to be imagined as 9 two-dimensional Hough spaces stacked above each other. The acceptance ranges of the parameters are listed in Tab. 5.1. The increased binning alone allows for a more precise parameter estimation compared to the 2DFinder, resulting in better input data for the neural network. Furthermore, the θ estimate could now be used as input (or as expert networks) in the neural network architecture.

While the ϕ and θ bins can be linearly mapped to corresponding track parameters, the ω

[1] This was done by Sebastian Skambraks in his PhD thesis (see [4]).

Table 5.1: The acceptance ranges and the binning of the Hough space of the 3DFinder.

Parameter	# Bins	Acceptance
ω	40	$p_\mathrm{T} \in [0.2, 10]\,\mathrm{GeV}/c$
ϕ	384	$\phi \in [0, 360]^\circ$
θ	9	$\theta \in [19, 140]^\circ$

bin includes the charge and the inverse of the transverse momentum. Using Eq. 3.3, ω can be expressed like in the 2DFinder case as

$$\omega = \frac{q}{r_\mathrm{2d}} = \frac{qB}{p_\mathrm{T}}\,. \tag{5.1}$$

The value of ω is mapped to the bin values $[0, 39]$, where the subset $[0, 19.5)$ denotes positively charged tracks, i.e., the trajectories have a clockwise curvature, and $(19.5, 39]$ describes negatively charged tracks, i.e., the trajectories possess a counterclockwise curvature. Note that an ω value of exactly 19.5 would correspond to infinite momentum, i.e., the track is a straight line. The ω-bin value is transformed to p_T as

$$p_\mathrm{T}[\mathrm{GeV}/c] = \frac{-1}{-5 + (\omega_\mathrm{bin} + 0.5) \cdot 0.25}\,, \tag{5.2}$$

which is plotted in Fig. 5.1. The resolution for lower-momentum tracks is considerably

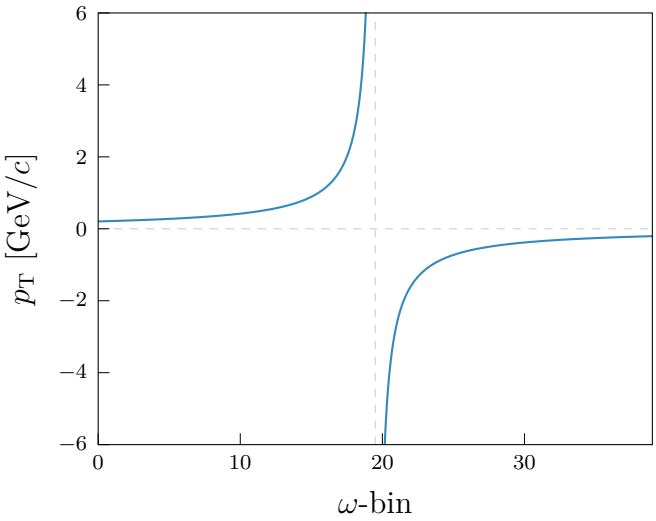


Figure 5.1: The relation between the ω-bin number and the transverse momentum p_T in the three-dimensional Hough space.

better than for high-momentum tracks. As can be observed in Fig. 5.1 for either charge, nearly 16 ω-bins cover the $p_\mathrm{T} \in [0.2, 1]\,\mathrm{GeV}/c$ area, while less than five bins have to cover the remaining transverse momentum spectrum. This is expected since the radius of the curved trajectories increases with greater transverse momentum. When the radius is large relative to the CDC dimensions, the track is nearly a straight line within the CDC, and therefore the momentum estimate is very imprecise. Since the predicted trajectory

of the 3DFinder only needs to be compatible with the real trajectory, a deviation of large momentum predictions does barely change the trajectory in the CDC, as the track is nearly a straight line anyway. Furthermore, the majority of tracks created by physics events are from the low-momentum domain. For example, the average momentum in B-events is only about $500\,\mathrm{MeV}/c$.

While the two-dimensional Hough transformation maps a track segment in geometrical space to a two-dimensional curve of sinusoidal shape in parameter space, the 3DFinder transforms a track segment to a three-dimensional curved plane in the Hough space. Those planes are called hit-representations and are stored in lookup tables (LUTs). Every cell in such a Hough plane has a weight contribution of $w \in [1, 2, \ldots, 7]$. Hence, 3 bits are used for the storage of a weight. The weight is a measure for the path length of a track traversing the three-dimensional Hough cell. When some of the 2336 track segments get activated in an event, the corresponding Hough planes are added into an empty Hough space, which is initialized with zeros only.

With respect to the ϕ-dimension, two features have to be mentioned here. Since the CDC wire pattern repeats itself every 11.25°, it is possible to divide the CDC into 32 ϕ-sectors (12 bins each), reducing the number of unique track segments to be stored in LUTs from 2336 to only 73. Furthermore, the Hough space is "wrapped" in the ϕ-dimension because $360° \mathrel{\hat{=}} 0°$. This means that, for example, bin $383 + 1$ must correspond to bin 0, and bin $0 - 1$ must correspond to bin 383. Therefore, Hough planes describing track parameters in the $\phi = 0°$ region continue on the other side of the Hough space as well.

A real track following a helical trajectory originating from the IP must now create an intersection of all the Hough planes in the three-dimensional Hough space. Hence, the intersection is a three-dimensional cluster containing the weight contributions of each individual track segment of the track. In Fig. 5.2, an example of the Hough space is displayed, containing a single muon track generated by Monte Carlo. Note that only the relevant ϕ-range around the track is displayed, not the complete Hough space. Each θ-bin is displayed in a different two-dimensional plot for illustration purposes, while the real three-dimensional Hough space is constructed by stacking each of the bins on top of each other. In this example, at θ-bin 6, all the planes intersect at a single point, causing a global maximum of the added weight contributions in the Hough space. An important observation is that the θ-bins further away from the global maximum do not intersect in a single Hough cell and therefore do not create a significant peak. When a track is displaced with respect to the IP, a single intersection point will similarly not be found, which actively suppresses background tracks.

The hit representations were created with a machine learning approach [4]. A Monte Carlo dataset of single muon tracks was created, covering the complete phase space. Using Bayesian parameter estimation, the hit representations were trained such that the cluster found by a clustering algorithm would correspond to the muon track. Hence, no analytical calculation using a predetermined track model was necessary.

When considering Fig. 5.2, a difference between the stereo and the axial Hough planes can be observed. Since the axial track segments only possess two-dimensional information and therefore no θ information, their corresponding planes (lines) are the same in every θ-bin, i.e., they do not change their directions in each of the θ-bins. This is not the case with the

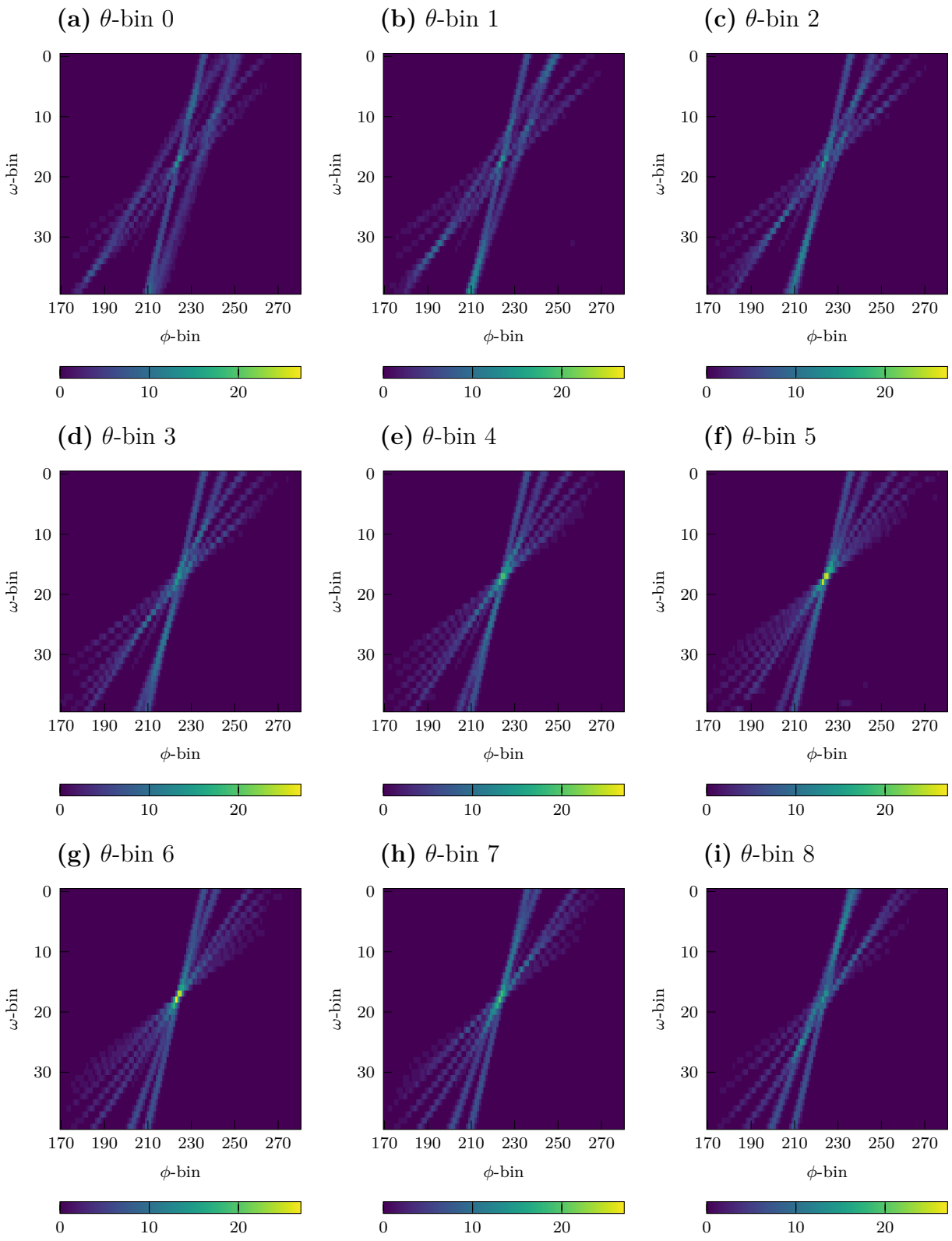


Figure 5.2: Heatmaps of the Hough space for each θ-bin of a single muon particle gun track. All heatmaps are normed to the same values.

stereo track segments. Here the planes are tilted with respect to the θ-axis, i.e., they have a different direction in each θ-bin. Therefore, the stereo hits determine the intersection point, which makes it now possible to precisely determine the stereo track segments contributing to a track.

It should be noted here that the steepness of the Hough planes with respect to the ϕ-axis is dependent on the super layer position. Track segments that are close to the IP are much steeper than those in the outer super layers because the ϕ-angle is more restricted by an inner track segment. An outer track segment could be hit with multiple trajectories of different curvatures and, therefore, a wide range of emission angles.

5.1.2 Track-Finding in the Hough Space

Given a physical event in the CDC with possibly multiple real tracks and some fake hits, the tracks now have to be found by a clustering algorithm in the Hough space of that event. For this purpose, a density-based scanning clustering algorithm (DBSCAN [22]) was originally used [4]. The first step of this algorithm is to determine a set of candidate cells in the Hough space that are considered in the clustering. A candidate cell is a cell that has a minimum weight of at least `minweight`. Now, starting in this algorithm from a randomly chosen cell, the surrounding cells are checked for neighbors. If there are at least `minpts` cells with a weight of at least `minweight` surrounding the cell, it is considered a cell belonging to the cluster. The algorithm can now be continued in two ways: The checked cells are either in the same bin in each of the six spatial directions, or, if `diagonal` is set to `true`, diagonal neighbors are included as well. Hence, a $3 \times 3 \times 3$ volume is considered, i.e., 26 cells. This procedure is repeated for each of those neighbors with a minimum weight of `minweight`. As a consequence, the cluster can expand in all directions and take any form governed only by the local density of the Hough space. The cluster finding is done iteratively until every candidate cell has been checked and has now either been assigned to a cluster or discarded. A cluster is required to now have at least $1 +$ `minpts` cells, but is only kept if it has a total number of at least `mincells` cluster cells.

Given a cluster, the corresponding hits ("priority wires") making up the cluster have to be found. This is done using a two-dimensional matrix ("confusion matrix"), where the track segment hits are listed as rows, while the columns are determined by the clusters. Now the weight contribution $w_{i,j}$ of the hit i to the cluster j is written into this matrix. Note that this can be any integer between 0 and 7. A hit is now assigned to a cluster if it contributes the highest weight to that cluster and does not have a relative weight contribution of less than `minassign` to the second-largest contribution of a different cluster[2]. When a cluster has been assigned less than `minhits` track segments, the cluster is deleted. In an iterative process, the hits of those deleted clusters are attempted to be reassigned to the clusters that survived the `minhits` cut. Furthermore, a parameter `minhits_axial` is introduced, which makes a cut on the minimum number of axial track segments along with the `minhits` cut, which considers all track segments.

The remaining clusters now have their own unique set of track segments associated with them and are considered to be found tracks. Now the track parameters are to be deter-

[2] A detailed explanation and analysis of the `minassign` parameter and the whole hit-to-cluster association can be found in Chap. 8.

mined. By calculating the weighted mean of the cluster, all cells in the cluster that have a total weight of at least `thresh` times the peak weight of this cluster are used in the calculation. This step significantly improves parameter estimation. When, for example, a track has a polar emission angle θ between two of the 9 bins, the weights in the θ-layers above and below will have a similar weight, resulting in a mean position right in between those bins. Therefore, the parameter estimation is more precise than the limited binning in the Hough space when calculating this center of gravity instead of just using the peak cell of a cluster.

The mentioned parameters and their default values are listed in Tab. 5.2. Note that a

Table 5.2: The 3DFinder parameters with their default values [4].

Parameter	Value	Description
`minweight`	24	minimum weight of a cell in Hough space
`minpts`	1	minimum number of neighbor cells with minweight
`diagonal`	`True`	consider diagonal neighbors
`mincells`	1	minimum number of cells for a cluster
`minhits`	4	minimum number of hits related to a cluster
`minhits_axial`	0	minimum number of axial hits
`minassign`	0.2	minimum relative weight contribution to the largest cluster
`thresh`	0.85	minimum weight of a cluster cell relative to the peak

`minweight` of 24 means that at least 3.5 hits from the same track have to be present in such a cluster. In the default parameters, the `mincells` and the `minhits_axial` variables are not used.

5.1.3 Implementation at the L1 Trigger

A new generation of FPGA boards (UT4) makes it now possible to upgrade the present neural trigger by implementing both the preselection (i.e., the 3DFinder) and the neural network on the same board [3]. With this feature, the time-consuming transmission of the 2DFinder track to the Neuro Trigger board can be avoided, which increases the latency budget to 700 ns. In Fig. 5.3, the simplified L1 trigger pipeline with the planned upgrade is displayed. Note that four different UT4 boards are going to be used, where each board is responsible for one quadrant of the CDC. The 4 quadrants are extended in ϕ to have sufficient overlap, and all the operations in the quadrants are executed in parallel.

5.2 Signal Studies

The increased latency budget will now make it possible to implement deep neural networks with up to 4 hidden layers, which significantly improve the z-vertex resolution [21]. But to use this possibility, a fast and efficient 3DFinder clustering algorithm and hit-to-cluster association need to be developed, which is the main research result of this thesis. To get a basic understanding of the capabilities of the original implementation of the 3DFinder, signal studies were conducted using the Belle II analysis software framework (basf2)

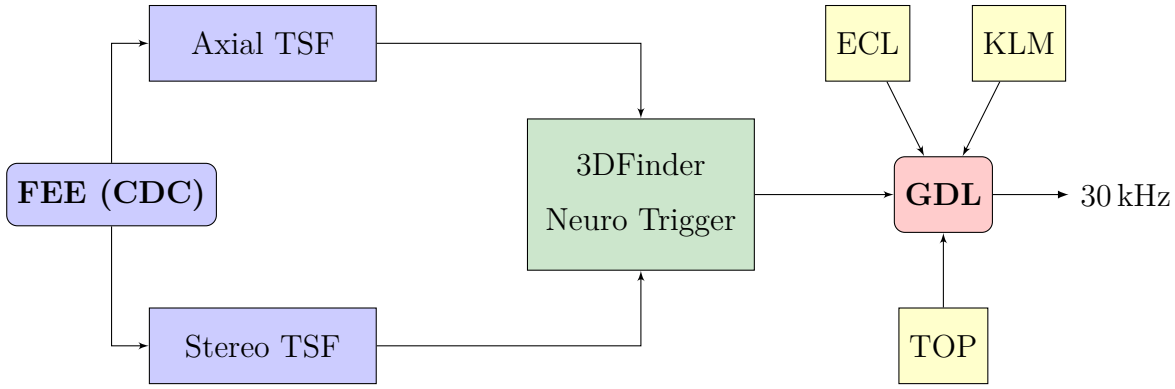


Figure 5.3: The new L1 CDC trigger pipeline is extended with 3DFinder input for the neural network. Track-finding and the neural network are both implemented on the same FPGA board.

[17]. For this purpose, only simulated Monte Carlo data was used in this chapter. Here, a "particle gun" generates a single muon with a predetermined charge, transverse momentum, emission angle, and vertex origin per event. With the software Geant4 [23], the passage of the particle through the complete Belle II detector was simulated. On this detector data, track trigger modules like the 3DFinder or the 2DFinder were tested.

This results in a very controlled environment for parameter studies of the 3DFinder. Since a muon is approximately 200 times heavier than the electron, it is less likely to interact with any detector materials in such a way that its trajectory deviates from the helical track hypothesis. Furthermore, muons have a mean lifetime of $\tau_\mu \approx 2.2\,\mu s$, which makes a decay within the CDC very unlikely[3].

To achieve the most clean events possible, the datasets in this section are 10,000 single muon track events with no background in the detector. Hence, only the track segments of a single track are written into the Hough space of the 3DFinder per event, which should make track-finding possible with a high resolution and high efficiency.

It is important to note that the neural network utilized in this thesis is the standard architecture depicted in Fig. 4.8, featuring only a single hidden layer with 81 nodes. This network has been trained exclusively on 2DFinder track candidates of low luminosity reconstructed tracks. Only in Chap. 8 a newly trained, deep neural network is introduced.

5.2.1 The Full CDC Acceptance Range with Default Parameters

The first dataset to be analyzed are tracks originating from the IP ($z = 0\,cm$) in the full CDC acceptance range, i.e., the range where the trajectories of the particles cross all 9 super layers. To ensure this, a polar emission angle $\theta \in [35, 123]^\circ$ is enforced. All the particle gun parameters are listed in Tab. 5.3. Note that the charge of each particle is picked randomly, and $(x, y) = (0, 0)$ is used throughout the Monte Carlo studies.

[3]The probability for a decay within the CDC can be calculated with $P_{\text{decay}}(d, v) = 1 - e^{-\frac{1}{\tau_\mu} \cdot \frac{d}{v}}$, where the velocity of the muon is $v = \frac{pc}{\sqrt{mc^2+p^2}}$. With a travel distance d within the CDC of under 2 m, the probability for momenta larger than $350\,MeV/c$ is below 0.3%.

Table 5.3: Parameters of the particle gun single tracks for the full CDC acceptance range.

Parameter	Range	Distribution
z	0 cm	fixed
θ	$[35, 123]^\circ$	uniform in $\cos(\theta)$
ϕ	$[-180, 180]^\circ$	uniform
p_T	$[0.4, 3]\,\mathrm{GeV}/c$	uniform

To assess the performance of the track-finding algorithms, the found track candidates are passed to the same neural network. This is the default one mentioned in Sec. 4.6 with only one hidden layer. If the track-finding algorithm is working well, the selected track segments and the track parameter predictions should yield good input parameters for the neural network. This will be sufficient for a reasonably precise z- and θ-prediction and will therefore be a good assessment of the track-finding capabilities. Furthermore, the track-finding efficiency translates directly into the efficiency of the neural network because all track candidates return a neurotrack[4]. To have a benchmark for comparison, the standard 2DFinder with its default configuration and its corresponding neurotracks are used throughout this thesis. It is very important to note here that the neural network used throughout this chapter has only been trained on 2DFinder track candidates yet. Hence, the resolution using the 3DFinder tracks may be worse than their actual potential. Nevertheless, this network is sufficient to gain insight into the current problems of the 3DFinder.

As a first step, the default parameters of the 3DFinder, as listed in Tab. 5.2, are used. Currently, two differently trained hit representations of the track segments are provided in [4] and have to be investigated. One of those hit representations is called `comp` ("complete"), which has an equally large weight contribution of all super layers. The second hit representation is called `shallow`, which has a larger weight contribution in the inner ("shallow") track segments while the weights of the outer ones are smaller. Here, the intention was to increase the efficiency of the 3DFinder for low-momentum and shallow-θ tracks.

In Fig. 5.4, the z- and θ-resolutions for the neural network predictions with 2DFinder and 3DFinder inputs are displayed. For the 3DFinder, both hit representations, the `shallow` and the `comp`, are plotted. The resolution of a given variable x is defined as $\Delta x = x_\mathrm{neuro} - x_\mathrm{reco}$, i.e., the difference between the neural network prediction and the full offline reconstruction, which uses every detector component. Due to their exceptional accuracy and efficiency, those reconstructed tracks will be considered the "truth" throughout this thesis, upon which the neurotracks efficiencies and their track hypothesis will be evaluated. Note that the maximum of the data distribution is governed by the chosen binning. To keep the plots comparable, the binning is the same in all six subfigures.

The distributions of the neural network predictions using 2DFinder input are phenomeno-

[4]Sometimes neurotracks cannot be related to a reconstructed track, resulting in their loss.

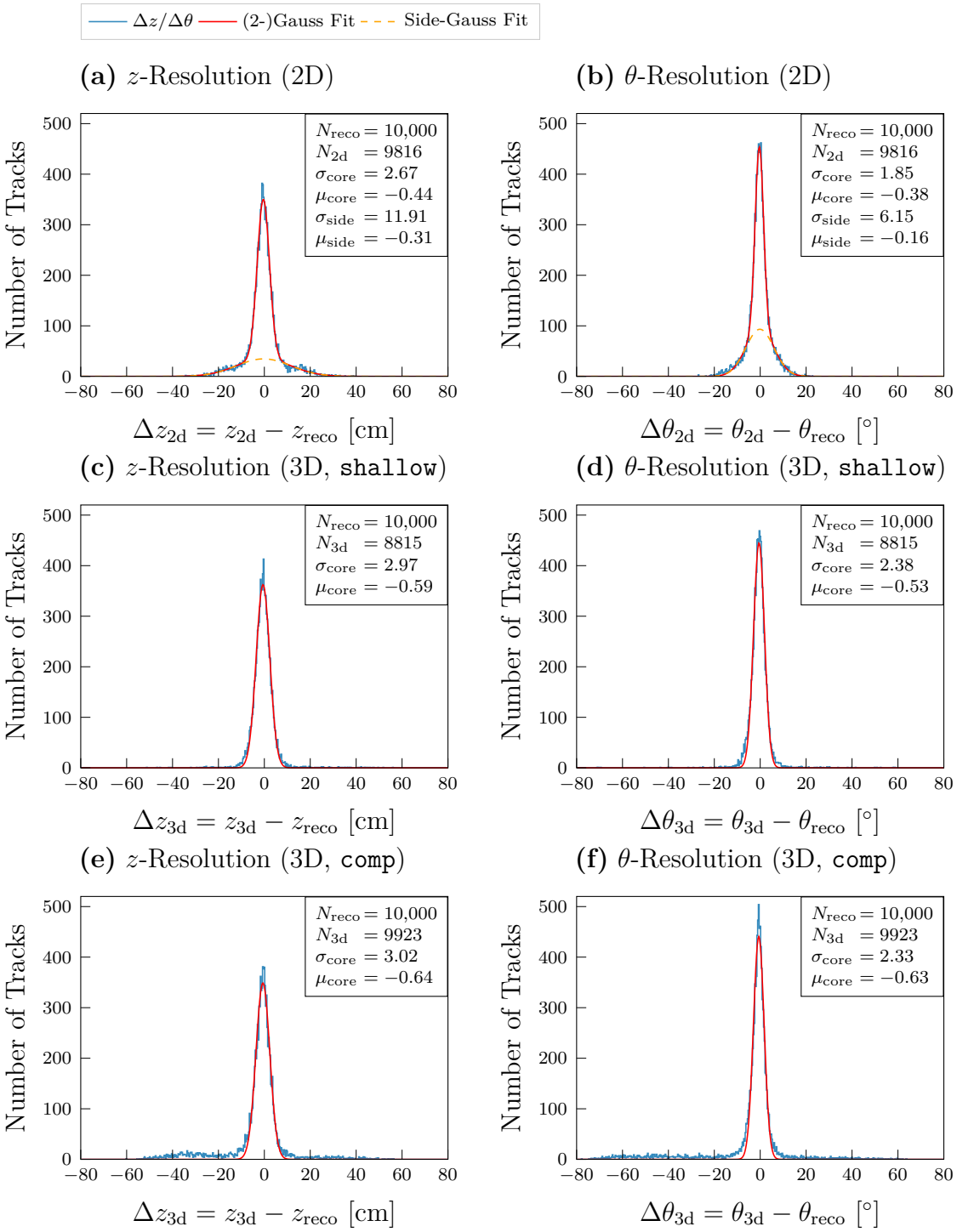


Figure 5.4: Comparison of the z- and θ-resolutions of the neural network trigger with 2DFinder and 3DFinder input. The two different hit representations (`comp` and `shallow`) were used (see text).

logically fitted with a double Gaussian, given by

$$f(x) = A_{\text{core}} \cdot \exp\left(-\frac{(x-\mu_{\text{core}})^2}{2\sigma_{\text{core}}^2}\right) + A_{\text{side}} \cdot \exp\left(-\frac{(x-\mu_{\text{side}})^2}{2\sigma_{\text{side}}^2}\right), \tag{5.3}$$

where A denotes the amplitude, σ the standard deviation, and μ the mean. A "core" Gaussian refers to the Gaussian with the smaller standard deviation. Note that for the 3DFinder tracks in Fig. 5.4, a single Gaussian fit was sufficient. While for the `shallow` hit representations, this may seem like a great improvement in the resolution, the finding efficiency is very low. Only 88.2% of all tracks have been found, while the 2DFinder found 98.2% of the reconstructed tracks. A similar problem is observed with the `comp` representations. The efficiency, at 99.2%, is very high, but a lot of tracks are displaced along the z- and θ-axis. In Tab. 5.4, the percentages of the neural network predictions within an interval of three standard deviations of the core Gaussian, i.e., $3\sigma_{\text{core}}$, are listed. Considering $N_{\text{out}}/N_{\text{found}}$, it gets clear that both hit representations are equally bad with

Table 5.4: Distributions of Δz and $\Delta\theta$ of the neural network trigger with default parameters.

Fig. 5.4	**Track Finder**	N_{found}	N_{in}	N_{out}	$N_{\text{in}}/N_{\text{found}}$	$N_{\text{out}}/N_{\text{found}}$
(a): z	2DFinder	9816	8247	1569	84.0%	16.0%
(b): θ	2DFinder	9816	8115	1701	82.7%	17.3%
(c): z	3DFinder (`shallow`)	8815	8566	249	97.2%	2.8%
(d): θ	3DFinder (`shallow`)	8815	8500	315	96.4%	3.6%
(e): z	3DFinder (`comp`)	9923	8392	1531	84.6%	15.4%
(f): θ	3DFinder (`comp`)	9923	8291	1632	83.6%	16.4%

the default parameters. The high efficiency of the `comp` representation is counteracted by poor resolution, whereas the good resolution of the `shallow` representation is useless due to the poor efficiency. This calls for an explanation, as laid out in the section below.

5.2.2 First Parameter Optimization

Investigation of the `shallow` Hit Representations

There could be two reasons for the low efficiency of the `shallow` representations. On the one hand, the clusters may not be found in the first place because the combined weights at the maximum are not large enough. This could occur either because of the small individual weight contributions for each track segment of the `shallow` hit representations or due to a non-proper intersection. Here, the `minweight` parameter only allows for clusters with at least a weight of 24. Due to `minpts` = 1, two of those cells have to be in the same $3 \times 3 \times 3$ volume to be considered a cluster. On the other hand, it may be possible that the clusters are found but are rejected by the `minhits` cut on the track segments. An incorrect association of the track segments to the cluster may cause a rejection of the cluster.

To investigate this, the number of related track segments for each track is considered. In Fig. 5.5, the relative frequencies of the number of found track segments are plotted

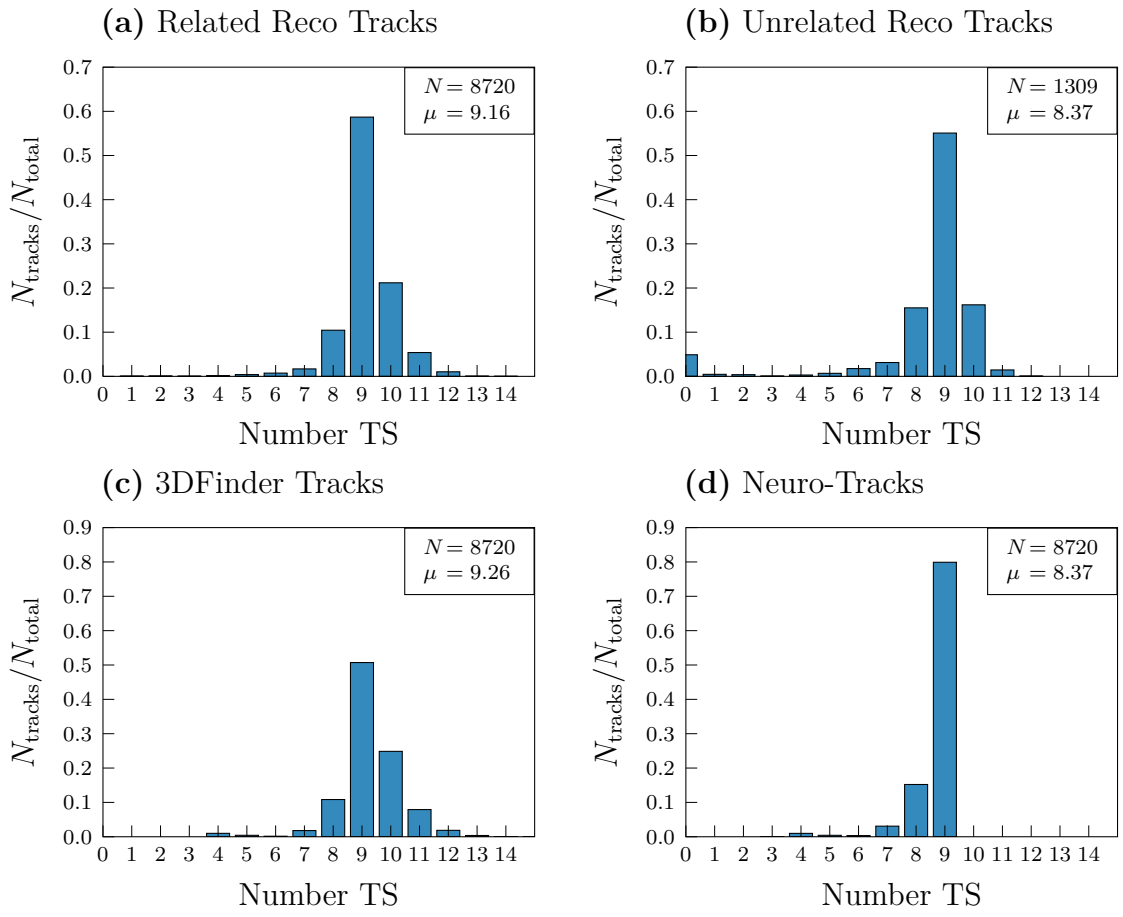


Figure 5.5: The track segment distributions of the reconstructed, 3DFinder, and neurotracks with the `shallow` hit representations. Subfigures (a) and (b) differentiate between the reconstructed tracks that have been successfully related to a neurotrack and those that are unrelated.

for different cases. In subfigure (a), only the reconstructed tracks that were successfully related to a neurotrack are displayed. A basf2 module is used to relate trigger tracks to the reconstructed tracks. Here, the track segments belonging to the tracks are compared in order to estimate whether they belong to each other. As expected, most tracks have 9 track segments, which correspond to the 9 super layers in the CDC. Note that there are a considerable number of tracks with more than 9 track segments. Those are called "duplicate" track segments, which are created when a track passes right in between two potential track segments, triggering the creation of both. In subfigure (b), the complement of (a) is plotted, i.e., all reconstructed tracks where no neurotrack has been found. This distribution is similar to (a), which indicates that the inefficiency is indeed a problem of the present set of parameters chosen for the 3DFinder and not of the tracks themselves, which should be detectable. In subfigure (c), the 3DFinder track segments are displayed, while in (d), the corresponding neurotrack track segments are plotted. Hence, nearly all track segments are correctly found when a track has been found. However, it is striking that approximately 35% of all 3DFinder tracks have more than 9 track segments, while only 50.7% have exactly 9. When considering the unrelated reconstructed tracks in subfigure

(b), only 17.7% of all tracks have more than 9 track segments. Apparently, the 3DFinder is benefiting from duplicate track segments, which increase the cell weight at the intersection point in the Hough space. Note that the neurotracks do not have any duplicate track segments. This is a requirement since the neural network can at most take one track segment per super layer as input, which causes the upper bound of 9.

Since there is no problem with the number of track segments in the unrelated tracks, the `minweight` parameter appears to reject those clusters. Therefore, the same dataset is tested with six different values of `minweight` that are smaller than the default 24. In Tab. 5.5, the counts of the different parameters are listed, and in Fig. 5.6, the corresponding z-resolutions are plotted. When decreasing `minweight`, the number of found tracks increases significantly from 88.5% to 99.0% when a value of 21 is used. Hence, the tracks were not found because their clusters in the Hough space do not have a peak weight of 24 or more. Furthermore, the z-resolution of the neural network does not decrease and can still be fitted with a single Gaussian. Consequently, changing only one of the default parameters,

Table 5.5: Fit parameters of Δz of the neural network trigger with 3DFinder input (`shallow` hit representations) under variation of the `minweight` parameter.

Fig. 5.6	`minweight`	N_{neuro}	N_{3d}	N_{out}	σ_{core}	μ_{core}	A_{core}	σ_{total}	μ_{total}
(a)	19	9913	9953	622	2.98	-0.59	392.37	8.02	-0.04
(b)	20	9902	9942	385	2.97	-0.59	402.54	6.09	-0.52
(c)	21	9862	9900	329	2.97	-0.57	403.68	5.32	-0.62
(d)	22	9741	9779	277	2.97	-0.57	401.26	4.74	-0.69
(e)	23	9436	9470	264	2.98	-0.58	387.92	4.64	-0.73
(f)	24	8821	8853	251	2.97	-0.59	362.37	4.65	-0.68

the 3DFinder outperforms the 2DFinder on single signal particle gun tracks in the full CDC acceptance range. The efficiencies for just finding the tracks are very similar, but the wide second Gaussian is not present in neural network z-predictions with 3DFinder input. When considering a cut on the z-vertex of at least 15 cm, the efficiency of the 3DFinder is considerably better due to the narrower Gaussian distribution. A side-by-side comparison of those two IP track resolutions can be seen in Fig. 5.7.

Investigation of the `comp` Hit Representations

While more than 99% of all tracks with `comp` hit representations have been found, approximately 15% of those tracks have a seemingly arbitrary z- and θ-prediction, as observed in Fig. 5.4 (e) and (f). This has to be a problem of either the clustering or the hit-to-cluster assignment. In Fig. 5.8, the track segments of those neurotracks are analyzed. In subfigure (a), all tracks are displayed, while in subfigure (b), only the tracks outside the 3 standard deviations of the Gaussian fit are considered. Most of those displaced tracks have only 5 to 6 track segments, while the vast majority of correctly predicted tracks have 9 associated track segments. Hence, the bad resolution can be attributed to missing track segments in the 3DFinder tracks, leading to missing input for the neural network. This makes a precise prediction impossible. As can be seen in subfigures (c) and (d), both axial and stereo track segments are missing for the inaccurate tracks. In (e) and (f), the correlation between the

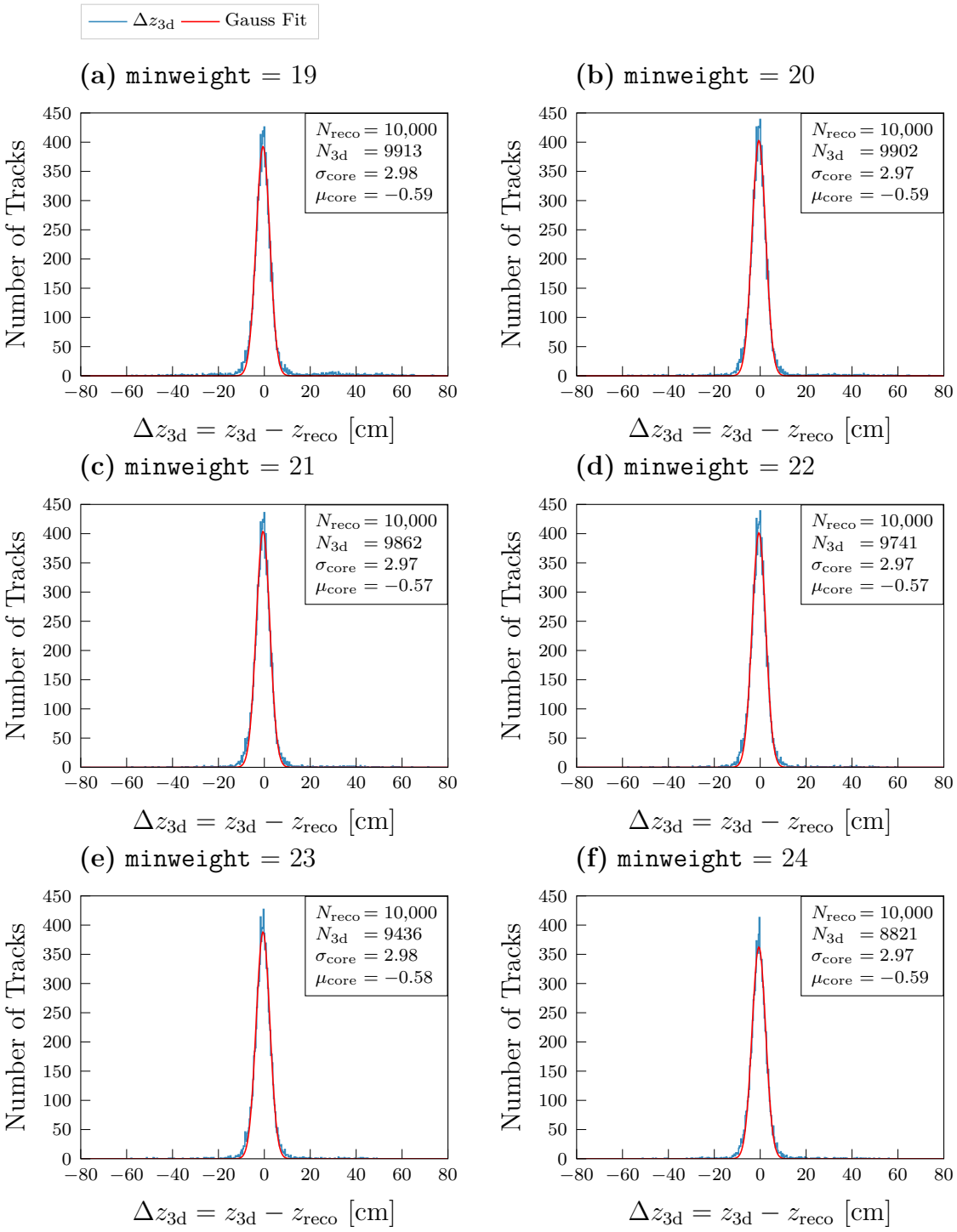


Figure 5.6: Comparison of the z-resolution of the neural network trigger with 3DFinder input of particle gun single tracks using different `minweight` parameters. The `shallow` hit representations were used, while the other parameters were left at their default ones.

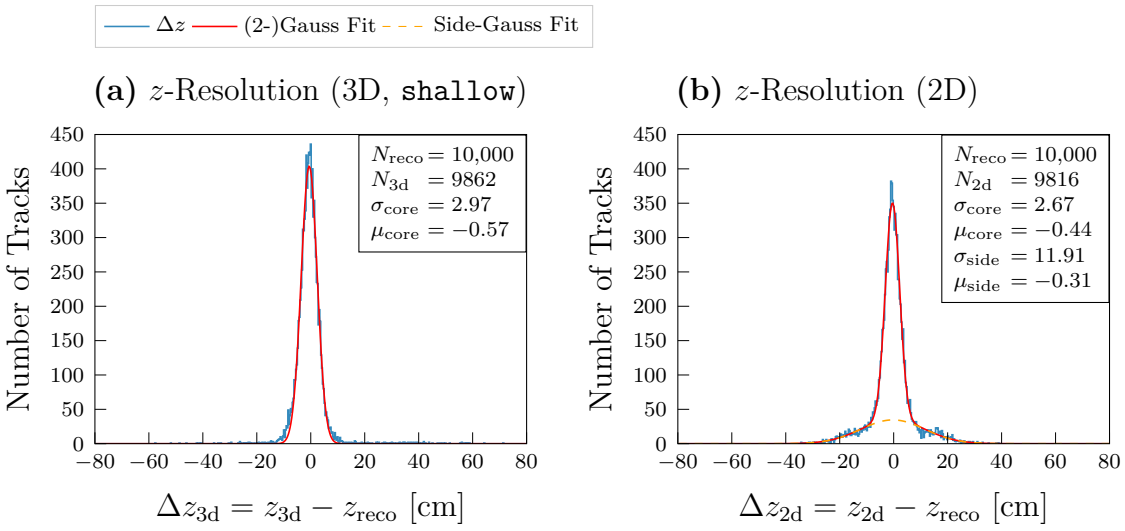


Figure 5.7: Comparison of the z-resolution of the neural network trigger with 3DFinder (a) and 2DFinder (b) input of particle gun single vertex tracks without background. Note that the `shallow` hit representations with a `minweight` of 21 have been used in (a).

found axial and stereo track segments is plotted for the two z-intervals. There seems to be no systematic problem with either wire type in this hit representation.

To find the cause of the missing track segments, the original DBSCAN clustering algorithm has to be analyzed (see Chap. 6). Furthermore, given those clusters, the hit-to-cluster association may be the cause of those missing track segments (see Chap. 8). Understanding this resolution problem at those two fundamental parts of the algorithm allows for a parameter adjustment. Due to the better performance of the `shallow` hit representations motivated by Fig. 5.7, these are used for the rest of this chapter with a `minweight` of 21.

5.3 Background Studies

As described in Sec. 4.1, a type of background are particles originating from outside the IP. The vast majority of those particles have their vertex along the beam line, i.e., along the z-axis. In experiment 16, with a comparatively low luminosity, already more than 80% of tracks were from outside the IP, as can be seen in Fig. 4.7 [3]. This will only get worse when increasing the luminosity. Hence, it is important for the L1 trigger to reject those tracks as early as possible in order to keep the trigger rate and, correspondingly, the dead time of the data acquisition low.

Due to the IP hypothesis of the 3DFinder, i.e., $(x, y, z) = (0, 0, 0)$, it is expected that tracks with a large absolute value of z are not found and will therefore not create a (potentially incorrect) neurotrack. To study this, a new simulated dataset with a random $z \in [-100, 100]\,\text{cm}$ using the same Monte Carlo particle gun has been created. The particle gun parameters of the 100,000 muons are listed in Tab. 5.6. Note that both the θ- and the p_T-distribution have been increased to get more varied trajectories within the CDC. In Fig. 5.9, the reconstructed z-distribution of this dataset is plotted. Only 74,775 of the 100,000 tracks were reconstructed. This is expected since the polar emission angle

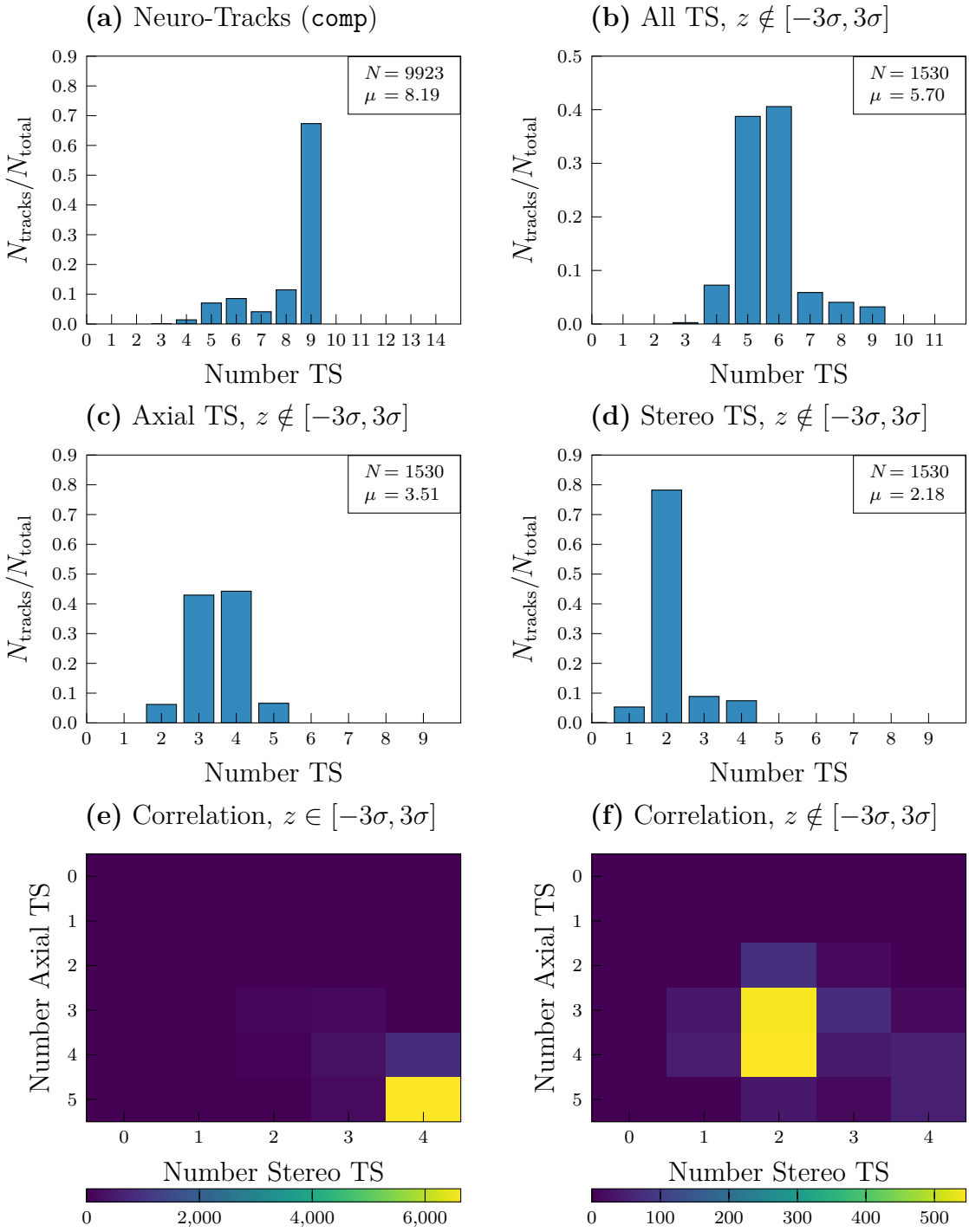


Figure 5.8: Histograms of the related track segments of single particle gun tracks using the `comp` hit representations for the neural network trigger with 3DFinder input. The bad neural network predictions are due to missing track segments.

Table 5.6: Parameters of the particle gun single muons for the large dataset for the background studies.

Parameter	Range	Distribution
z	$[-100, 100]$ cm	uniform
θ	$[19, 140]^\circ$	uniform in $\cos(\theta)$
ϕ	$[-180, 180]^\circ$	uniform
p_T	$[0.35, 10.2]$ GeV/c	uniform

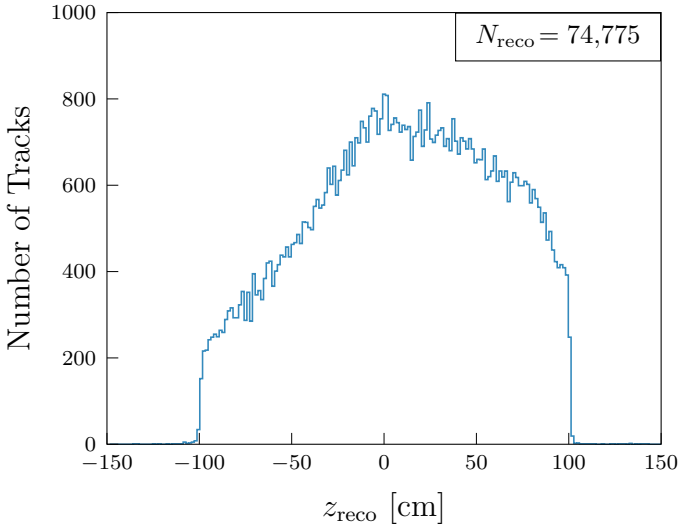


Figure 5.9: The z-distribution of the reconstructed tracks of the large dataset for the background studies.

θ may be quite shallow for some tracks, crossing only parts of the CDC. Hence, tracks towards the outer parts of the CDC are less likely to go into the CDC and cause wire hits. Despite the uniform z-distribution of the Monte Carlo tracks, an asymmetric distribution is observed, which is expected as well. Since the CDC is asymmetric and extends further into the forward region ($z > 0$), it is more efficient there than in the backward region.

In Sub. 5.3.1, no simulated background (bkg) is added to the large dataset, while in Sub. 5.3.2 and Sub. 5.3.3, simulated Monte Carlo backgrounds of different intensities are added. Here, the two background campaigns "early phase-3" (ep3) and "nominal phase-3" (np3) are used in order to study the performance of the 3DFinder. While the early phase-3 background corresponds to the observed backgrounds during the early running phases of Belle II, where the luminosity was low, the nominal phase-3 background models the expected background for the target luminosity [24]. When generating physics tracks, the simulated machine backgrounds (see Sec. 4.1) are overlaid.

In essence, there are two different types of backgrounds. One type are the real physical tracks that can be found by reconstruction but do not have their z-vertex origin within the IP. This is addressed by an accurate vertex prediction at the L1 trigger and by suppression of those tracks. The other type is due to activated sense wires in the CDC caused, for example, by synchrotron photons. This type cannot be reconstructed and, when mistakenly

triggering the 2DFinder or 3DFinder, will therefore produce so-called fake tracks. Those fake tracks have to be addressed as well since they increase the trigger rate with random events.

5.3.1 Background-free Tracks

In Fig. 5.10 (a), the z-distributions of the reconstructed and the neurotracks using 2DFinder and 3DFinder input are compared. Note that only neurotracks that have been

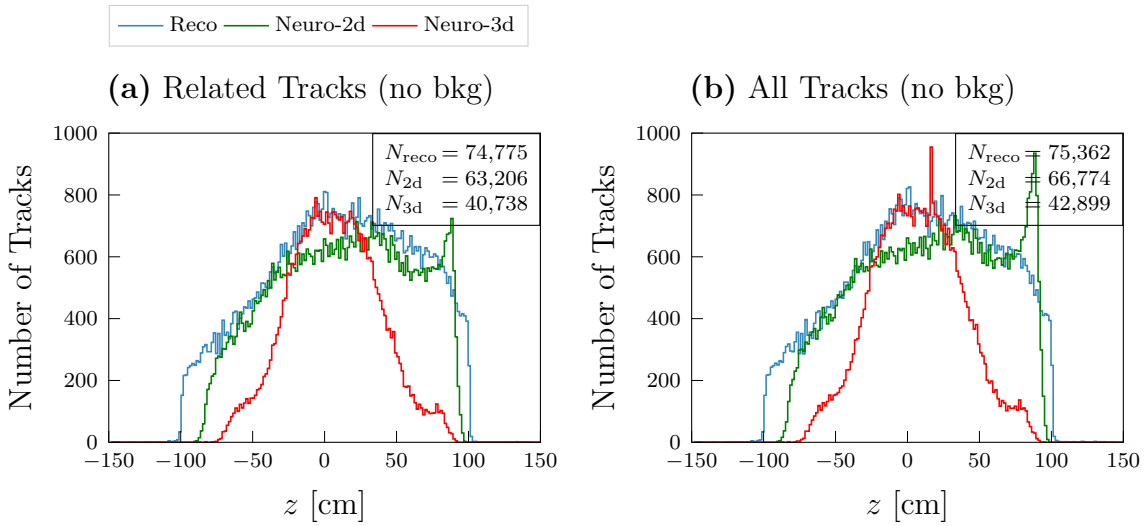


Figure 5.10: The complete z-distributions of the reconstructed, neuro-2d, and neuro-3d tracks, scaled to the same y-axis, without any simulated background. Subfigure (a) shows all tracks related to a reconstructed track that was related to a particle with a production time of 0, while subfigure (b) shows all tracks.

successfully related to a reconstructed track are plotted. While the neuro-2d tracks are efficient nearly through the whole z-range from -100 to 100 cm, the neuro-3d efficiency drops off significantly after ± 20 cm. Above $|z| > 50$ cm, nearly all background tracks are rejected. This is very important to mitigate the feed-down effects observed when using the 2DFinder as the track finder, especially when reaching higher luminosities. Furthermore, the efficiency around the IP seems to be considerably better than that of the neuro-2d tracks and is very close to the efficiency of the offline reconstruction. Since only the absolute z prediction of the neural network is plotted, this has to be verified by resolution plots, as the efficiency increase may be due to displaced tracks that get mapped incorrectly into the IP region (feed-down).

In Fig. 5.10 (b), the z-predictions of all found neurotracks are displayed, including tracks from possible secondary interactions. This plotting procedure is repeated in Sub. 5.3.2 and Sub. 5.3.3 in order to assess the amount of fake tracks found. For the neuro-3d tracks, an unusual peak at approximately 16 cm is observed. This peak corresponds to some 3DFinder tracks where the Neuro Trigger mistakenly creates an empty input for every input node of the neural network. Hence, a fixed prediction is observed that is purely caused by the fixed biases of every node. This anomaly will be ignored for now since this is not a problem of the 3DFinder but rather of missing inputs to the Neuro Trigger. As will be

seen in Chap. 8, with a newly defined output of the 3DFinder and a retrained network, the peak disappears.

In Fig. 5.11, the track segments of the 3DFinder are plotted for the related tracks in subfigure (a) and for all detected tracks in (b). Since no background is simulated, the

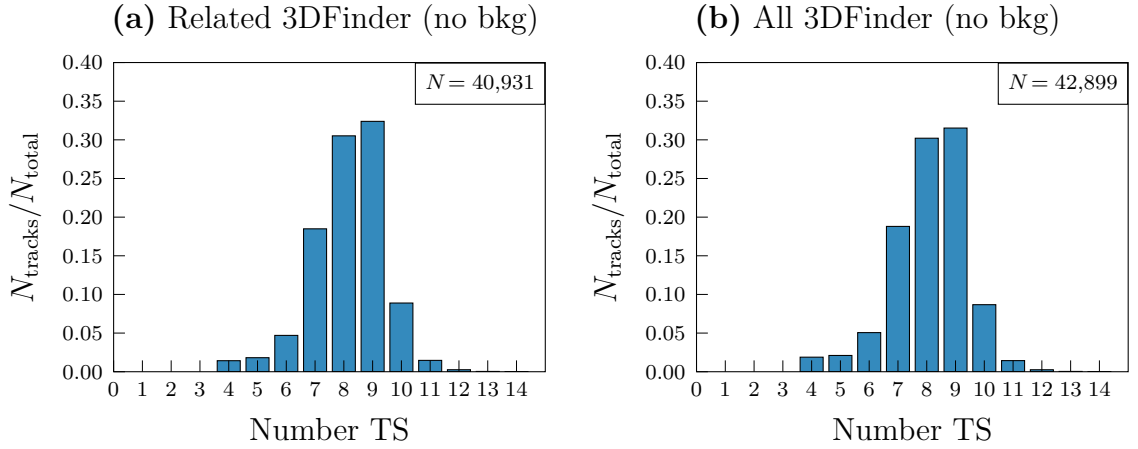


Figure 5.11: Bar plots of the number of 3DFinder track segments with no background. Subfigure (a) shows the related tracks, while subfigure (b) shows all tracks.

distributions are nearly the same. Considerably more tracks with fewer than 9 track segments are observed, which is expected because the possible phase space is now larger due to shallow θ-angles and lower transverse momentum. Furthermore, the tracks can cross the CDC starting from a displaced vertex, resulting in fewer inner super layer hits. This figure will serve as a reference when considering the simulated backgrounds later.

To verify that the z-distributions in Fig. 5.10 are accurate, correlation heatmap plots are done. In Fig. 5.12, the reconstructed z or θ is plotted against the corresponding neural network prediction, using input from either the 3DFinder or the 2DFinder. Note that the heatmaps have a logarithmic scale with a base of 10. Additionally, both z heatmaps and both θ heatmaps are normalized to the same values, respectively[5]. The expectation for a perfect resolution is a diagonal line at exactly 45°. When comparing the neuro-2d with the neuro-3d tracks, three differences are observed. While the diagonal line of the neuro-2d z-correlation spans from $(-100, -100)$ cm to $(100, 100)$ cm, the neuro-3d diagonal is noticeably shorter. This is desired because it demonstrates again that displaced tracks are not initially found by the 3DFinder. Additionally, the distributions of the neuro-3d predictions fall off considerably quicker than those of the neuro-2d tracks along the opposite diagonal. Hence, the resolution is better along this diagonal. The only problem with the neuro-3d correlation is that some tracks have random predictions all over the z- and θ-intervals. This may be due to the 3DFinder tracks that only possess 4-6 track segments. The neural network input based on the 2DFinder requires at least 3 stereo track segments and has a `minhits_axial` cut of 4, rendering those low track segment counts impossible. Furthermore, the neural network has only been trained on the 2DFinder candidates, which contain at least 7 track segments.

[5]Most heatmaps in this thesis that ought to be compared are normed to the highest value of either one.

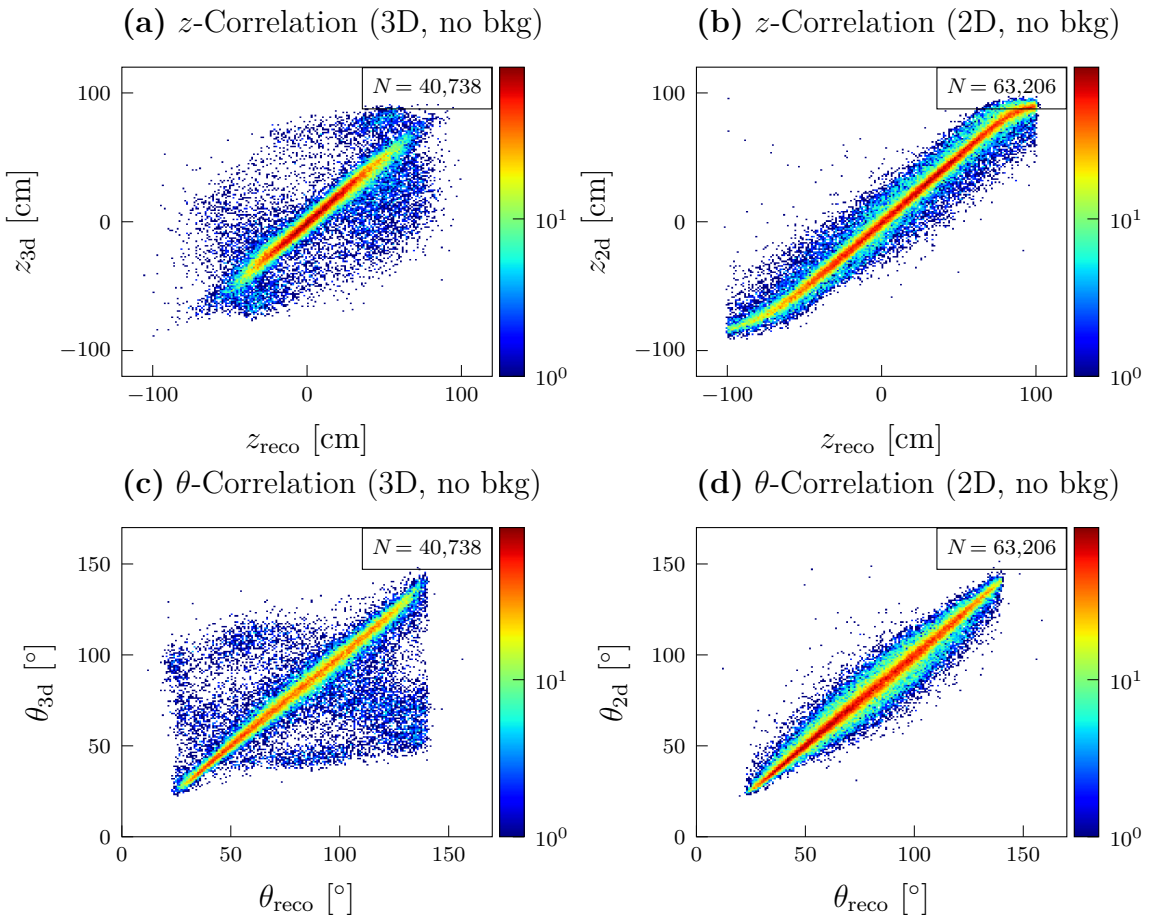


Figure 5.12: Logarithmic heatmaps comparing the reconstructed z- and θ-distribution with the respective neural network predictions with no background.

To check the resolutions and efficiencies around the IP, all reconstructed tracks within $z \in [-10, 10]$ cm are selected. If a reconstructed track was successfully related to a neuro-track, the corresponding resolution is plotted in Fig. 5.13. While the 2DFinder only has an efficiency of 85.3%, the 3DFinder possesses an efficiency of 90.6%. Thus, the better efficiency observed in Fig. 5.10 is not only due to the increased feed-down in Fig. 5.12 (a). This can be explained by tracks that have few track segments because of shallow θ-angles and low transverse momentum tracks, which cannot be found by the 2DFinder. Note that the second Gaussian is still considerably wider in the neuro-2d case and that the z-resolution is closely related to the θ-resolution. This is not surprising because an incorrect z-prediction likely causes a different θ-prediction in order to counteract this mistake in the trajectory candidate.

In conclusion, the 3DFinder outperforms the 2DFinder in efficiency, resolution, and background rejection. However, this conclusion has been reached using a data sample without background, so now simulated background will be added in the next subsections.

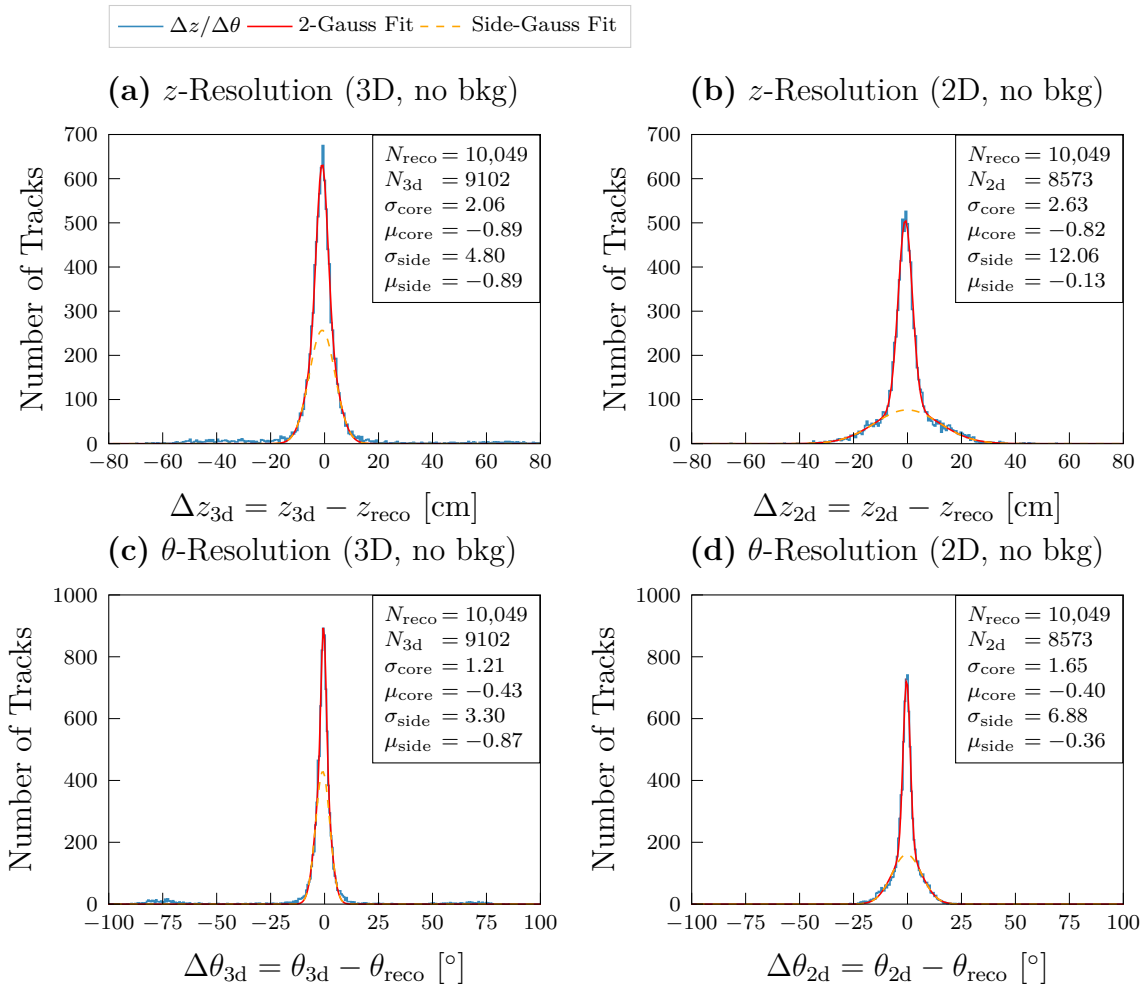


Figure 5.13: Comparison of the z- and θ-resolution for all reconstructed tracks in $z \in [-10, 10]$ cm that were successfully related to the neural network tracks with 3DFinder/2DFinder input of particle gun single tracks with no background.

5.3.2 Early Phase-3 Background

When adding early phase-3 background to the data generation, the total z-distributions in Fig. 5.14 are similar to the ones observed in Fig. 5.10. The 3DFinder is still rejecting displaced tracks but is now less efficient around the IP as is the 2DFinder. When comparing the track counts N_{3d} in the subfigures, the fake rate increases slightly. While in Fig. 5.10 this count increased by 2161 tracks, here 5491 unrelated tracks were found. More importantly, however, the number of related neuro-3d tracks decreases by 7801 when early phase-3 background is added. Although the neuro-2d track count decreased by approximately 10,000 tracks, those tracks seem to be missing mostly from the displaced tracks. Note that a cut requiring at least one track segment for the reconstructed tracks in subfigure (b) was applied. When this cut is not used, a very high peak around the IP is observed. In Sec. 5.4, the reconstructed tracks are analyzed, and an explanation for those tracks is given.

In Fig. 5.15, the 3DFinder track segment multiplicities are plotted again. There are two

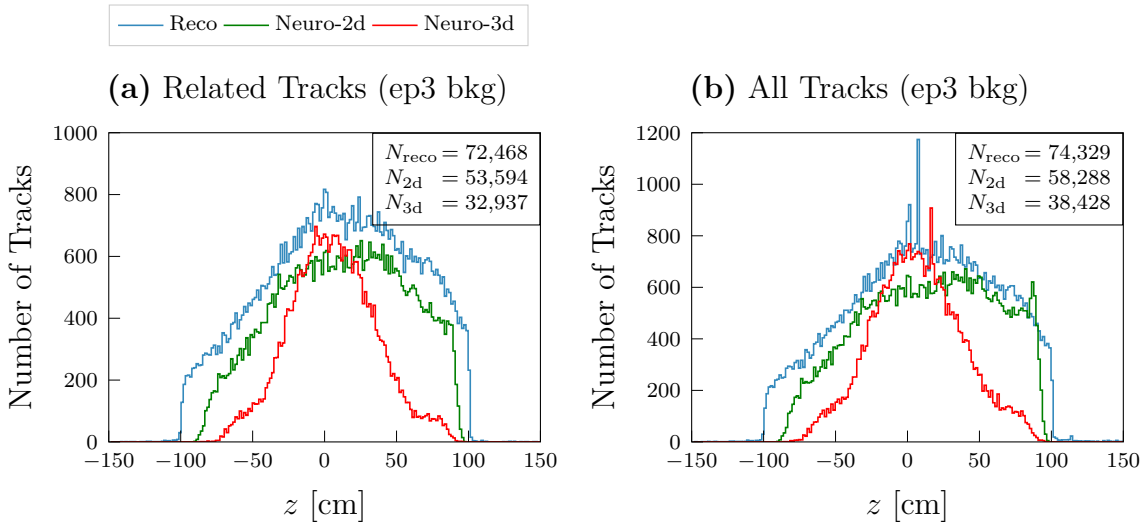


Figure 5.14: The complete z-distributions of the reconstructed, neuro-2d, and neuro-3d tracks with early phase-3 (ep3) background. Subfigure (a) shows all tracks related to a reconstructed track that was related to a particle with a production time of 0, while subfigure (b) shows all tracks with at least one track segment. Note that the y-axes of the two subfigures are scaled differently.

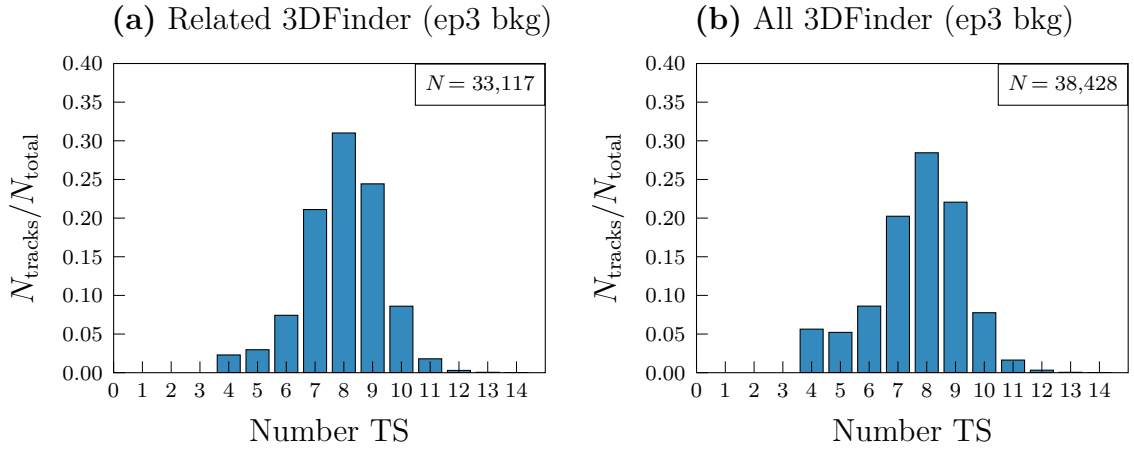


Figure 5.15: Bar plots of the number of 3DFinder track segments with early phase-3 background. Subfigure (a) shows the related tracks, while subfigure (b) shows all tracks.

notable differences to be observed. On the one hand, the track segment number of 8 has now surpassed the previously most frequent 9. Consequently, either some track segments are lost for individual tracks or tracks with lower track segment counts use background track segments to fill up empty spots. On the other hand, in subfigure (b), the number of 4 and 5 track segments more than doubles. Those are likely fake tracks that result from background only, as they could not be related to any reconstructed track.

When considering the z- and θ correlation in Fig. 5.16, the distributions are similar to the no background case. Nevertheless, some dispersion of the neural network predictions

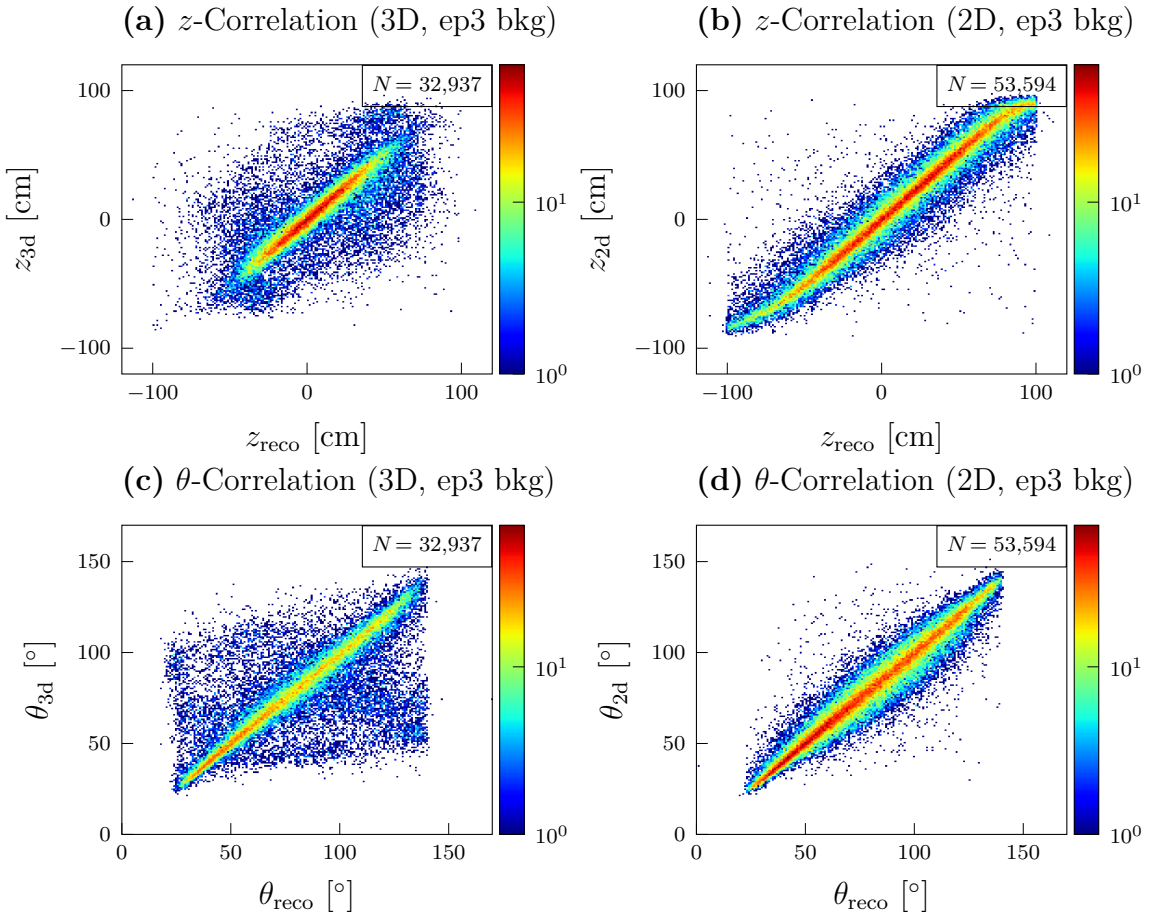


Figure 5.16: Logarithmic heatmaps comparing the reconstructed z- and θ-distribution with the respective neural network predictions with early phase-3 background.

is observed with both the 3DFinder and the 2DFinder.

In Fig. 5.17, the neural network predictions related to the reconstructed tracks originating from $z \in [-10, 10]$ cm are displayed. The 3DFinder efficiency is with 78.9% now nearly that of the 2DFinder with 78.3%. Thus, the inclusion of early phase-3 background leads to a reduction in the 3DFinder efficiency around the IP by nearly 11%, while only a decrease of 7% for the 2DFinder tracks is observed. While the neuro-3d z- and θ-resolution is still better than the one of the neuro-2d tracks, more displaced tracks and a wider double Gaussian fit are observed.

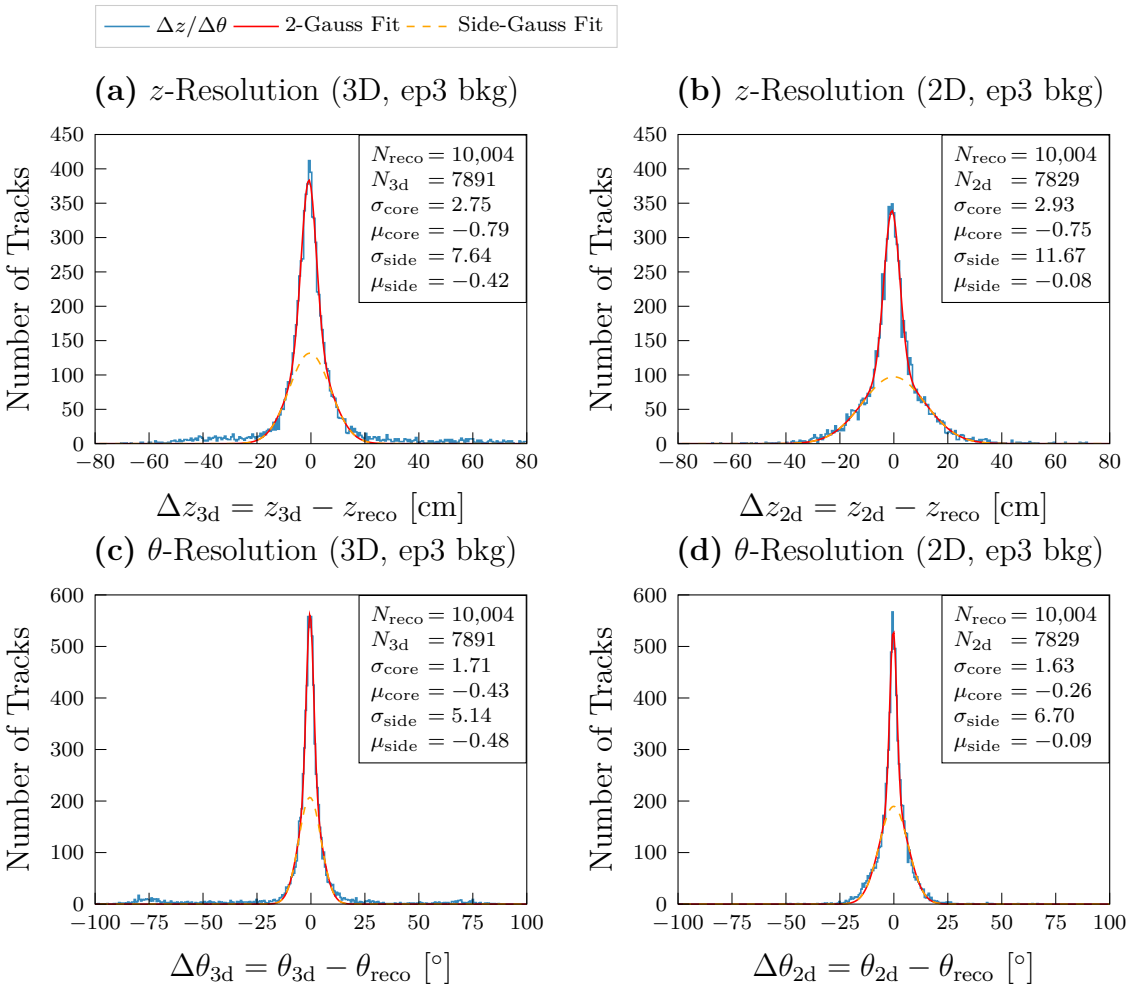


Figure 5.17: Comparison of the z- and θ-resolution for all reconstructed tracks in $z \in [-10, 10]$ cm that were successfully related to the neural network tracks with 3DFinder/2DFinder input of particle gun single tracks with early phase-3 background.

Although some efficiency loss and resolution deterioration are expected when introducing simulated background, the significant disparity already observed with the comparatively low levels of background from the early phase-3 simulation has to be addressed. Before adjusting the parameters of the 3DFinder, however, the effects of the significantly stronger nominal phase-3 background are studied in the next subsection.

5.3.3 Nominal Phase-3 Background

Fig. 5.18 (b) now clearly shows that the original 3DFinder produces a very high fake rate when nominal phase-3 background is added. While only 59,910 reconstructed tracks with a production time of 0 were found in this dataset, 195,513 3DFinder track candidates were identified. This results in a fake rate of 3.26 : 1, while the 2DFinder only has a fake rate of 1.26 : 1. Furthermore, the neuro-3d z-distribution of the related tracks in subfigure (a) indicates a significant feed-down effect into the trigger region of $z \in [-15, 15]$ cm because the peak is now higher than the one of the reconstructed tracks. Note that in subfigure (b), all reconstructed tracks are displayed for comparison. The origin of those tracks is

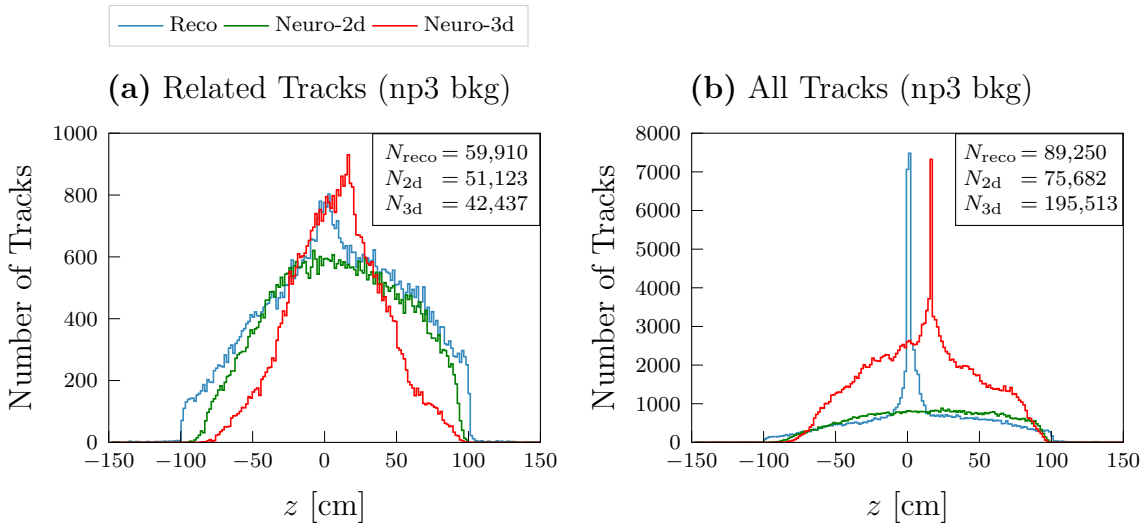


Figure 5.18: The complete z-distributions of the reconstructed, neuro-2d, and neuro-3d tracks with nominal phase-3 background. Subfigure (a) shows all tracks related to a reconstructed track that was related to a particle with a production time of 0, while subfigure (b) shows all tracks.

analyzed in Sec. 5.4.

When considering the track segment counts of the 3DFinder in Fig. 5.19, it is obvious that most of the fake tracks are tracks containing only 4-6 track segments. This has already

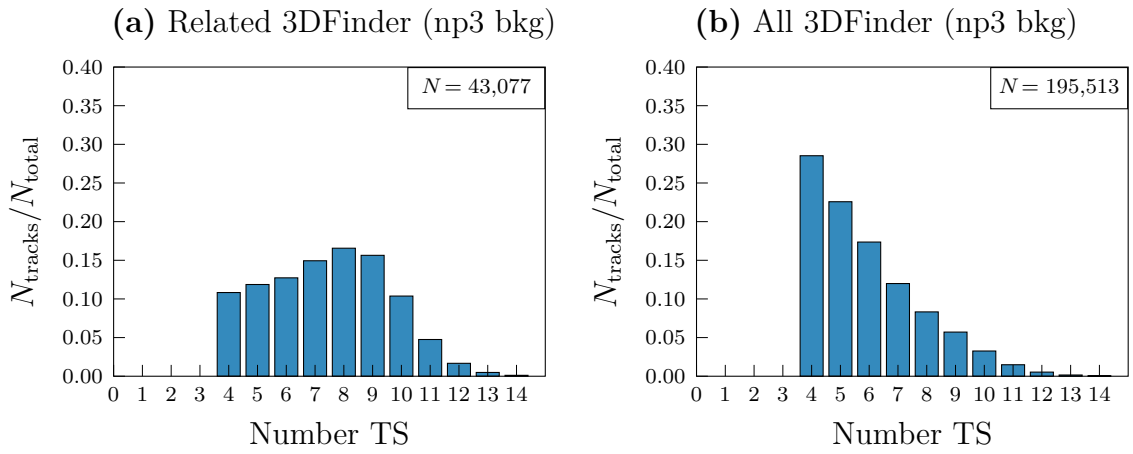


Figure 5.19: Bar plots of the number of 3DFinder track segments with nominal phase-3 background. Subfigure (a) shows the related tracks, while subfigure (b) shows all tracks.

been indicated in the early phase-3 background data. In addition, the related 3DFinder tracks show higher counts in the low track segment numbers as well. Given the abundance of background track segments in the CDC due to the nominal phase-3 background overlay, this is contrary to the expectation that otherwise empty spots on the real tracks can now be filled with background track segments. Since this is apparently not the case, this suggests that the hit-to-cluster association does not work properly, as some track segments get lost instead.

While the logarithmic heatmaps depicted in Fig. 5.20 deteriorate even more than before, which is expected, the neuro-3d resolution is considerably worse. A lot of the random

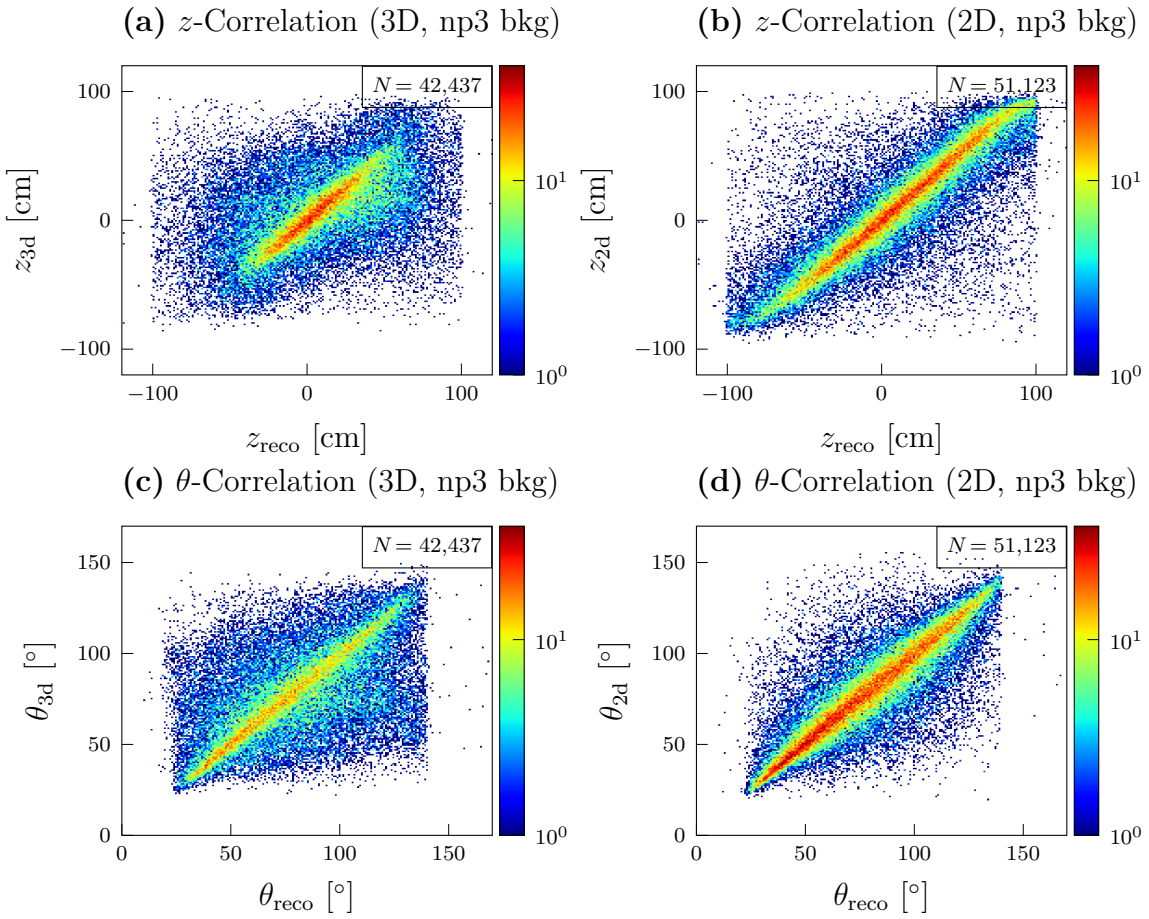


Figure 5.20: Logarithmic heatmaps comparing the reconstructed z- and θ-distribution with the respective neural network predictions with nominal phase-3 background.

neural network predictions can possibly be explained by the 4-6 track segment tracks. Such a low number of track segments is apparently not enough to properly reconstruct the z-vertex origin. It may also be possible that some genuine track segments get mistakenly replaced by background track segments. In Fig. 5.21 (a), the z-resolution of the neural network with 3DFinder input for tracks originating from $z \in [-10, 10]\,\text{cm}$ exhibits a very large second Gaussian with a standard deviation of nearly 28 cm. The neuro-3d resolutions are now worse than the neuro-2d ones. It has to be noted here again that the neural network has not been trained on such high backgrounds yet.

All in all, it is now very clear that the default parameters of the 3DFinder with `minweight` = 21 and the `shallow` hit representations are completely unusable in the L1 trigger when high backgrounds are expected. A fundamental goal of this thesis will be to improve the 3DFinder algorithm so that it can be used with such high backgrounds, as similar intensities are likely to occur in future experiments at Belle II. As the track segment plots strongly demonstrate, at least a modification of the track segment cuts `minhits` = 4 and `minhits_axial` = 0 has to be made. While this is investigated in Sec. 5.5, the

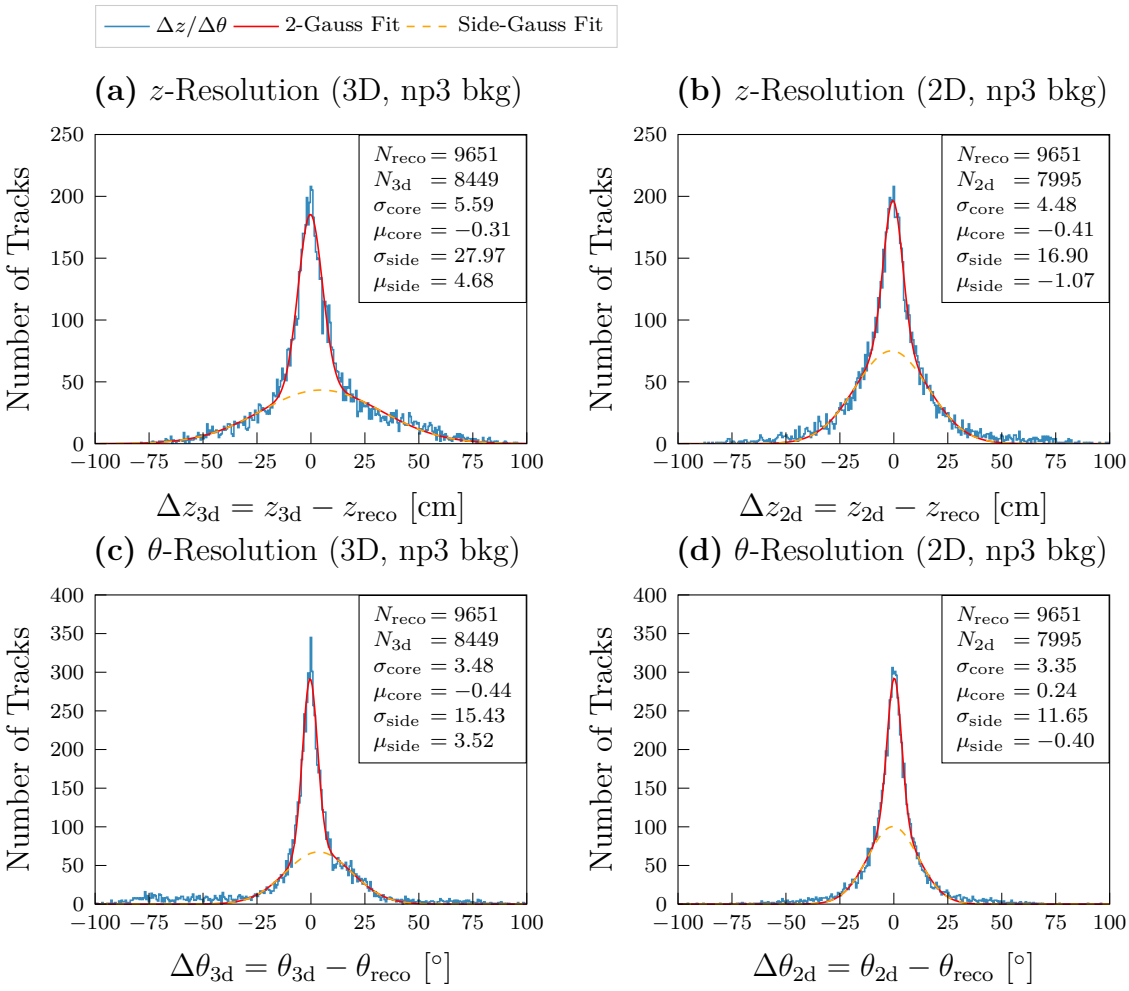


Figure 5.21: Comparison of the z- and θ-resolution for all reconstructed tracks in $z \in [-10, 10]$ cm that were successfully related to the neural network tracks with 3DFinder/2D-Finder input of particle gun single tracks with nominal phase-3 background.

reconstructed tracks of the early and nominal phase-3 datasets are analyzed first in Sec. 5.4.

5.4 Analysis of the Reconstructed Tracks

When conducting Monte Carlo studies, the initial parameters of the tracks are given precisely. For example, a cut on the production time of all the tracks can be applied, which eliminates all tracks of secondary particles. Since such a cut is not possible when dealing with real data (as in Chap. 8), the reconstructed tracks of the two previous subsections are analyzed. Moreover, the absence of any CDC dead spots in the detector simulation can be verified.

In Fig. 5.22, the distribution of all reconstructed tracks is compared with the distribution of reconstructed tracks that have at least one associated track segment. This cut alone removes nearly all the excess IP tracks that are generated by the simulated background.

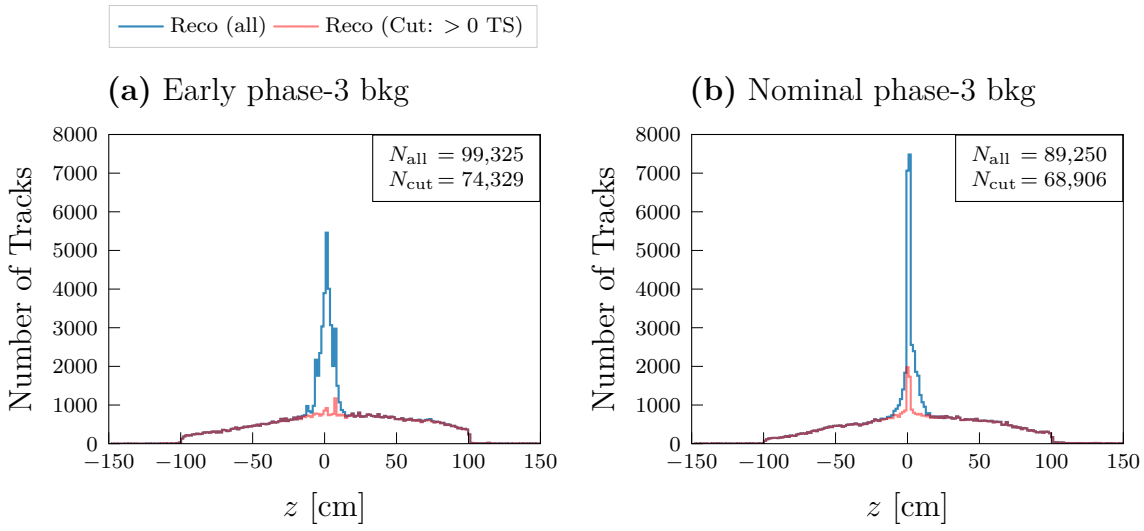


Figure 5.22: The z-distributions of the reconstructed tracks with both early and nominal phase-3 background. The blue plot represents all reconstructed tracks while, the red plot only includes reconstructed tracks with at least one related track segment.

When considering the nominal phase-3 background, there is still a small peak in the distribution. Thus, most of the excess track must be reconstructed by VXD instead of the CDC.

Fig. 5.24 confirms that nearly all of those zero track segment tracks have a very low transverse momentum p_{T}. Hence, the momentum is not large enough, causing the tracks to curl back before reaching the CDC in such a way that at least one track segment gets activated. The small peak observed in subfigure (f) that is still left after the track segment cut has low momentum as well. Those tracks do not penetrate the CDC deep enough to be able to find them with the trigger algorithms that demand at least four track segments. As can be seen in Fig. 5.24, most of those tracks originate from the forward region of the detector. In conclusion, it will be necessary to introduce track segment cuts on the reconstructed tracks when analyzing the efficiency of the trigger algorithms on real data.

In order to check for dead spots in the CDC, the reconstructed ϕ-angle is analyzed. Regarding Fig. 5.25 (a), the reconstructed ϕ is plotted against the corresponding reconstructed θ. The dataset consists of Monte Carlo single particle gun tracks originating from the IP, i.e., $z \in [-1, 1]\,\text{cm}$, that are contained in the full CDC acceptance range. Since the observed distribution is very uniform, there are no dead spots in the simulation. In subfigure (b), only the reconstructed ϕ is plotted from the large signal dataset of Sub. 5.3.1, where wide angles and displaced tracks are contained as well. This confirms again that the CDC is assumed to be fully functional in the basf2 simulation.

5.5 Introduction of Track Segment Cuts

As the introduction of nominal phase-3 background clearly indicated, cuts in the number of track segments could significantly reduce the number of fake tracks. In the default parameters, only `minhits` = 4 are required for the total number of track segments in a cluster.

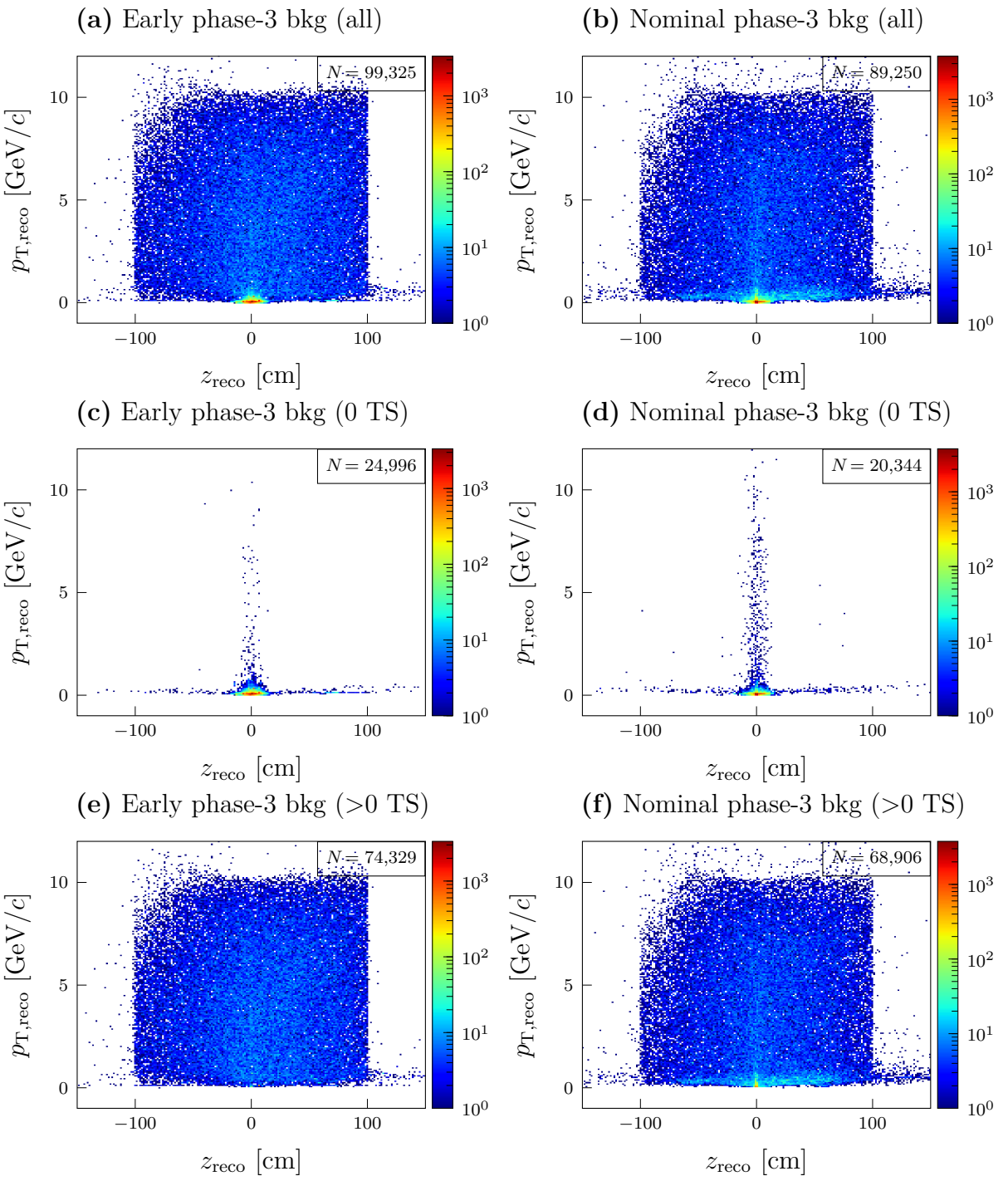


Figure 5.23: Logarithmic heatmaps comparing the reconstructed z-distribution with the respective p_T-distribution with both early and nominal phase-3 background. Plots (c) and (d) show the distributions where no track segment was related to the reconstructed track, while plots (e) and (f) show the distribution where at least one track segment was found.

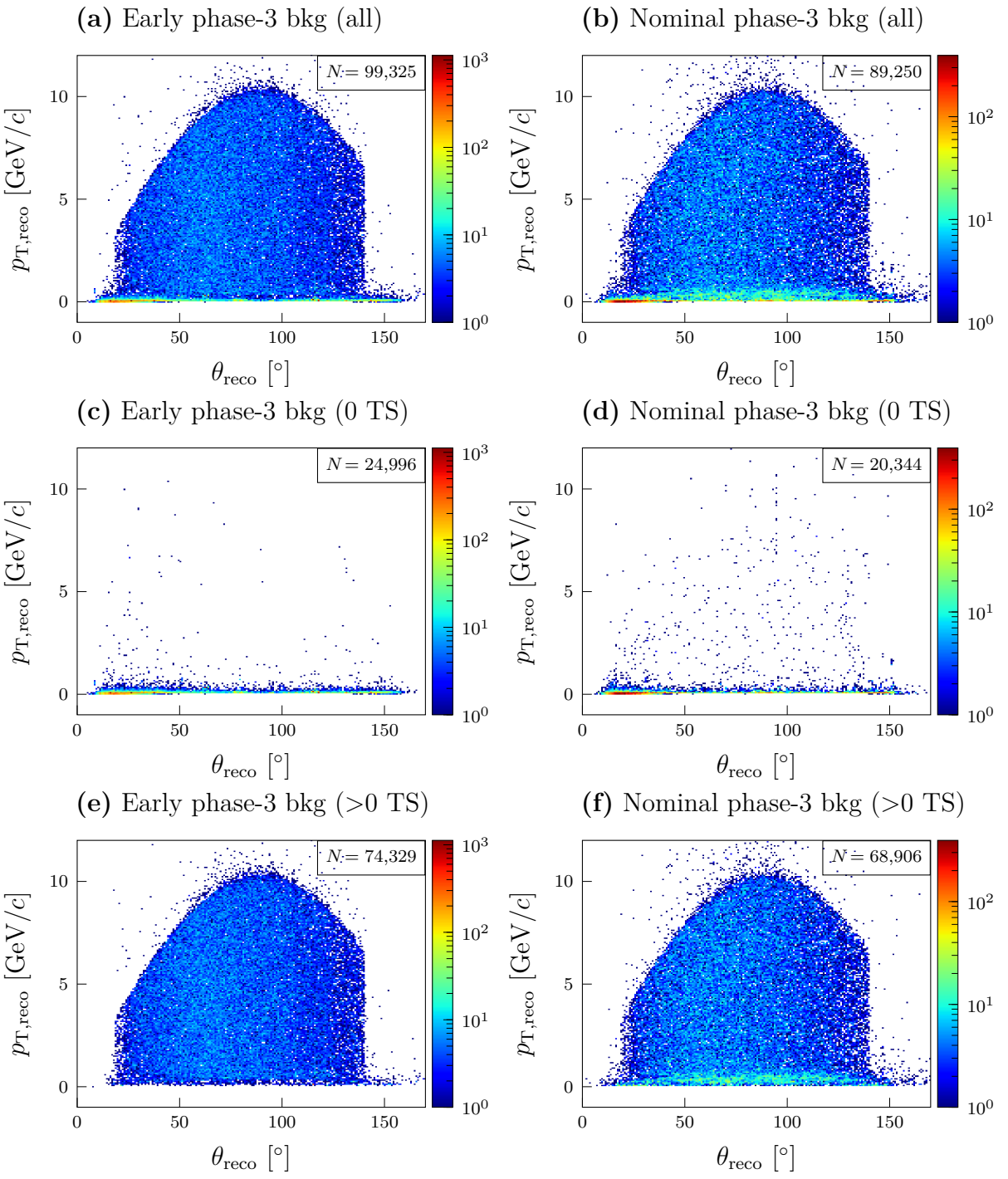


Figure 5.24: Logarithmic heatmaps comparing the reconstructed θ-distribution with the respective p_T-distribution with both early and nominal phase-3 background. Plots (c) and (d) show the distributions where no track segment was related to the reconstructed track, while plots (e) and (f) show the distribution where at least one track segment was found.

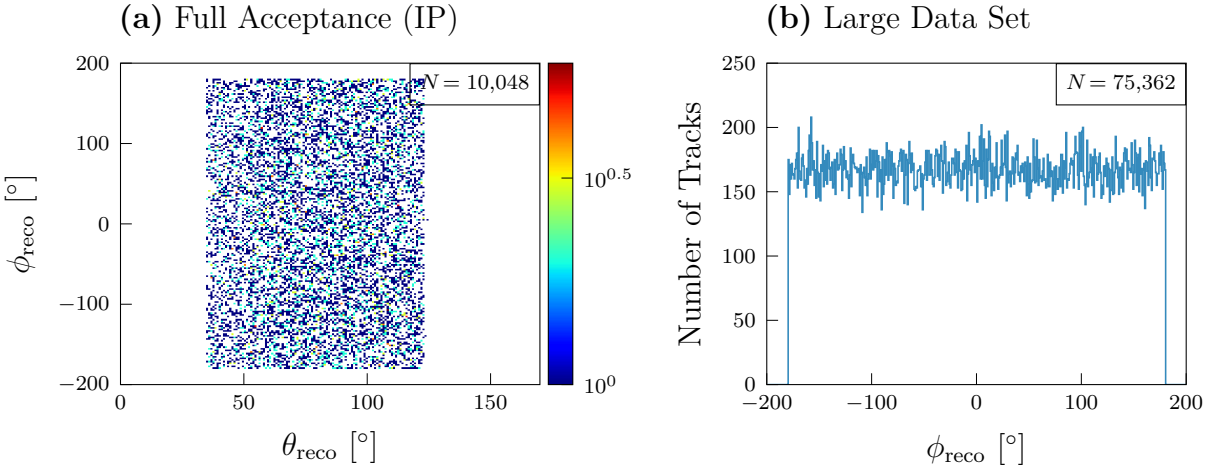


Figure 5.25: Subfigure (a): Logarithmic heatmaps comparing the reconstructed θ-distribution with the respective ϕ-distribution without background. Subfigure (b): Reconstructed ϕ-distribution of the large dataset.

Hence, a first attempt can be made by increasing `minhits` and using the `minhits_axial` cut. The test of those parameter combinations is conducted on IP tracks within the full CDC acceptance range with a nominal phase-3 background overlay (see Tab. 5.7). A total

Table 5.7: Parameters of the particle gun single tracks for the full CDC acceptance range.

Parameter	Range	Distribution
z	$[-1, 1]$ cm	uniform
θ	$[35, 123]°$	uniform in $\cos(\theta)$
ϕ	$[-180, 180]°$	uniform
p_{T}	$[0.35, 6]$ GeV/c	uniform

of 12,981 reconstructed tracks and 11,051 neuro-2d tracks were found while generating 10,000 particle gun tracks.

In Fig. 5.26, four different combinations of track segment cuts are displayed. Here, "mh" is an abbreviation for `minhits` and "ma" for `minhits_axial`. For each 3DFinder setting, the complete neuro-3d z-distribution is displayed. Since the Monte Carlo tracks originate from the IP, those z-distributions are approximately the Δz-resolutions. With the default `minhits` = 4, a total of 25,193 tracks are found, and a very broad distribution is observed. When increasing `minhits` to 7, more than half of those tracks get removed, resulting in much smaller tails while still being efficient around the IP. As the track segment cuts become stricter, the tails decrease in size, resulting in fewer fake tracks found by the 3D-Finder.

In Fig. 5.27, the z-resolution for `minhits` = 4 is compared to the `minhits` = 7 and `minhits_axial` = 4 one. While only 8623 neuro-3d tracks with the stricter track segment cuts could be related to the reconstructed tracks compared to the 9846 of the `minhits`

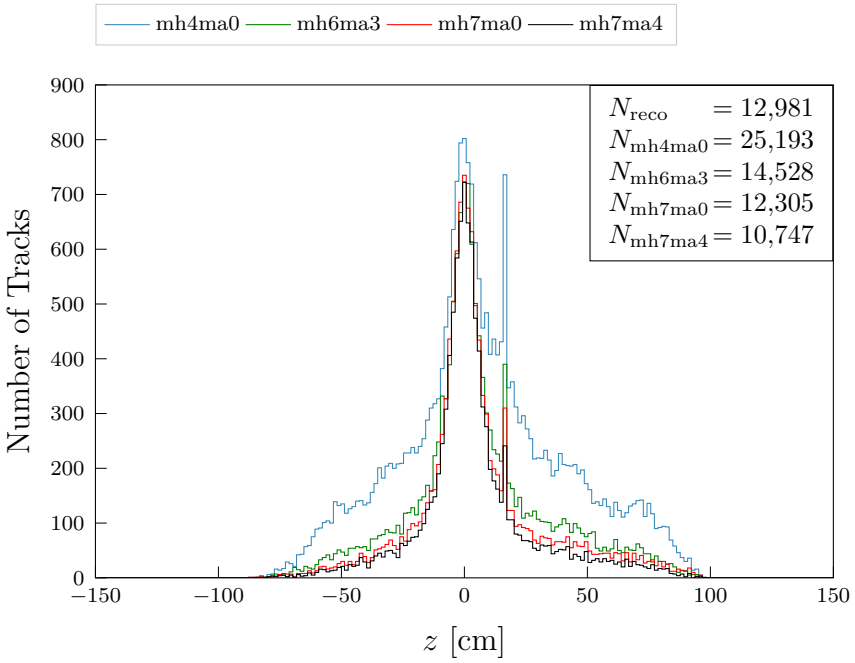


Figure 5.26: The complete z-distributions of the neuro-3d tracks of single particle gun tracks originating from $z \in [-1, 1]$ with nominal phase-3 background for different track segment cuts.

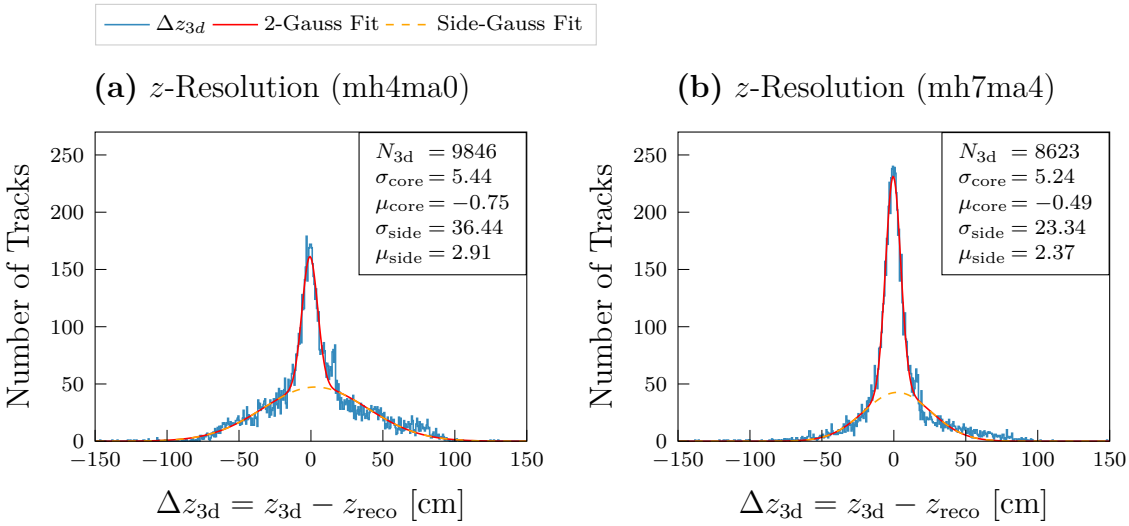


Figure 5.27: Comparison of the z-resolution for different track segment cuts for single particle gun tracks originating from $z \in [-1, 1]$ with nominal phase-3 background.

= 4 cut, the maximum amplitude does increase significantly. Furthermore, the standard deviation of the side Gaussian is notably smaller. The increase in the amplitude strongly indicates that the hit-to-cluster association is not working properly. Apparently some track segments of the real tracks get lost when the fake rate is high, explaining the bad resolution in subfigure (a) compared to (b). It is important to note here again that the neural network has neither been trained on high backgrounds nor with 3DFinder track candidates yet. Nevertheless, the efficiency and resolution are still worse than when using the 2DFinder as a comparison.

When comparing the track segments found by the 3DFinder with those used by the neural network, a big disparity is observed. In Fig. 5.28 (a), the number of 3DFinder track segments is plotted against the neuro-3d z-prediction, while in subfigure (b), the track segments of the neuro-3d tracks are plotted against the same z-prediction. Note that the

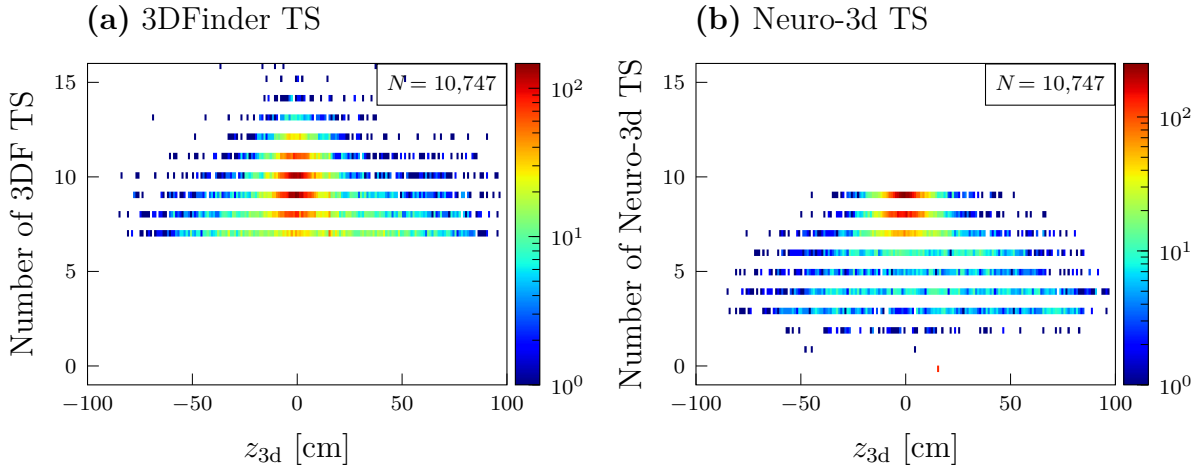


Figure 5.28: Logarithmic heatmaps comparing the z-prediction and the number of track segments with nominal phase-3 background.

dataset using the `minhits` = 7 and `minhits_axial` = 4 cuts is displayed. Due to the `minhits` = 7 cut, the minimum number of track segments in subfigure (a) is 7, while in (b), a lot of tracks with 6 or fewer track segments are observed, causing the bad resolution as expected. Thus, the `minhits` cut does not guarantee an efficient cut since duplicate track segments in the same super layer are counted individually. As can be seen in the later chapters, both real and simulated backgrounds are predominately activating track segments next to each other in the same super layer. Since the neural network can only use one track segment per super layer and can therefore not gain any new information about those duplicate track segments, such a `minhits` cut is questionable.

In conclusion, the problem of the fake tracks cannot be sufficiently solved with the `minhits` parameters due to efficiency loss and bad resolutions. There seems to be a problem in the hit-to-cluster association algorithm that must be addressed. Furthermore, the original DBSCAN clustering algorithm cannot be implemented in hardware at the L1 trigger due to its complexity and non-determined execution time. Therefore, a statistical cluster analysis is conducted in the next chapter. Understanding the cluster shapes and sizes allows for the construction of a new clustering algorithm (see Chap. 7) and a new hit-to-cluster association (see Chap. 8) suitable for the hardware implementation.

Chapter 6

Hough Space and Detailed Cluster Analysis

The analysis of the clusters within the Hough space is conducted for various reasons. Firstly, this allows for an understanding of the observed cluster shapes for different kinematic trajectories and for the two different hit representations. Those clusters can be compared with the clusters identified by the DBSCAN algorithm in order to understand its weaknesses. Furthermore, by generating a statistical analysis of the average cluster shapes, insights for designing a new clustering algorithm that meets the latency requirements of the L1 trigger can be derived. Lastly, a comparison of the real clusters with the nominal phase-3 fake clusters may allow for an additional way to suppress fake tracks.

6.1 Visualization of the Hough Space

The complete Hough space is binned in a $40 \times 384 \times 9$ three-dimensional matrix, which is necessary for the hardware application. For each event, this matrix can be extracted and displayed either in a two-dimensional heatmap for a specified θ-bin or with a three-dimensional heatmap where the cluster cells are displayed. In this chapter, simulated Monte Carlo events are used for the analysis. This allows for the creation of specific tracks with and without background overlays where the track kinematics are exactly known. The first example are single muon tracks of both charges. In Fig. 6.1, an excerpt of the Hough spaces surrounding the intersection of the corresponding hits is displayed. The Monte Carlo parameters in all four subfigures were fixed to $z_{\mathrm{MC}} = 0\,\mathrm{cm}$, $\theta_{\mathrm{MC}} = 80°$, $\phi_{\mathrm{MC}} = 100°$, and $p_{\mathrm{T,MC}} = 0.5\,\mathrm{GeV}/c$. This specific choice was made for two reasons: On the one hand, a comparison of the two different hit representations can be made by using the exact same parameters for the muon. In the two Hough spaces representing the same charged muon, all 8 track segment IDs were identical, and no duplicates were found. On the other hand, the cluster shapes and positions can be assessed when only the charge of the muons is altered. In subfigures (a) and (b), the muon is positively charged, while in (c) and (d), it is negatively charged. As described in Sub. 5.1.1, the Hough planes intersect at the upper half of the ω-axis for positively charged muon tracks, whereas for negatively charged particles, the intersection occurs in the lower half. Hence, all traces of the track segments forming the cluster caused by a physical track in the CDC should have a similar shape, i.e., a tilt toward the upper right corner.

When considering the two different hit representations in Fig. 6.1, a few notable observations emerge. The curves of the 8 track segments seem to be in very similar positions in both Hough spaces. While the `comp` hit representation appears to have a very similar weight contribution for each plane, the weights of the same hits in the `shallow` representa-

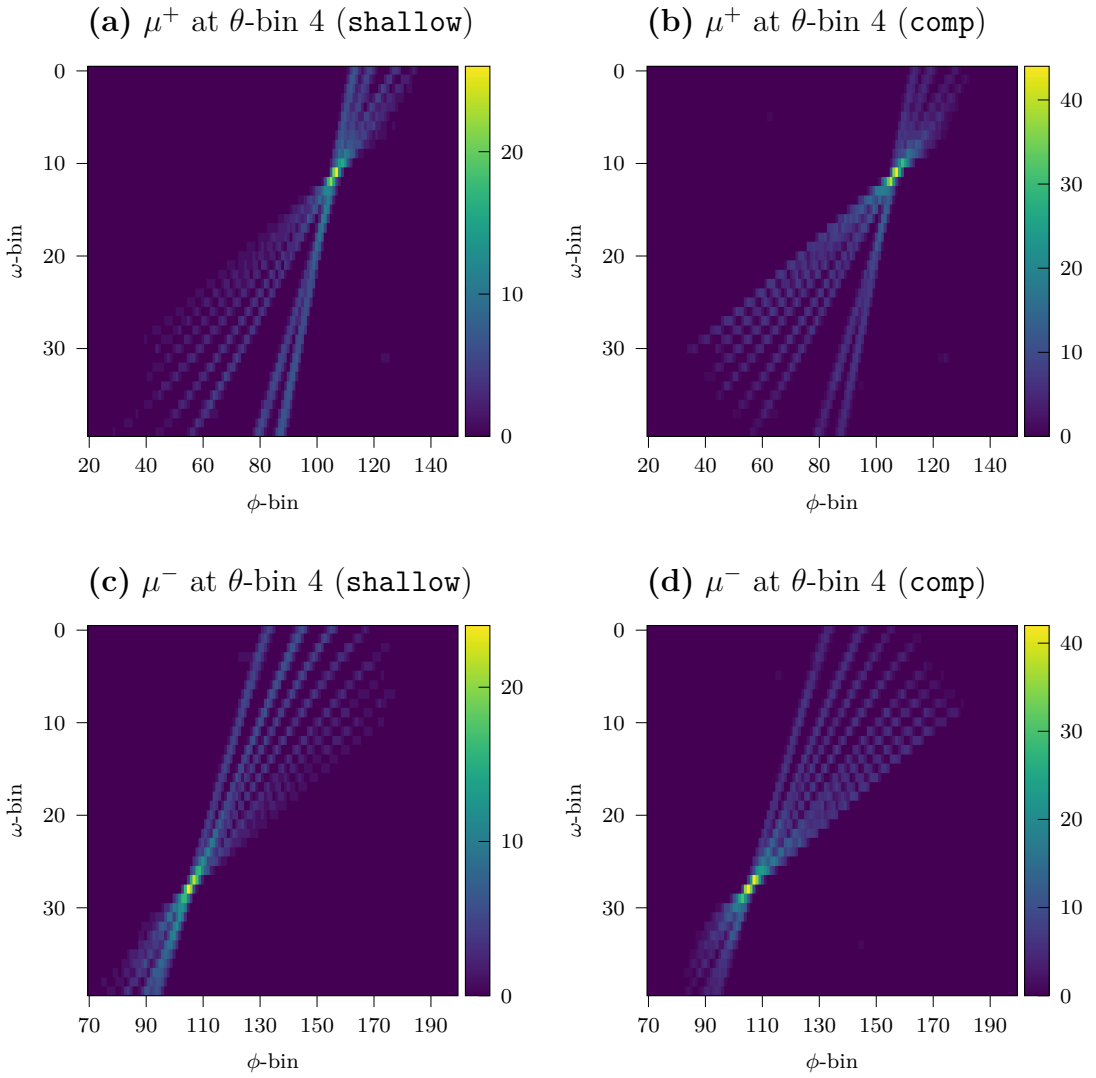


Figure 6.1: Heatmaps of the Hough space for θ-bin = 4 of a signal particle gun track. The only difference in the track parameters is the charge of the muon. While plots (a) and (b) use the shallow hit representations, (c) and (d) use the comp hit representations. The 3DFinder tracks of the same charge had an association with exactly the same track segments.

tions are dependent on the slope of the Hough plane. The steeper the plane, the higher the observed weight contribution to the Hough space. A steeper slope indicates a more precise estimation of the ϕ-angle. Therefore, those planes stem from the inner track segments, which restrict the possible phase space more severely in the ϕ-dimension. When an outer super layer is hit, a lot of different track curvatures are allowed, where each curvature corresponds to a significantly different ϕ emission angle. For an inner super layer, the phase space for the track curvatures is just as large; the ϕ-angle is similar for each of those curvatures, however.

In Fig. 6.2, excerpts of the Hough space displayed in Fig. 6.1 are illustrated. The weight of

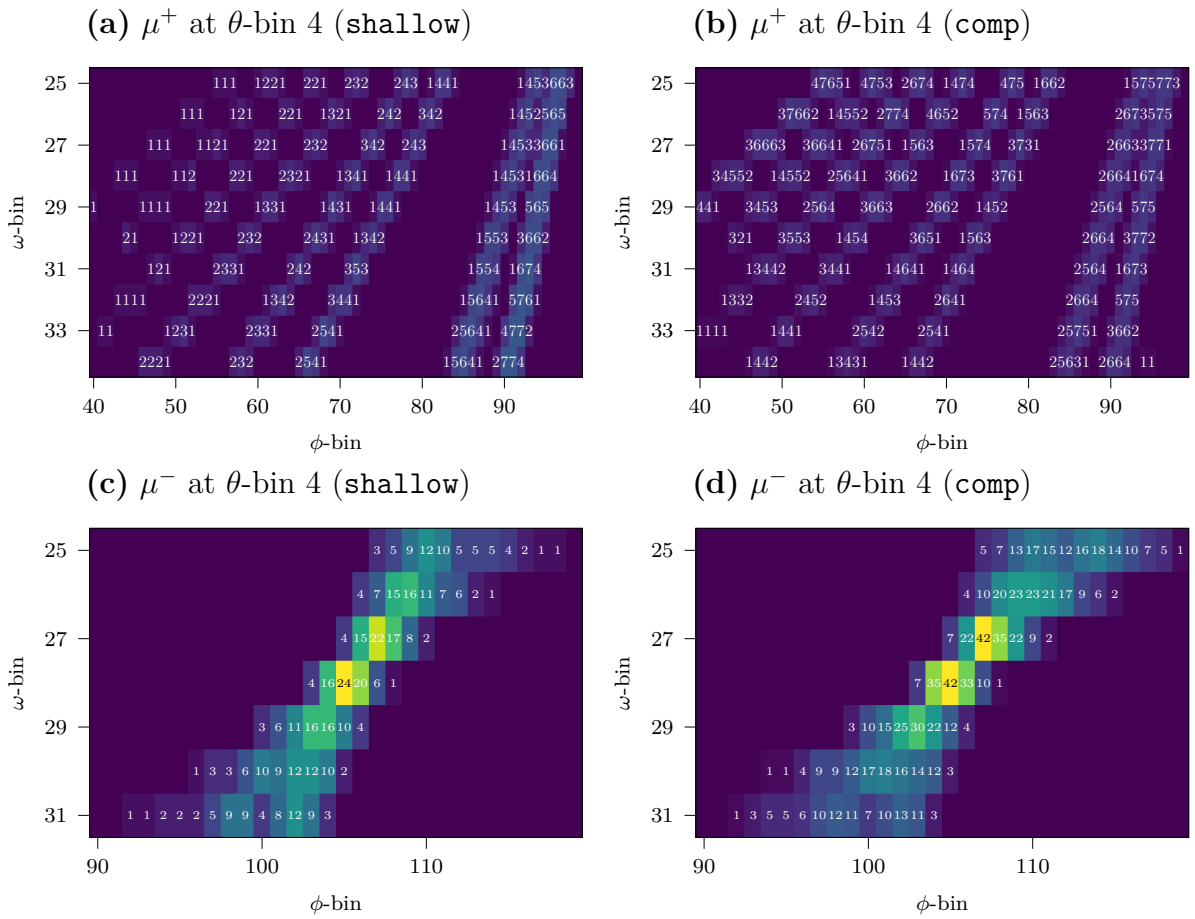


Figure 6.2: Comparison of the Hough space weights for the two different hit representations. The 3DFinder tracks of the same charge had an association with exactly the same track segments.

each cell is written into the heatmap. Note that a cell weight of 0 is omitted in this figure for clarity. Subfigure (a) demonstrates clearly the large difference in weight contribution for each super layer of the `shallow` hit representations. Super layer 8 only contributes weights of one, while the three outermost super layers have individual cell weights that do not exceed two. Such small weight contributions seem to be negligible and may explain the efficiency problems observed in Chap. 5. In comparison, the `comp` representations exhibit a very similar weight contribution in each track segment for the various super layers. The

maximum weight of 7 is observed in every track segment (3 bits are chosen for each Hough cell).

In subfigures (c) and (d), the clusters around the peak weight are depicted. The most notable difference is in the peak weight itself. While the peak for the 8 track segment trajectory is only 24 for the `shallow` hit representations, a peak weight of 42 is observed for the `comp` representations. When using a `minweight` of 21, only 2 cluster cells were found by the DBSCAN algorithm when using the `shallow` hit representations for both muon charges. In contrast, the clusters of the `comp` hit representations contained 44 (μ^+) and 42 (μ^-) cluster cells, respectively. All four clusters are illustrated in Fig. 6.3 in a three-dimensional cluster plot, where the color of each cuboid represents the corresponding cell weight. When using `minweight` = 21, apparently both hit representations have a problem.

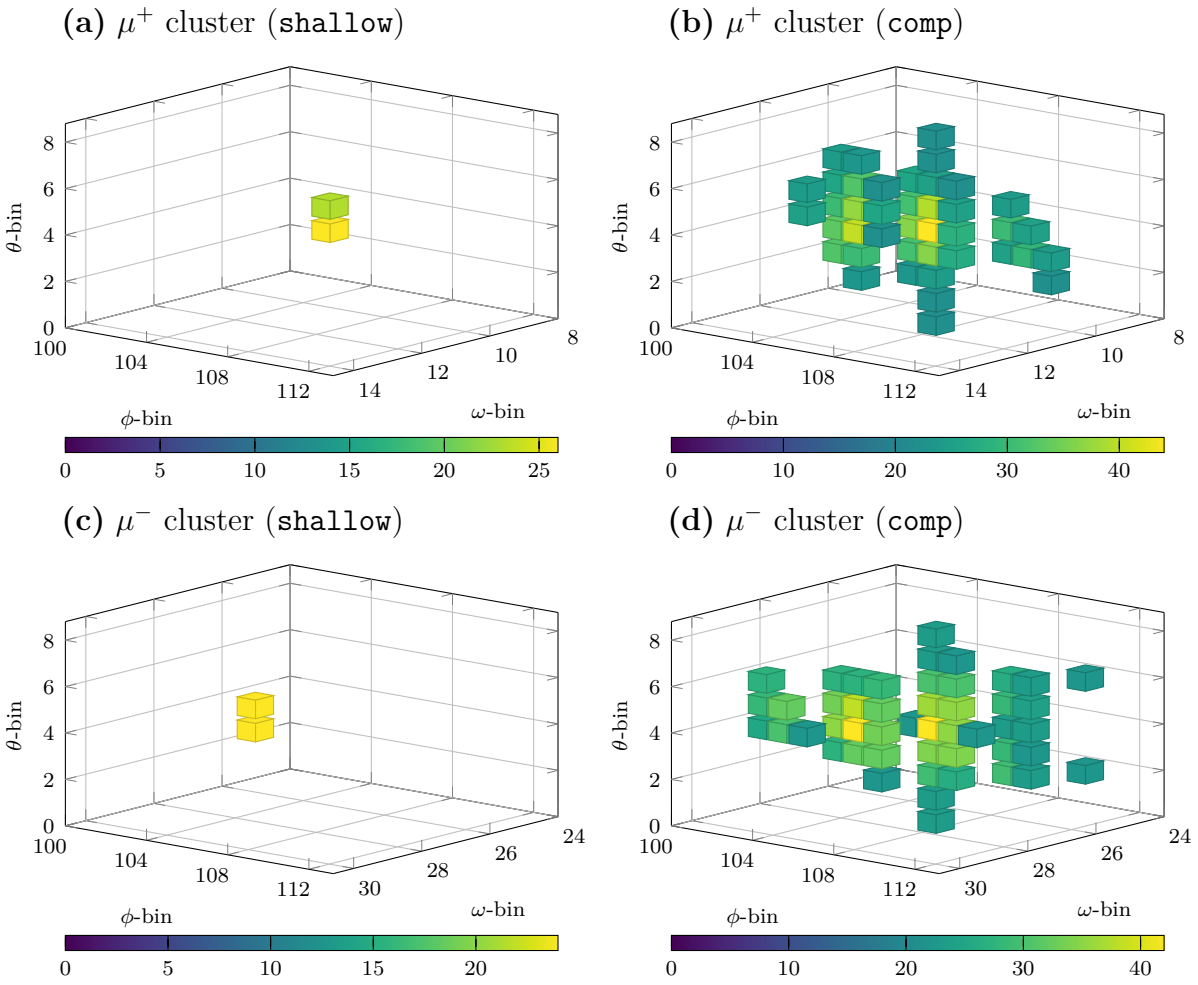


Figure 6.3: Clusters of the four single particle gun tracks of Fig. 6.1 with different charges and hit representations. Note that `minweight` = 21 was used.

With 8 track segments found, a cluster of only 2 cells with a maximum weight of 24 seems to be problematic. Hence, low-momentum and shallow-θ particles, activating only 6 or 7 track segments, may not be found when using the `shallow` representations. Additionally, dead spots, especially in the inner CDC layers, may contribute to further loss of track-finding efficiency. Conversely, the clusters of the `comp` hit representations get so large that they

span over all 9 θ-bins in the Hough space. This is of no use for the parameter estimation because the `thresh` parameter only considers cells with a weight of at least 85% of the peak weight. Furthermore, the track segments contributing to all of those distant cluster cells are considered in the hit-to-cluster association, which may include incorrect hits. In Chap. 5, an inefficient association of the track segments to the clusters was observed when using those hit representations.

To get an impression of the cluster shapes when considering different track curvatures, six single muons with a varied transverse momentum were created. Note that the `comp` representations were used in this sample. With $p_{\mathrm{T,MC}} \in [0.3, 0.375, 0.5, 0.75, 1.5, 10.2]\,\mathrm{GeV}/c$, the curvature change in this trajectory sample is approximately linear. In Fig. 6.4, the DBSCAN clusters of $p_{\mathrm{T}} = 0.3\,\mathrm{GeV}/c$ and $p_{\mathrm{T}} = 0.75\,\mathrm{GeV}/c$ are displayed. Due to the low

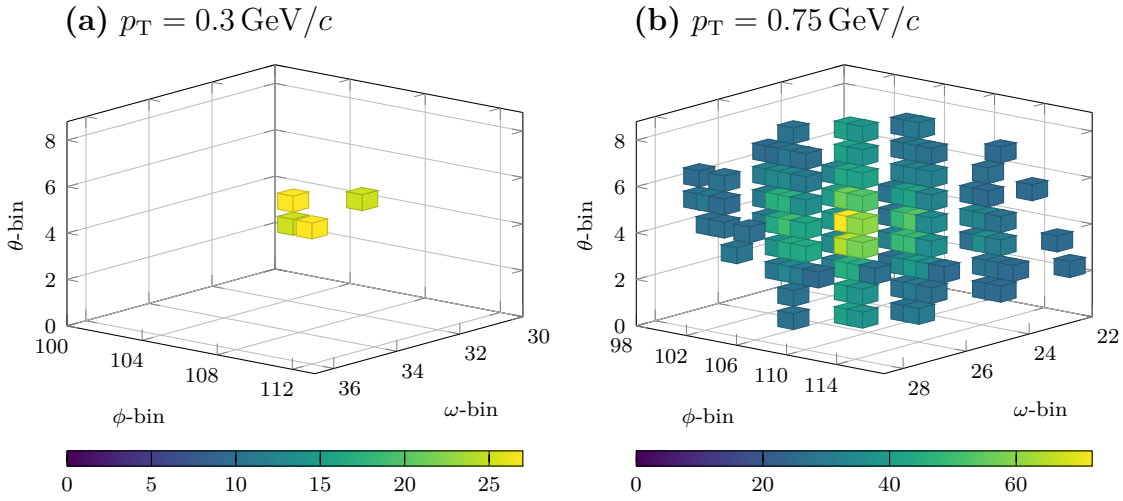


Figure 6.4: Clusters of the six single particle gun tracks with different transverse momentum using the `comp` hit representations. Note that `minweight` = 24 was used.

momentum, only 6 track segments were found in subfigure (a), resulting in a cluster of only 4 cells with a peak weight of 27. Conversely, in subfigure (b), all 9 super layers produced a total of 11 track segments, resulting in a cluster with 107 cells with a peak weight of 72. The two extra track segments are duplicates in super layers 5 and 8. Hence, very large discrepancies in the clustering and, consequently, the execution time have already been observed. Note that those tracks do not contain any background, as the inclusion of background may enlarge the clusters even more. Furthermore, a `minweight` of 24 was used again as 21 was found to be too small when using the `comp` hit representations. In Fig. 6.5, the corresponding Hough space intersections of all 6 momentum parameters are illustrated. The clusters exhibit an equidistant shift along the ω-axis as a result of their linear change in curvature. As the cluster shapes remain approximately constant, a statistical cluster analysis to determine the average cluster shape and size is appropriate.

In such a statistic, fake clusters from nominal phase-3 background can be analyzed as well. To get an idea of how the Hough space changes when adding simulated background, two Hough spaces are illustrated in Fig. 6.6. In subfigure (a), a fake track was created from pure background alone. In contrast, subfigure (b) includes a single particle gun track. Note that both heatmaps are normed to the same value in order to assess the severity of

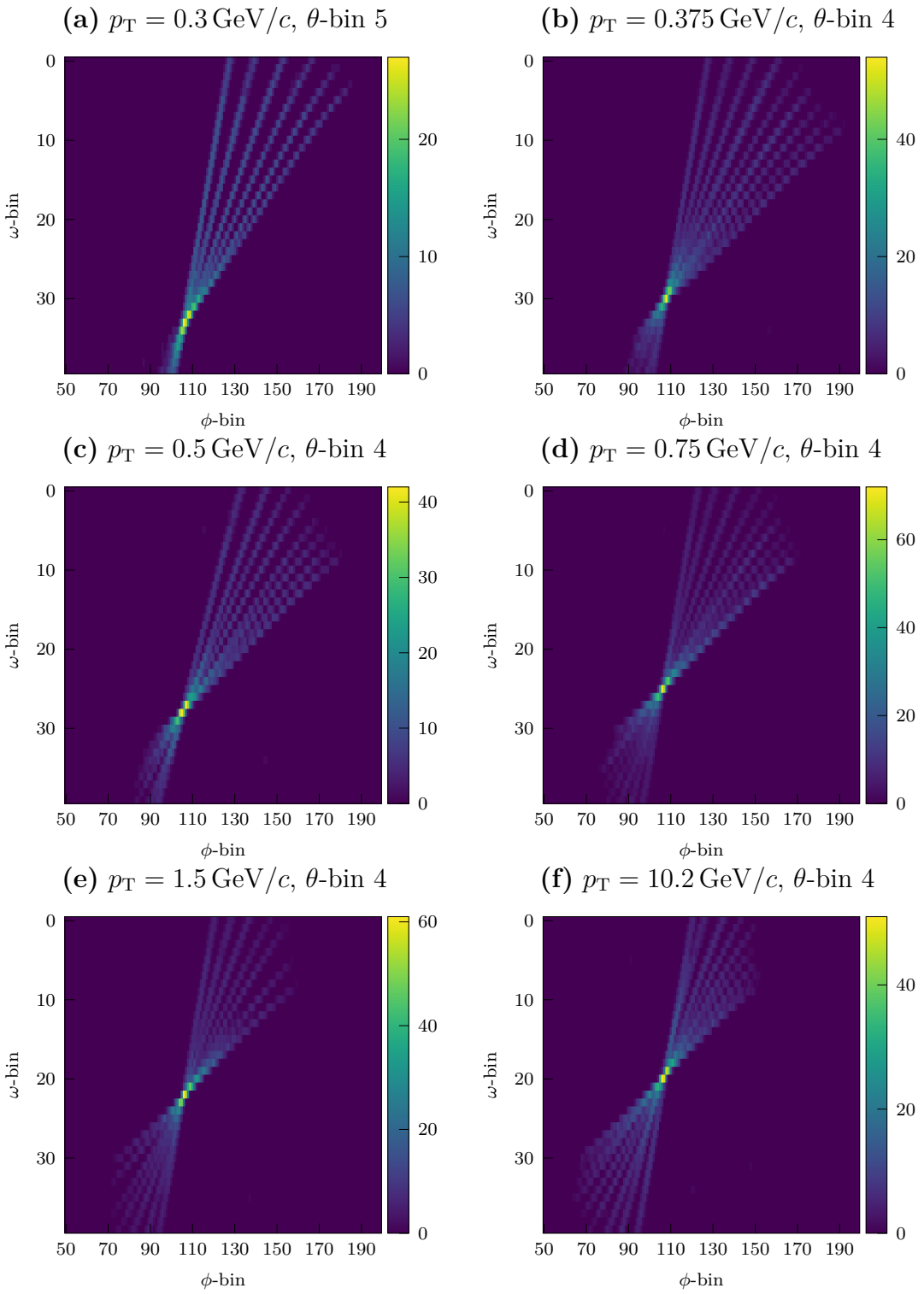


Figure 6.5: Heatmaps of the Hough space for the maximum θ-bin of a signal particle gun track using the `comp` hit representations. The transverse momentum p_T was varied uniformly with respect to its inverse.

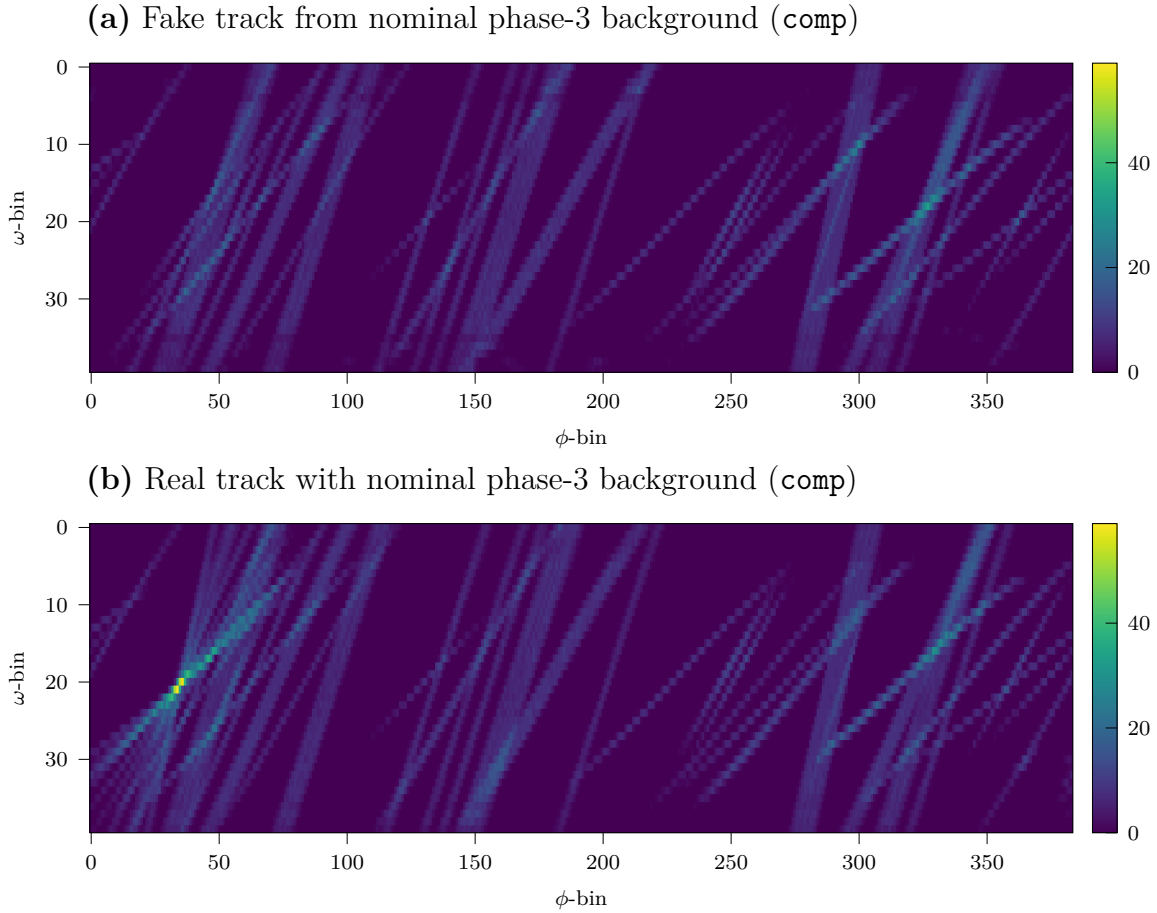


Figure 6.6: Comparison of the Hough space from a real and a fake track with nominal phase-3 using the `comp` hit representations. Both heatmaps are normalized to the maximum value of subfigure (b) to make them comparable.

the background peaks. Numerous random hits throughout the entire ϕ-axis are observed. When comparing the background peaks with the real clusters, an important difference is evident. Since a real trajectory crosses nearly all super layers, the Hough planes fan out on both sides of the intersection. This is not observed in the background-only Hough space. Here, many track segments are very close to one another in the same super layer, creating thick bands throughout the whole Hough space with a fixed slope. This may be useful when differentiating between real and fake tracks during clustering.

6.2 Cluster Statistics

For the cluster statistics, six datasets of 10,000 tracks each were created: Signal tracks without background, fake tracks produced by using only nominal phase-3 background, and signal tracks with background overlay. Each set of tracks was analyzed using both hit representations. The datasets are listed in Tab. 6.1. Note that the signal tracks are primary Monte Carlo muons found by the offline reconstruction, which were successfully related to a 3DFinder track. Those particle gun tracks originate from the IP and are contained within the full CDC acceptance range. The parameters were the same as in

Table 6.1: Data samples for the cluster statistics with nominal phase-3 background.

Sample	Representation	`minweight`	# Tracks
Signal Tracks (no bkg)	`shallow`	21	10,000
Signal Tracks (no bkg)	`comp`	24	10,000
Fake Tracks (only bkg)	`shallow`	21	10,000
Fake Tracks (only bkg)	`comp`	24	10,000
Signal Tracks (with bkg)	`shallow`	21	10,000
Signal Tracks (with bkg)	`comp`	24	10,000

Tab. 5.7. A different `minweight` was used for either hit representation as their weight contributions are distinct. The data was generated until a total of 10,000 tracks were found, i.e., the different efficiencies and fake rates were counteracted.

For the statistical analysis of the clusters, two methods are used. The first method is used to assess the DBSCAN performance, which can be compared to the "real" clusters later. In this method, all cluster cells found by the density-based scanning algorithm are counted. Note that this is heavily dependent on the choice of the `minweight` parameter. The analysis is carried out as follows: All the Hough space weights and their position in the Hough space found by the already implemented DBSCAN cluster algorithm are stored in a list. For one track or cluster, the position corresponding to the largest weight is defined as the cluster center, i.e., $(0, 0, 0)$, and all other cell positions in this cluster are now relative to this center. Thus, it is possible to compare all clusters together by counting a contribution of 1 if the cell was a cluster cell in one track. Trivially, the total count for the cell $(0, 0, 0)$ is 10,000. Using this, one can plot an average cluster where each cell has a weight corresponding to how often it was on average a cluster member in a track. In the following plots, different cuts on the number of total memberships are introduced. Only cells that have more tracks than the cut percentage times 10,000 are left in the plots. For example, a cut of 15% would mean that a cell in the plot needs at least 1,500 occurrences in the data sample. A heatmap is used to color the cells according to their relative frequency in the sample.

The second method examines the specific weights contributing to the clusters. This is only indirectly dependent on a clustering algorithm because the tracks have to be found by the DBSCAN algorithm in the first place. Using the maximum index, i.e., the position of the peak weight, found by the DBSCAN clustering algorithm, a $9 \times 9 \times 9$ matrix is centered around this index in the Hough space. For every cell index in this matrix, the corresponding weight in each of the 10,000 tracks is summed up. This total weight is divided by 10,000 to obtain the average weight for each cell around the maximum. Note that the peak cell index in the $9 \times 9 \times 9$ matrix is defined as $(0, 0, 0)$. As in the DBSCAN method, the color of each cell is determined by a heatmap describing the average weight.

6.2.1 Statistics of the `shallow` Hit Representations

In Fig. 6.7, the average DBSCAN clusters are depicted for the `shallow` hit representations. As expected from the cluster plots in Sec. 6.1, the average cluster size is small despite using

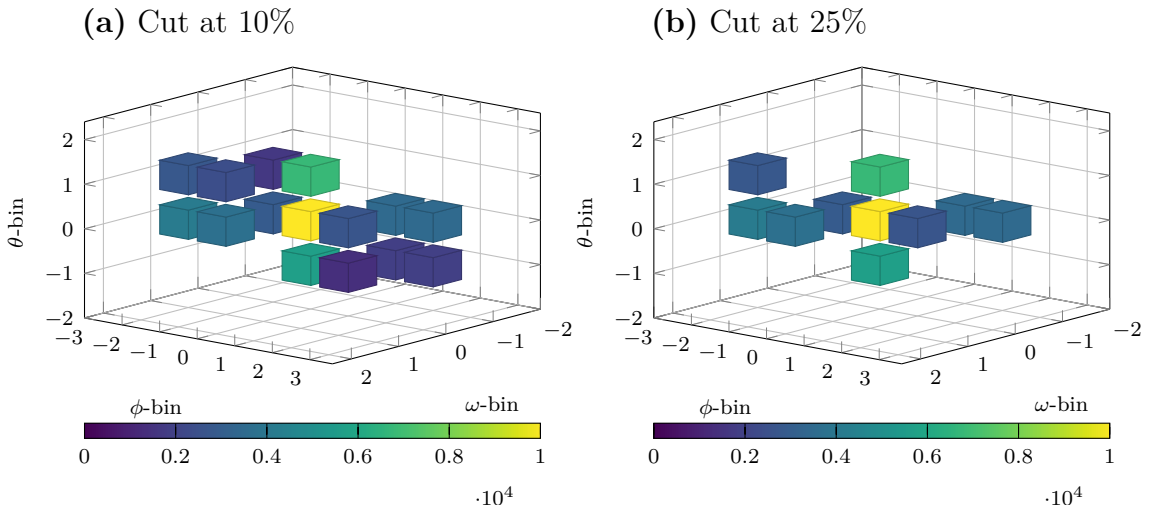


Figure 6.7: Statistics on the density-based clustering (DBSCAN) algorithm using signal tracks without background. Note that the `shallow` hit representations with `minweight` = 21 were used.

a `minweight` of 21. In subfigure (a), all DBSCAN cells that appeared in at least 10% of all clusters are displayed. When increasing this cut to 25% (subfigure (b)), the average cluster is essentially just in one θ-bin except for the two bins right above and below the cluster maximum. It is expected that these two bins are likely to be found since there are only 9 θ-bins. Hence, it is probable that a trajectory possesses a θ-value right in between the ones defined by the bins. Note that the data sample only includes tracks within the full CDC acceptance range with sufficient transverse momentum ($p_\mathrm{T} > 350\,\mathrm{MeV}/c$). When incorporating low-momentum tracks and shallow θ-angles, the number of cluster cells would decrease even further. This may lead to a more inaccurate track parameter prediction and hit selection.

In Fig. 6.8, the fake clusters are displayed. While some clusters can possibly get very large, as can be observed in subfigure (a), the average cluster in subfigure (b) appears to be very similar to the signal-only case.

Since the assumption of background-free clusters is unrealistic, real clusters with background overlays are displayed in Fig. 6.9. The clusters are considerably larger on average compared to the signal tracks. As observed in Fig. 6.6, a lot of fake hits are scattered randomly throughout the whole Hough space. A real cluster may be either above or in proximity of such a fake Hough plane and include those weights in the clustering.

The DBSCAN clusters must now be compared with the actual weights constituting those clusters. As before, three different datasets are used. In Fig. 6.10, all cells surrounding the maximum with an average weight of at least 14 (subfigure (a)) or of at least 18 (subfigure (b)) are displayed. The cells are very similar to the ones found by the DBSCAN clustering in Fig. 6.7. Leaving aside the maximum in the center, the average weights of all other cluster cells are below 25, despite using the full acceptance range of the CDC. This explains why the efficiency problems in Chap. 5 were fixed by decreasing the `minweight` parameter to 21.

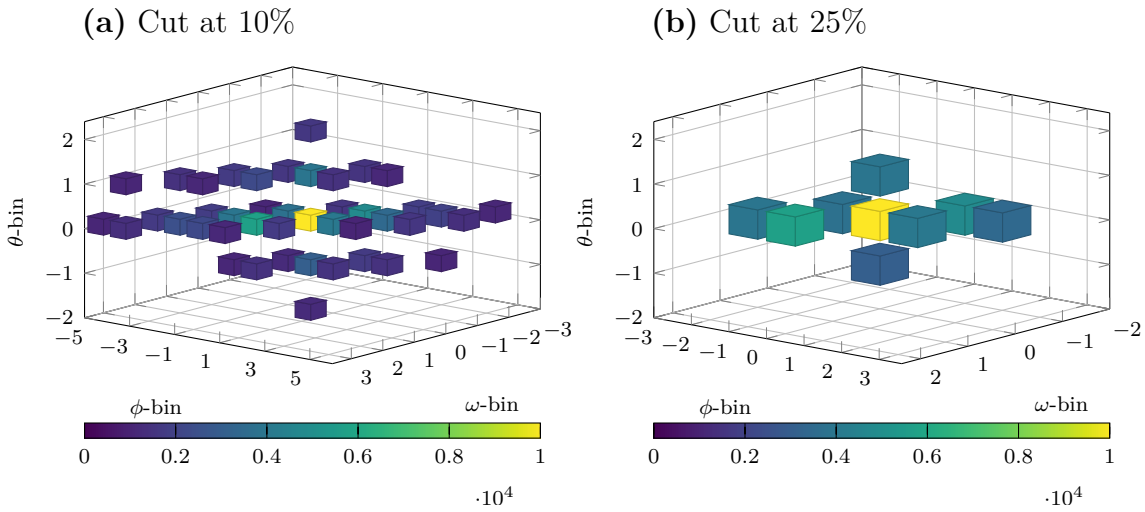


Figure 6.8: Statistics on the density-based clustering (DBSCAN) algorithm using fake tracks from nominal phase-3 background only. Note that the `shallow` hit representations with `minweight` = 21 were used.

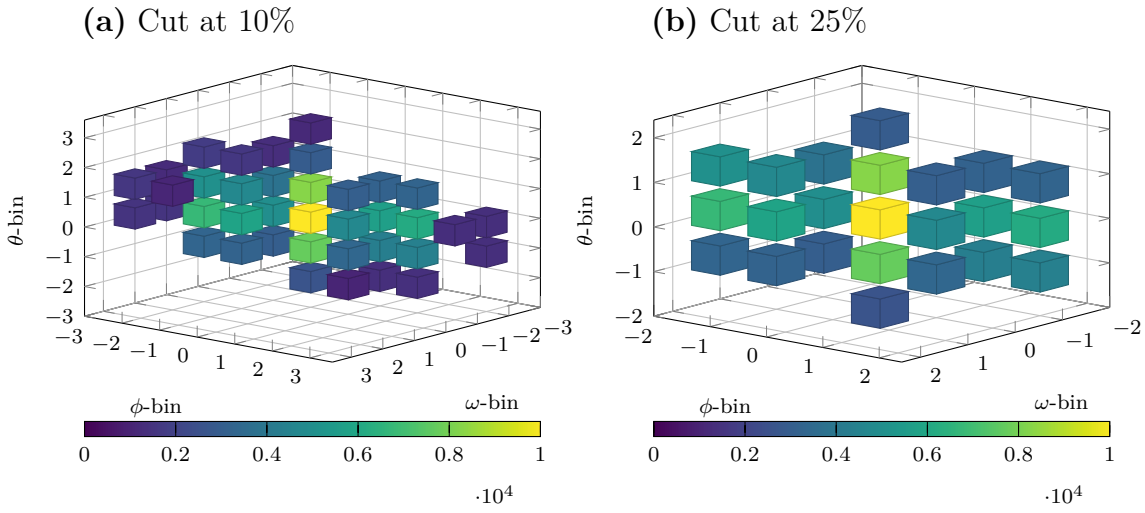


Figure 6.9: Statistics on the density-based clustering (DBSCAN) algorithm using signal tracks with nominal phase-3 background. Note that the `shallow` hit representations with `minweight` = 21 were used.

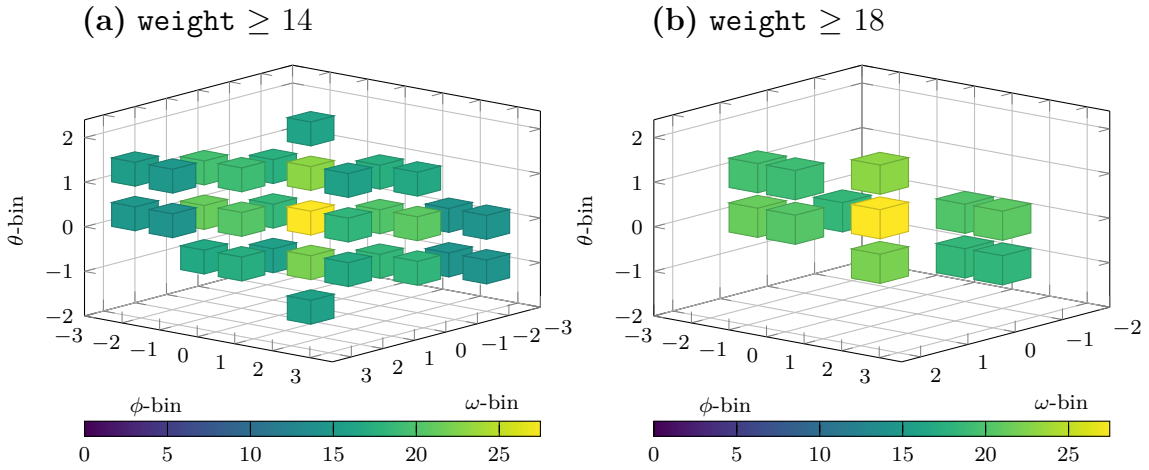


Figure 6.10: Visualization of the average cluster cell weights using signal tracks without background. Note that the `shallow` hit representations with `minweight` = 21 were used.

When comparing the real cluster averages with the fake cluster averages in Fig. 6.11, a significant difference is observed. The weights spread out considerably more in the θ-bin

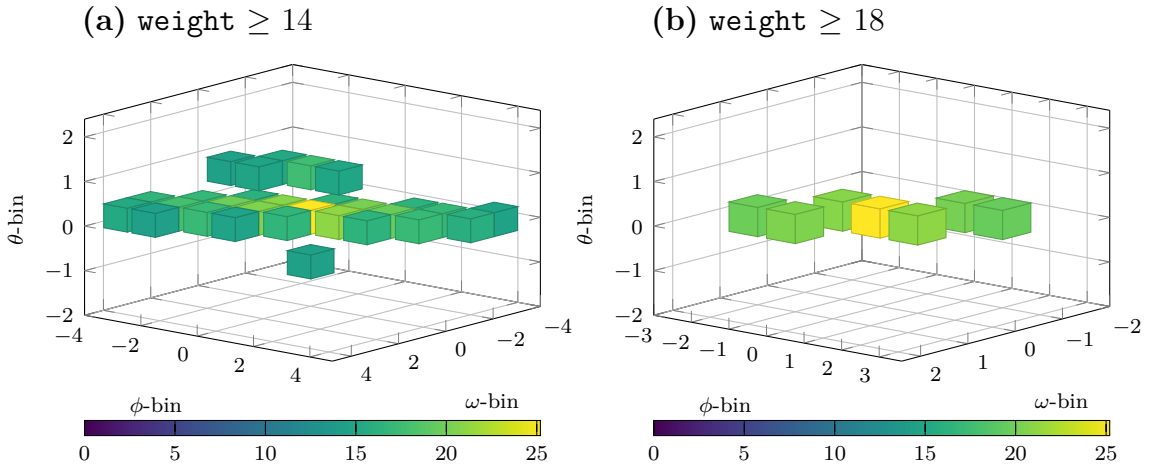


Figure 6.11: Visualization of the average cluster cell weights using fake tracks from nominal phase-3 background only. Note that the `shallow` hit representations with `minweight` = 21 were used.

of the peak cell. Such a difference in the weight distribution may be used to differentiate fake clusters from real clusters.

When including background to the real tracks, the average weights get considerably larger, which is illustrated in Fig. 6.12. The difference between the fake clusters and the real clusters with background is now even more significant. The θ-layers above and below the peak index show a comparable total weight to the layer in the middle.

In Fig. 6.13, the average cluster weights around the maximum weight are displayed for signal tracks with background overlay (subfigure (a)) and fake tracks (subfigure (b)). It is interesting to observe that the average peak weight of 29.6 for the signal clusters with

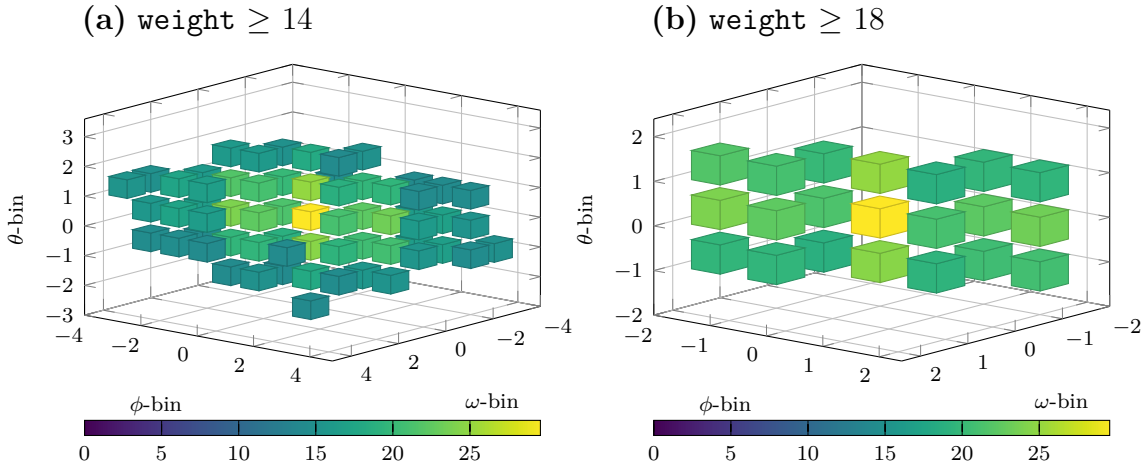


Figure 6.12: Visualization of the average cluster cell weights using signal tracks with nominal phase-3 background. Note that the `shallow` hit representations with `minweight` = 21 were used.

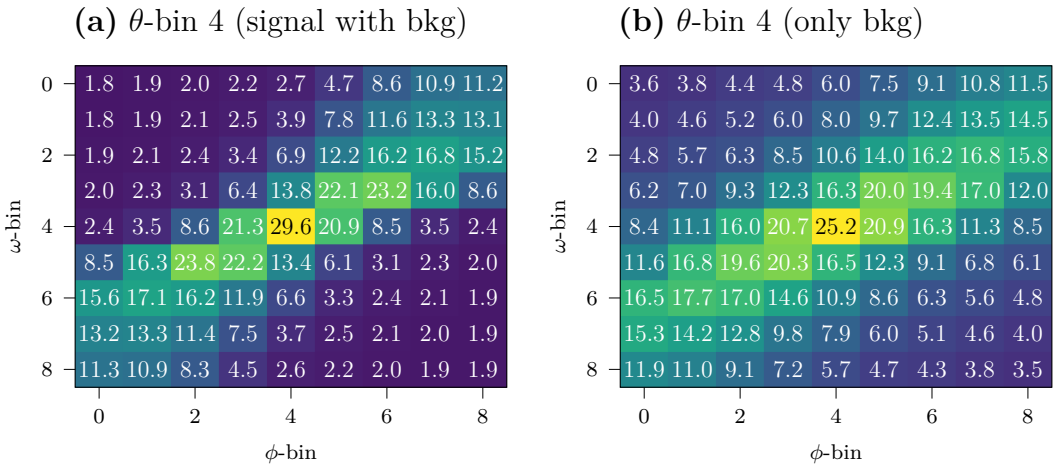


Figure 6.13: The average Hough space weights around the cluster peak, comparing signal tracks with background with nominal phase-3 background only fake tracks. Note that the `shallow` hit representations with `minweight` = 21 were used.

background overlay is only slightly larger than for the fake tracks with 25.2. This can be explained by the unequal weight distribution for every super layer. As the background appears to be confined to the innermost 3 super layers only[1] (see Fig. 6.6), large fake clusters are created when the background is activating track segments in the inner super layers. This seems to be problematic, especially when recalling the weight contributions of only 1–2 per cluster cell for the outer super layers in Fig. 6.2. One may be able to use the weight distribution to differentiate between the real and fake clusters, however. The cell weights of the real clusters fall off considerably quicker than in the fake clusters.

6.2.2 Statistics of the `comp` Hit Representations

The same analysis is done for the `comp` hit representations. In Fig. 6.14, the average DBSCAN clusters are displayed for real tracks without any background. Note that the

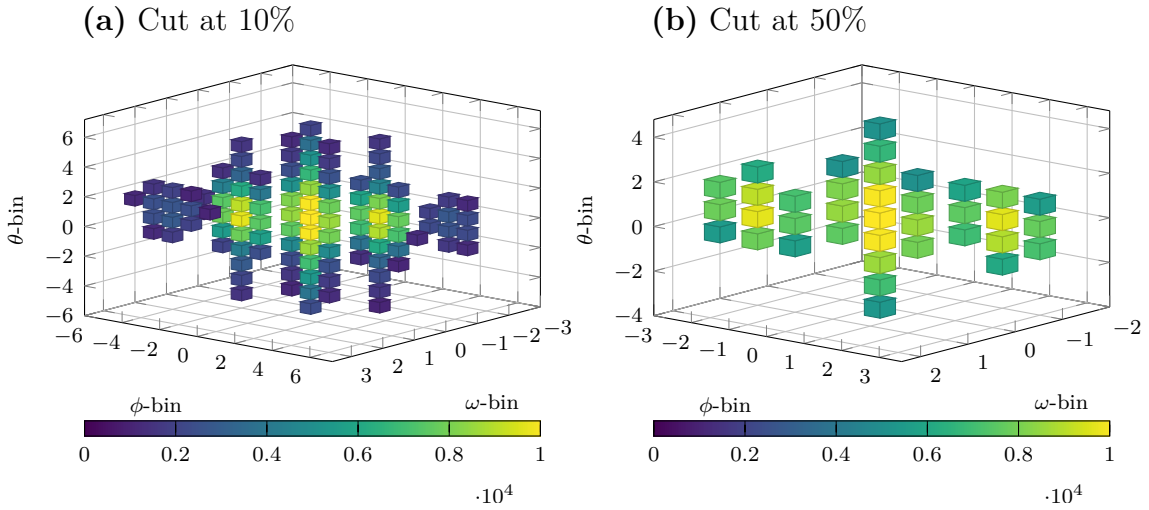


Figure 6.14: Statistics on the density-based clustering (DBSCAN) algorithm using signal tracks without background. Note that the `comp` hit representations with `minweight` = 24 were used.

cuts are now 10% for subfigure (a) and 50% in (b). Subfigure (b) contains 37 cells, each of which appeared in at least 50% of all clusters. This further confirms that the clusters can get very large when using the `comp` hit representations. Especially the extension along the θ-axis seems to be useless.

When comparing those clusters with the fake clusters in Fig. 6.15, a significant size difference is observed in subfigure (b). The cluster size is much smaller for the fake tracks, which has to be further investigated by the average weight plots. Such a big difference was not observed when using the `shallow` hit representations.

In Fig. 6.16, the background overlay is added to the signal tracks. As expected, the clusters get even larger, which may partly explain the deterioration of the neural network resolution when nominal phase-3 background is introduced. Therefore, a new clustering algorithm that avoids arbitrarily expanding the clusters becomes even more crucial at higher backgrounds when the `comp` hit representations are used.

[1] This is statistically confirmed in Chap. 7 and 8.

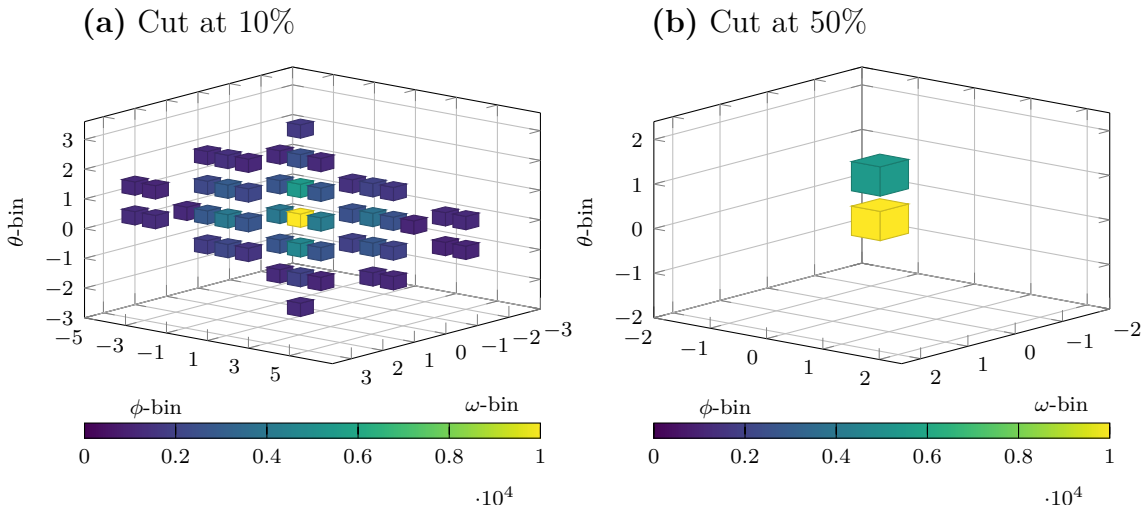


Figure 6.15: Statistics on the density-based clustering (DBSCAN) algorithm using fake tracks from nominal phase-3 background only. Note that the `comp` hit representations with `minweight` = 24 were used.

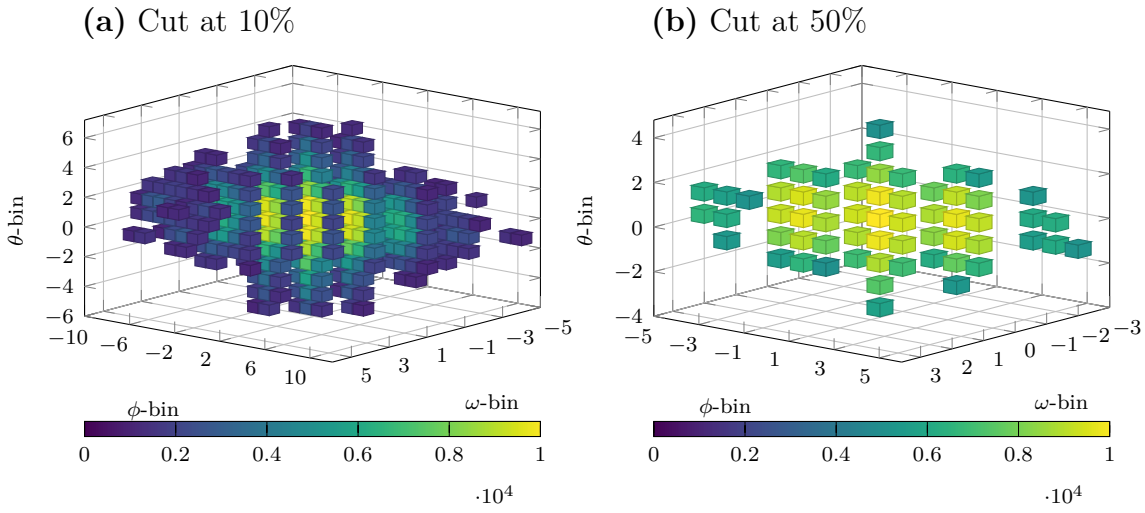


Figure 6.16: Statistics on the density-based clustering (DBSCAN) algorithm using signal tracks with nominal phase-3 background. Note that the `comp` hit representations with `minweight` = 24 were used.

When considering the average weights with `comp` hit representation, the clusters are significantly smaller. In Fig. 6.17, the average weights of real tracks with no background are illustrated. Note that the weight cuts have now been increased to a minimum average cell

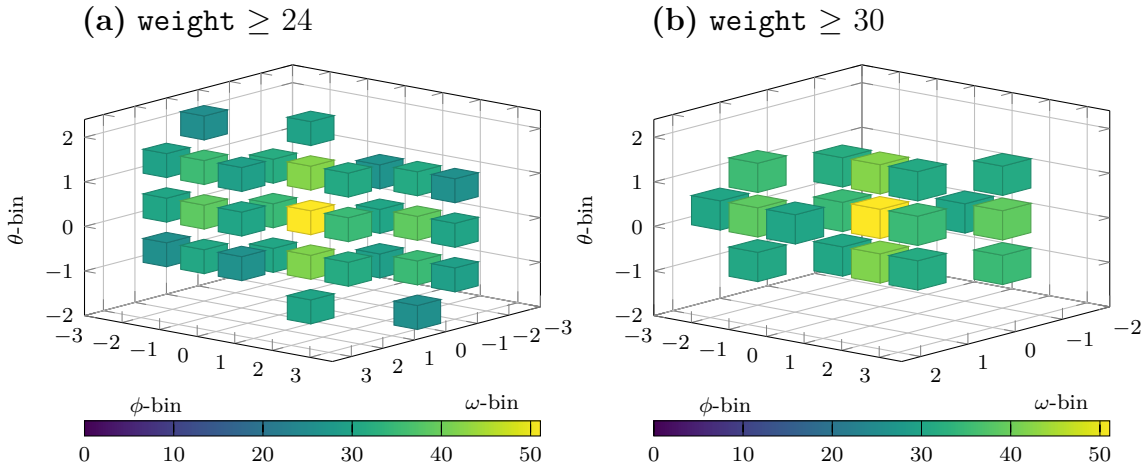


Figure 6.17: Visualization of the average cluster cell weights using signal tracks without background. Note that the `comp` hit representations with `minweight` = 24 were used.

weight of 24 in subfigure (a) and of 30 in (b). This average weight of 24 in subfigure (a) coincides with the `minweight` parameter of 24 that was used for the DBSCAN clusters before in Fig. 6.14 (a). Given the significant disparity in cluster sizes, it suggests that the DBSCAN algorithm produces a wide range of cluster sizes and, therefore, a wide range of execution times.

In Fig. 6.18, the average fake clusters are vastly different from the real clusters. Only

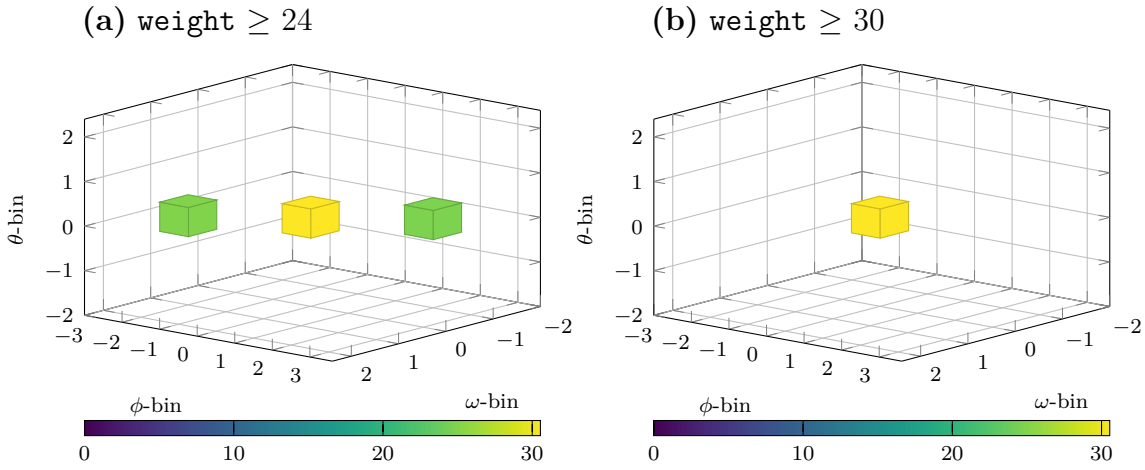


Figure 6.18: Visualization of the average cluster cell weights using fake tracks from nominal phase-3 background only. Note that the `comp` hit representations with `minweight` = 24 were used.

the peak cell itself has an average weight above 30, while the average signal-only cluster contained 18 cells with this threshold. When including background for the real tracks, this

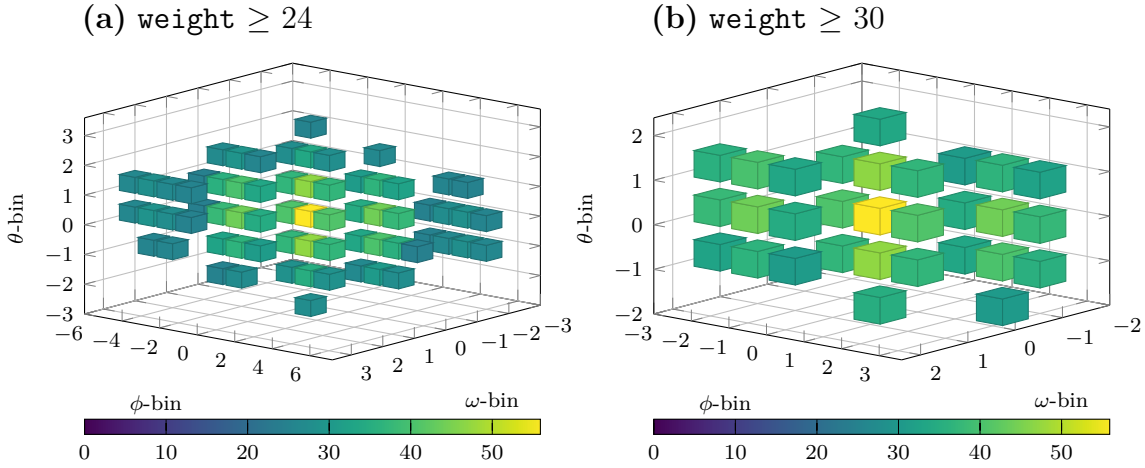


Figure 6.19: Visualization of the average cluster cell weights using signal tracks with nominal phase-3 background. Note that the `comp` hit representations with `minweight` = 24 were used.

number increases to 30, as depicted in Fig. 6.19 (b). Here, the total cluster weight may be used to effectively differentiate between real and fake tracks.

When considering Fig. 6.20, the average peak weight of 55.9 is now considerably larger for the real clusters, as the fake clusters are only at 30.5. As with the `shallow` hit represen-

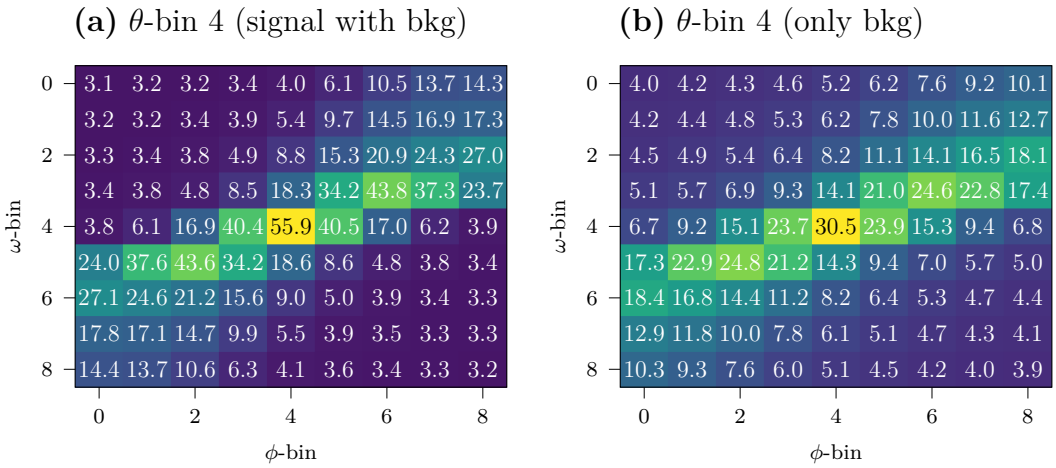


Figure 6.20: The average Hough space weights around the cluster peak, comparing signal tracks with background with nominal phase-3 background only fake tracks. Note that the `comp` hit representations with `minweight` = 24 were used.

tations, the cluster weights are more scattered around the maximum for the fake clusters. Thus, three potential cuts for the suppression of fake tracks are evident:

- Cut on the peak weight of the cluster.
- Cut on the total weight of the cluster.
- Cut on the relative weight, i.e., the weight distribution, of the cluster.

To efficiently use those cuts, however, a new clustering algorithm that does not create arbitrarily large clusters has to be invented. Such a new clustering algorithm can be motivated by the average weight distributions of the real clusters in this section. This algorithm is introduced in Chap. 7 and optimized in Chap. 8 using real data from experiment 26 instead of Monte Carlo simulations.

In conclusion, the `shallow` hit representations seem to be considerably inferior to the `comp` representations. The average peak weight, the total weight, and the number of cluster cells are not much larger when compared to the fake clusters. The very small weight contributions of the outer super layers pose significant problems. On the one hand, they do not contribute significantly to a cluster. This can create problems in the hit-to-cluster association algorithm, where the weight contribution of each hit is considered. On the other hand, background accumulating in the inner super layers can produce very large clusters compared to the real clusters in the Hough space. The `comp` representations do not have any of these problems; only the relative weight contribution of the fake tracks is slightly less distinct as in the `shallow` statistic.

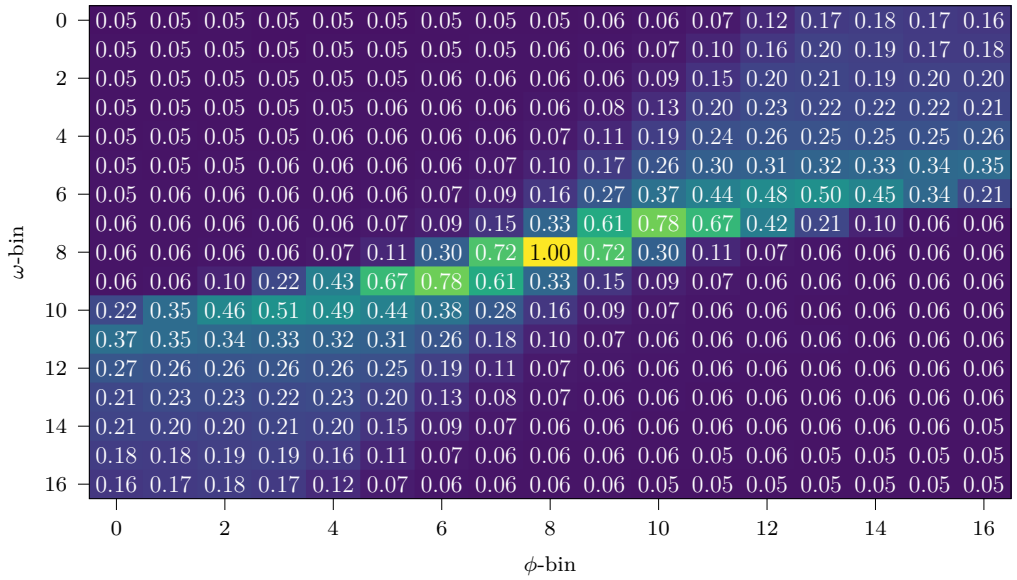
For these reasons, only the `comp` hit representations will be used throughout the remainder of this thesis.

6.2.3 Relative Cluster Weights

As the `comp` hit representations are considerably more promising for the hardware-oriented clustering algorithm, the relative weights of the average cluster weights are analyzed further. Since there was such a large difference between the average peak cells, the relative weight of a cell is defined as the cell weight divided by the peak weight. In Fig. 6.21, the average relative weights are displayed in a 17×17 matrix around the peak cell. Subfigure (a) contains the real clusters with background overlay, while subfigure (b) contains the fake tracks. Trivially, the relative weight of the peak cell is exactly 1, and all weights are in $[0, 1]$. Note that the same clusters from Sub. 6.2.2 are used.

To find the largest difference in relative weight, subfigure (a) is subtracted from subfigure (b). The corresponding differences are displayed in Fig. 6.22. The 8 cells with the largest differences are marked in yellow. A small number of cells like this should not pose a problem for the hardware implementation of the 3DFinder. It should be noted that the θ-bins above and below the maximum bin have also been investigated. However, the largest differences in relative weight were observed in those 8 cells. The sum of the yellow cells may be used to differentiate between fake and real clusters. After the introduction of the new cluster algorithm in Chap. 7, statistics on those yellow cells and the total relative weight are conducted.

(a) θ-bin 8 (signal with bkg)



(b) θ-bin 8 (only bkg)

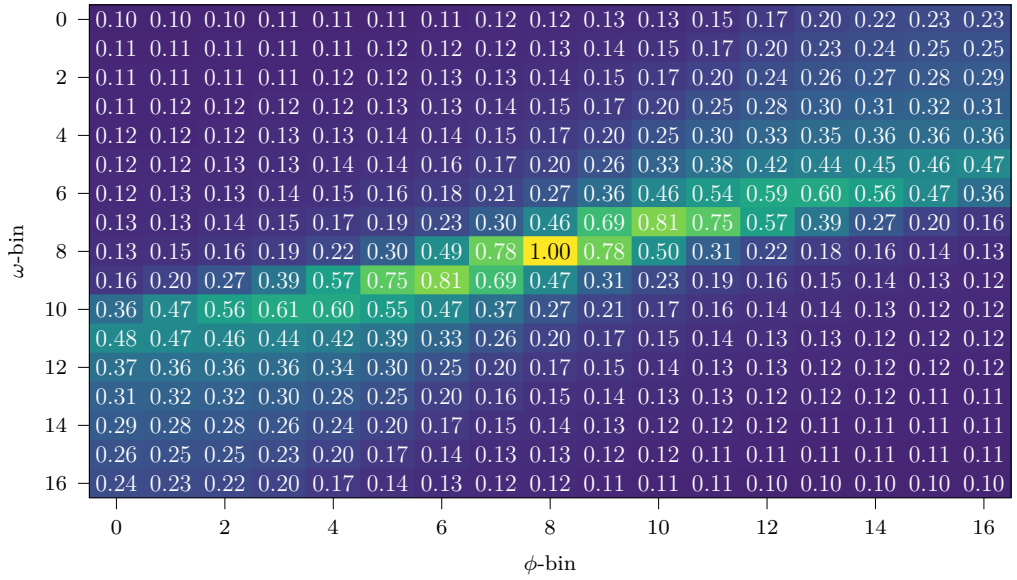


Figure 6.21: Comparison of the relative cell weights (cell weight divided by the peak weight of the cluster) at θ-bin 8, for real tracks (a) and fake tracks (b), both with and from nominal phase-3 background.

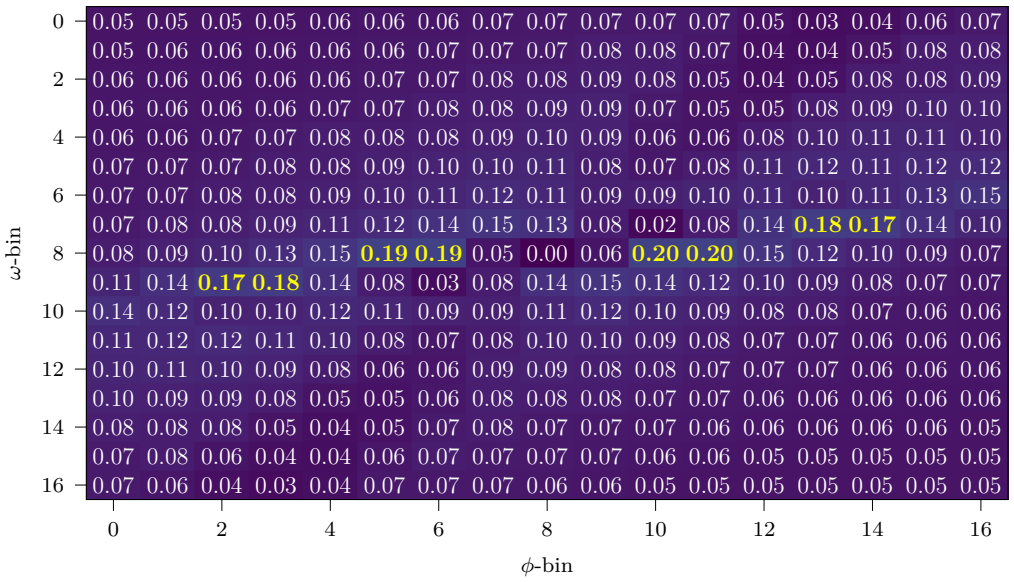


Figure 6.22: Subtraction of the real track relative weights from the fake track relative weights for θ-bin 8. Positive numbers mean that, on average, the relative weights of the fake clusters are larger than those of the real clusters.

Chapter 7

The New Clustering Algorithm

As mentioned in the earlier chapters, a new cluster algorithm should serve the following purposes:

- **Speed**: It has to be very fast, respecting the latency budget determined by the L1 trigger system. Every additional latency gain by the algorithm can be used for a more complex neural network architecture.
- **Implementability**: The algorithm has to be implementable on FPGA boards. This requires a deterministic execution length, simple calculations, and straightforward steps.
- **Efficiency**: It must be efficient on real tracks originating from the IP.
- **Robustness**: The algorithm has to suppress tracks from outside the IP and has to be resistant to high backgrounds, effectively limiting the fake rate.

The simplest, but also most promising, idea seems to be a fixed-volume clustering algorithm. Such an algorithm is proposed in this chapter, implemented in basf2, and analyzed with Monte Carlo data.

7.1 Implementation of Fixed-Volume Clustering

The basic concept of the so-called fixed-volume clustering algorithm is introduced in this section. Note that the algorithm is optimized in several steps throughout Chap. 7 and especially in Chap. 8, where real data is used for testing. In the end, the 3DFinder will only keep the `thresh` parameter, while all other original parameters of Tab. 5.2 are either substituted by new parameters or removed entirely, as the hit-to-cluster association will be improved in Chap. 8 as well. The C++ source code of the new clustering algorithm is included in Appendix A.1.

This fixed-volume clustering is executed in three main steps. These steps are repeated `iterations` times, where `iterations` $\in \mathbb{N}$ is a new clustering parameter.

- **Step 1**: The global maximum cell ("peak") is searched in the complete Hough space.
  - If the peak weight is below `minpeakweight`, the clustering is terminated and the current cluster is discarded.
- **Step 2**: A fixed volume of cells is put around the peak cell.
  - The weights of all cluster cells are summed up to the total weight of the cluster.
  - If the total weight is at least `mintotalweight`, the cluster is saved.

- **Step 3**: All cells in a fixed volume around the peak cell are set to zero.

A detailed explanation for each step is given in the following:

Step 1: A simple iteration over all the weights in the Hough space determines the maximum index, i.e., the position of the cell in the Hough space with the largest weight. Both the index and the weight are saved. If this global maximum weight is below `minpeakweight`, the clustering is terminated for this Hough space. Only the clusters that have been collected until now are used in the hit-to-cluster association.

Step 2: The cluster shape is the same for every cluster and is being displayed in 7.1. Note

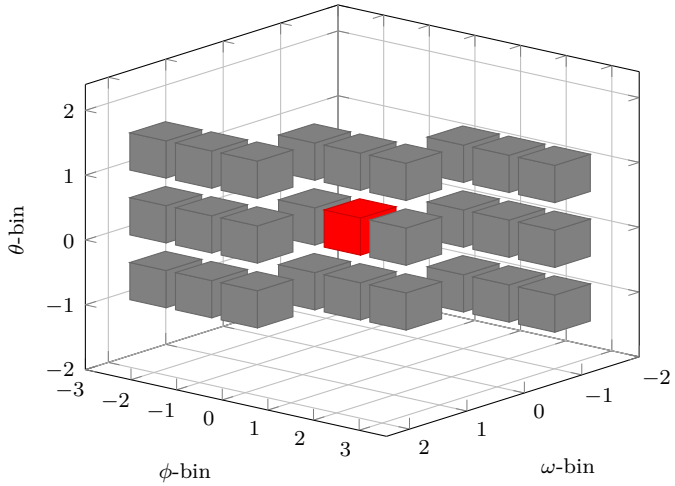


Figure 7.1: The fixed cluster shape that is put around the global maximum cell (here in red).

that this fixed cluster shape is chosen according to the cluster statistics in Chap. 6. With the `comp` representations in mind, the average weight distributions of Fig. 6.19 (b) and Fig. 6.17 (b) inspired the shape. The red cell defines where the global maximum of the Hough space is forced to be. It is important to note here how the borders of the Hough space are handled. If the global maximum is at θ-bin 0 or 8, the layer that should be below or above this maximum in the θ-dimension is ignored, i.e., the fixed cluster shape has 9 fewer cells. Likewise, if the global maximum occurs at ω-bins 0 or 39, the cells above or below are also excluded, resulting in 9 fewer cells too. The ϕ-axis boundaries are different. As $0° \triangleq 360°$, the cluster has to continue on the other side of the Hough space. This is ensured by using the modulo operator when determining the current cluster cells. Hence, every cluster now consists of no more than 27 cells, where the center cell has the maximum weight of the cluster. If the total weight, i.e., the sum of all the individual cell weights, of the cluster is at least `mintotalweight`, the cluster is accepted. Otherwise, this cluster is skipped and deleted. Afterwards, a new global maximum search is conducted, granted the next iteration number is within the `iterations` parameter.

Step 3: A new fixed shape determined by three new parameters, `omegatrim`, `phitrim`, and `thetatrim` deletes all cells around the global maximum index. The first implementation is a simple $(2 \cdot \texttt{omegatrim} + 1) \times (2 \cdot \texttt{phitrim} + 1) \times (2 \cdot \texttt{thetatrim} + 1)$ cube centered on the maximum index. Later, this shape is refined in order to delete the cluster according

to its average shape.

As the cluster has been deleted in the Hough space, a new global maximum search should find the next largest maximum cell of a different cluster. Hence, at most `iterations` clusters and therefore tracks can be found with this clustering algorithm. Since the purpose of the Neuro Trigger is not to find every single track but rather to give a positive or negative trigger decision based on the existence of at least a single IP track for the current event, this upper bound is acceptable. In Tab. 7.1, the first set of parameters for the fixed-volume clustering with some initial default values are listed. Thus, the number of possible

Table 7.1: The new clustering parameters for the fixed Hough space volume clustering. Some initial default values are assigned.

Parameter	Datatype	Default	Description
`dbscanning`	`bool`	`False`	switch between DBSCAN and fixed volume
`iterations`	`int`	5	number of cluster searches per Hough space
`mintotalweight`	`int`	500	minimum of the total cluster weight
`minpeakweight`	`int`	0	minimum weight of the peak cluster cell
`omegatrim`	`int`	5	number of deleted cells in each ω direction
`phitrim`	`int`	5	number of deleted cells in each ϕ direction
`thetatrim`	`int`	5	number of deleted cells in each θ direction

tracks that can be found in the Hough space of the current event is limited to 5. The `minpeakweight` parameter is disabled for now until distributions of this parameter are plotted for real and fake tracks, where the cut position is determined. A `mintotalweight` of 500 is chosen according to the average weights observed in Fig. 6.19 (b) and Fig. 6.17 (b). This is a necessary cut in order to get rid of very small clusters that appear when there are less than 5 tracks in the Hough space. In this case, the next global maximum may be on the tails of such a cluster, as all the center cells are already set to zero. The trimming parameters now define a $11 \times 11 \times 11$ cube centered on the maximum. This shape is appropriate to extensively delete a cluster for the first tests of this algorithm.

Note that only the `comp` hit representations are used, as they showed to be considerably more promising in Chap. 6. All parameters used were the default ones of the DBSCAN clustering (see Tab. 5.2) and the default ones of the fixed-volume clustering (see Tab. 7.1). As in the studies before, 10,000 single muon particle gun tracks were generated using the parameters of Tab. 5.7. To compare the z- and θ-resolutions of the fixed-volume clustering with both the DBSCAN and the 2DFinder resolutions, all three track-finding options are displayed in Fig. 7.2. All distributions are purposefully fitted with only one Gaussian in order to assess the quality of the distribution. As observed in Chap. 5, the 2D-Finder distributions show a large second Gaussian extending over $\pm 20\,\mathrm{cm}$. The DBSCAN clustering, on the other hand, disperses a considerable number of tracks all around the z-axis. In subfigures (e) and (f), however, when using fixed-volume clustering, those misfitted tracks are gone. Only a single Gaussian is necessary to describe the z-distribution, while the track-finding efficiency is above 99%. Thus, this new clustering algorithm is already better than DBSCAN with default parameters and seems to be very promising in future studies.

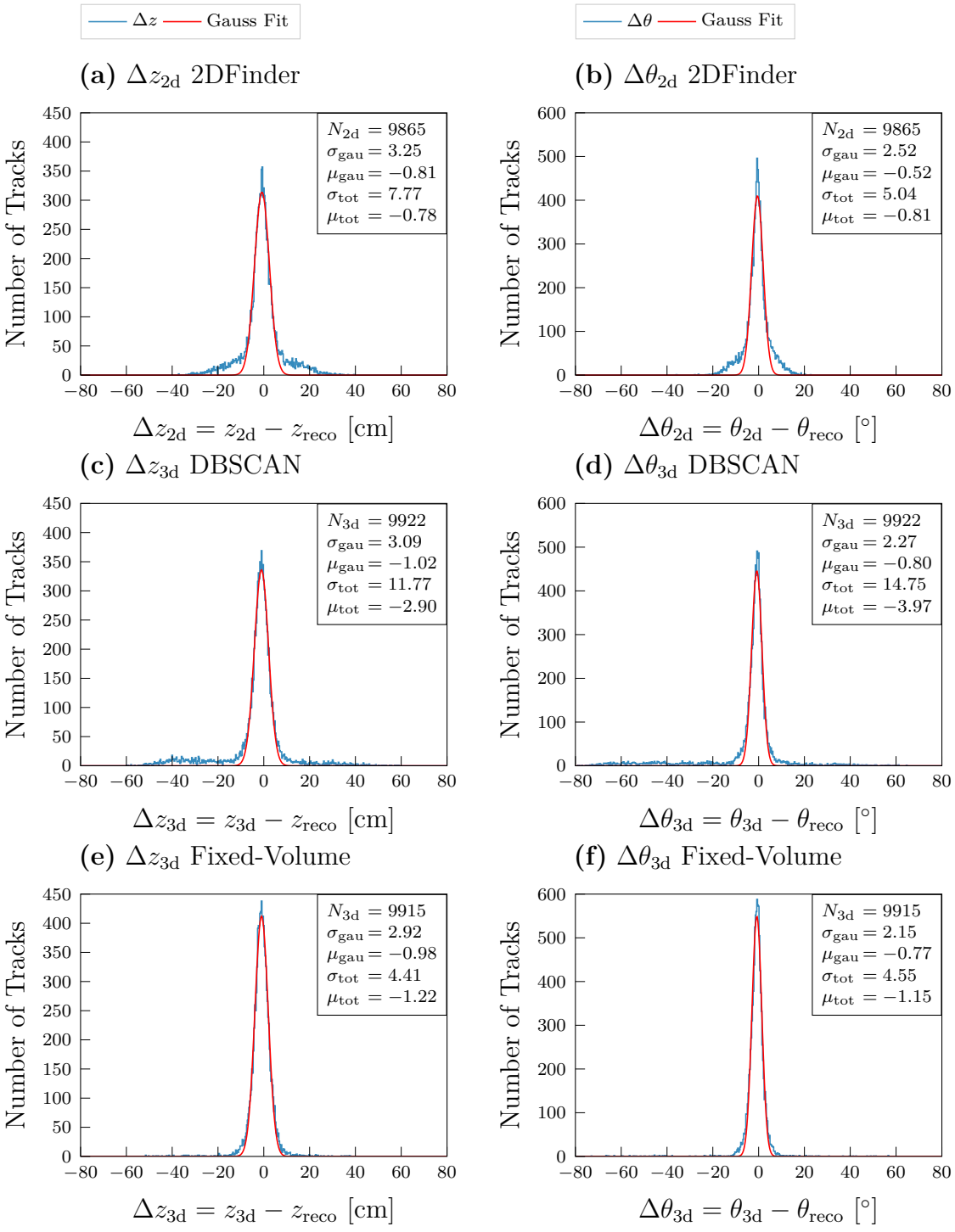


Figure 7.2: Comparison of the z-resolution and θ-resolution of the neural network with 2DFinder input and 3DFinder input using the two different clustering algorithms. The particle gun generated 10,000 single muon tracks in the full CDC acceptance range.

7.2 Analysis of the Weight Distributions

To determine the parameters `minpeakweight` and `mintotalweight` for the Monte Carlo studies, a weight analysis is conducted. These could be used in order to differentiate between fake and real tracks. Furthermore, a relative weight can be introduced to use the difference in average weight distribution of the fake clusters observed in Sub. 6.2.3.

7.2.1 Total and Peak Weight Distributions

As a first check of the new clustering algorithm, the average weights of the fixed shape are examined for both real and fake clusters. For this purpose, 10,000 single tracks without any background were generated, and the distribution is illustrated in Fig. 7.3 (a). This

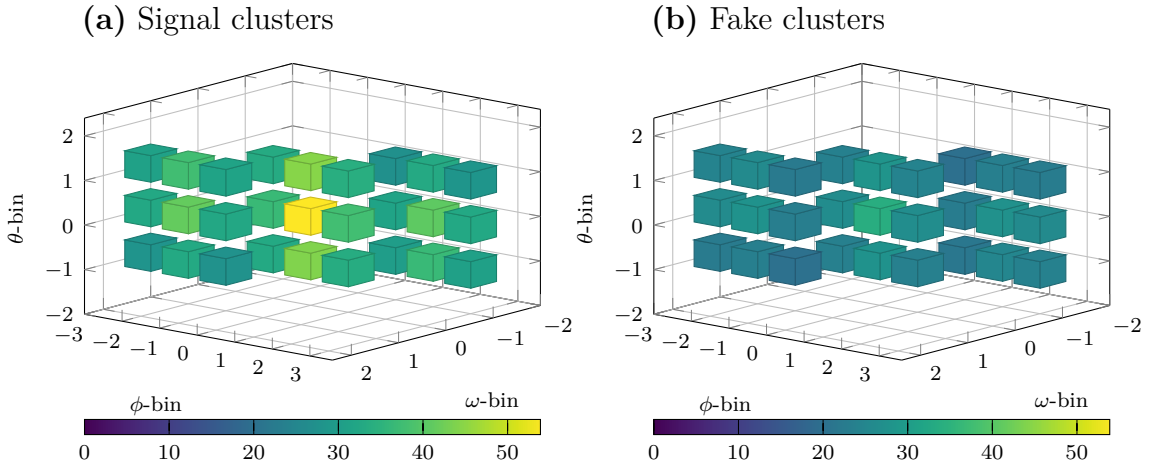


Figure 7.3: The average clusters using the new fixed-volume clustering algorithm for real and fake tracks.

distribution is very similar to the one observed in Fig. 6.17 (b). As in Chap. 6, nominal phase-3 background is used in order to determine the average fake cluster. This cluster is displayed in subfigure (b), where the heatmap is normed to the one in (a). Hence, a significant difference in average total weight and peak weight is observed. To determine optimal cut positions, the total cluster weight and peak weight distribution of those 10,000 tracks are displayed in Fig. 7.4. In subfigure (a), the total cluster weight is plotted for real tracks and fake tracks from nominal phase-3 background. Note that the default `mintotalweight` parameter of 500 removes a large part of the fake cluster distribution. As expected, the distributions are distinct enough and allow for a total weight cut that should reduce the fake rate. A cut of 730 is illustrated in this subfigure. However, the real tracks do not have any background in their data sample. When background is added, the distribution should change, making a new cut necessary. In subfigure (b), the peak weight distributions are depicted. While the average peak weight of the real tracks is 53.9, the fake weight is just 33.6. Thus, a cut of 43 for `minpeakweight` seems reasonable when considering Monte Carlo tracks.

To assess the total cluster weight and peak weight of the fixed clustering algorithm even further, a set of correlation plots is shown in Fig. 7.5. The correlation between the total

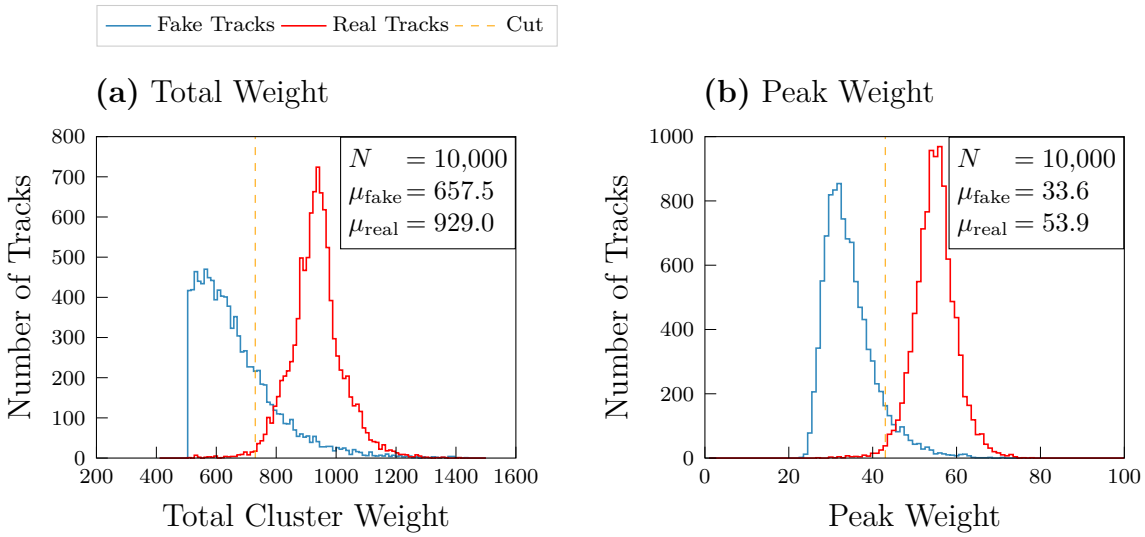


Figure 7.4: The distribution of the total cluster weight and the peak weight using the new fixed-volume clustering algorithm for real and fake tracks.

weight and the peak weight is plotted in subfigure (a) and, for fake tracks, in (b). It is not surprising that a positive correlation exists between the peak weight and the total weight. As the peak weight increases, the total weight increases as well, on average. In subfigure (c), the correlation between the total weight and the reconstructed transverse momentum is shown. The total weight is constant over nearly the whole momentum range, which confirms that the chosen cluster shape is indeed appropriate for different momentum tracks. Only at low momentum, i.e., $p_{\mathrm{T,reco}} \in [0.35, 0.5]\,\mathrm{GeV}/c$, the average total weight decreases significantly. This is expected, as some track segments are lost due to crossing angles of over 45° for the low-momentum tracks. The same is observed for the peak weight in subfigure (d). When considering the reconstructed θ-angle, an arched shape is observed for the total weight distribution in subfigure (e). A more shallow emission angle along the z-axis results in smaller clusters. This may be explained by a missing outer track segment. When considering the peak weight in subfigure (f), however, such an arch is not observed.

To check the effect of a total weight cut of 730 and a peak weight cut of 43 on the potential efficiency, low-momentum and shallow-θ tracks were generated. For this purpose, three datasets of single particle gun tracks at the CDC boundaries were created without background.

- A low-momentum set comprising 10,000 tracks with $p_{\mathrm{T,MC}} \in [0.35, 0.5]\,\mathrm{GeV}/c$.
- A small-θ set comprising 5000 tracks with $\theta_{\mathrm{s,MC}} \in [19, 35]°$.
- A large-θ set comprising 5000 tracks with $\theta_{\mathrm{l,MC}} \in [123, 140]°$.

In Fig. 7.6, the total cluster weight and the peak weight are displayed for those three datasets. It is evident that both cuts have the potential to significantly reduce efficiency, depending on the distributions observed in real data. The final cut positions have to be carefully determined by the fake rate and the required efficiency. Note that more than half of the shallow-θ tracks have not been found by the 3DFinder. While there is no cut on the peak weight, the `mintotalweight` = 500 threshold is apparently already rejecting some

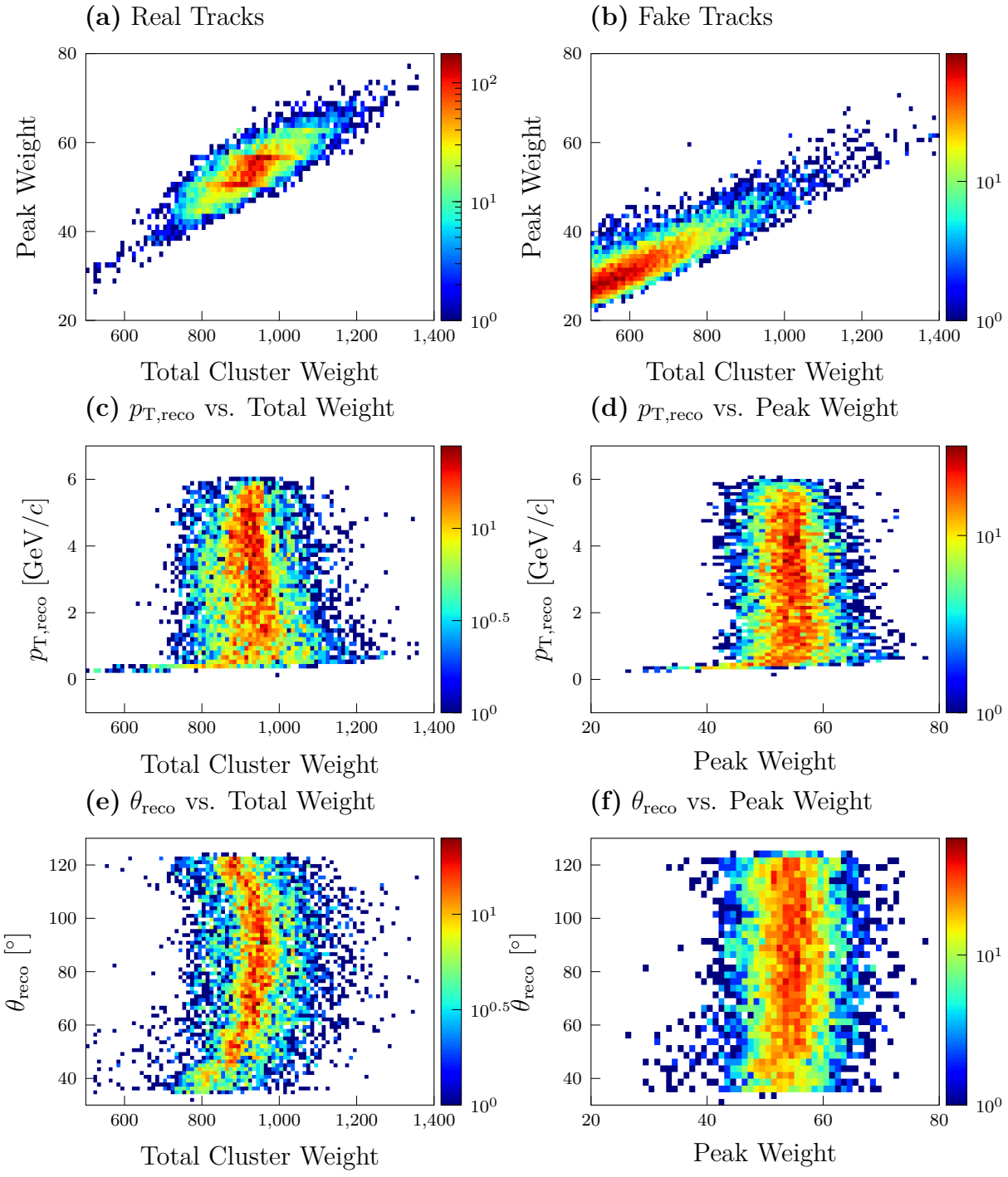


Figure 7.5: Logarithmic heatmaps of the correlation between the peak weight, the total cluster weight, the reconstructed θ, and the reconstructed p_{T} using the new fixed-volume clustering algorithm.

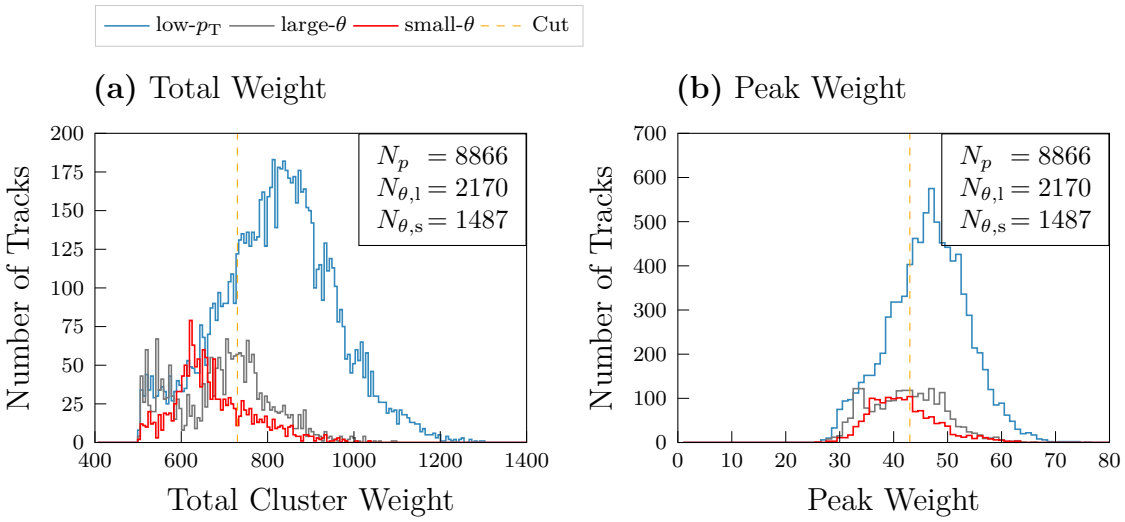


Figure 7.6: The total weight and peak weight distributions of tracks with low transverse momentum ($p_T \in [0.35, 0, 5]\,\mathrm{GeV}/c$), large θ angles ($\theta_l \in [123, 140]°$), and small θ angles ($\theta_s \in [19, 35]°$). Note that no background was used in those data samples, and for each θ range, only 5000 tracks were generated.

very small clusters.

Furthermore, the effect of background on the total and peak weights has to be assessed. In Fig. 7.7, nominal phase-3 background is introduced to the particle gun tracks. In the red

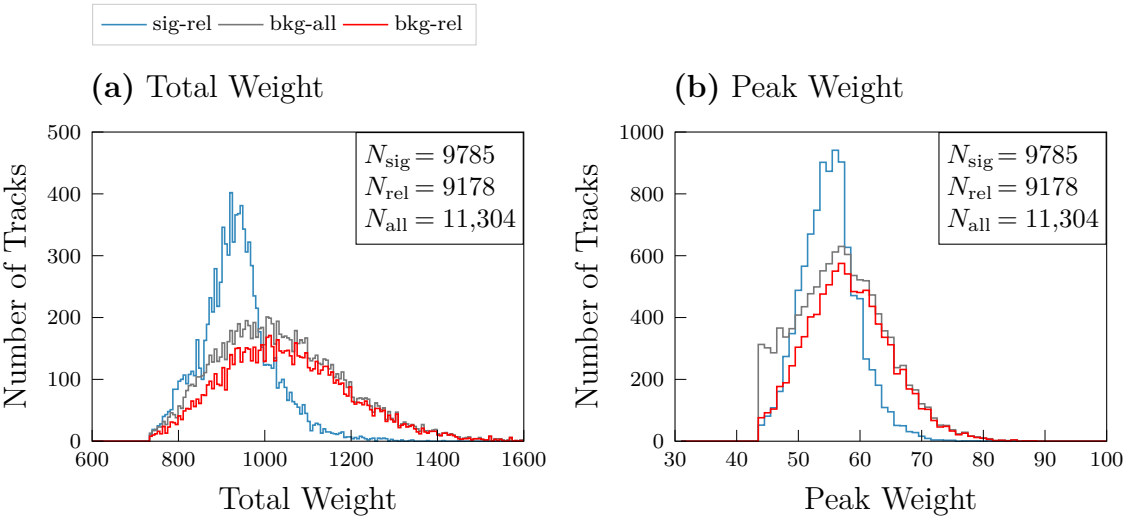


Figure 7.7: The total weight and peak weight distributions of IP tracks in the full CDC acceptance range with and without nominal phase-3 background.

distribution, all tracks that have been related to a reconstructed track are displayed, while the gray distribution includes all 3DFinder tracks. As a comparison, the blue distribution models related tracks of the dataset without any background. As expected, the distributions with background deteriorate towards larger average weights since fake track segments add to the cluster weight. The impact on the peak weight appears to be smaller, however.

Here, a cut at 43 can reduce fake tracks, as demonstrated by the gray distribution, which includes these tracks.

7.2.2 Relative Weight Distributions

As mentioned and motivated in Sub. 6.2.3, the relative weight of a cell was defined as the cell weight divided by the peak weight, which can help to distinguish fake from real clusters. Especially the number of fake clusters with a high total weight (≥ 700) that survive the `mintotalweight` cut could be reduced further with a cut on the relative weight. In Fig. 7.8, the relative weight distribution of all cluster cells is depicted. While subfigure (a)

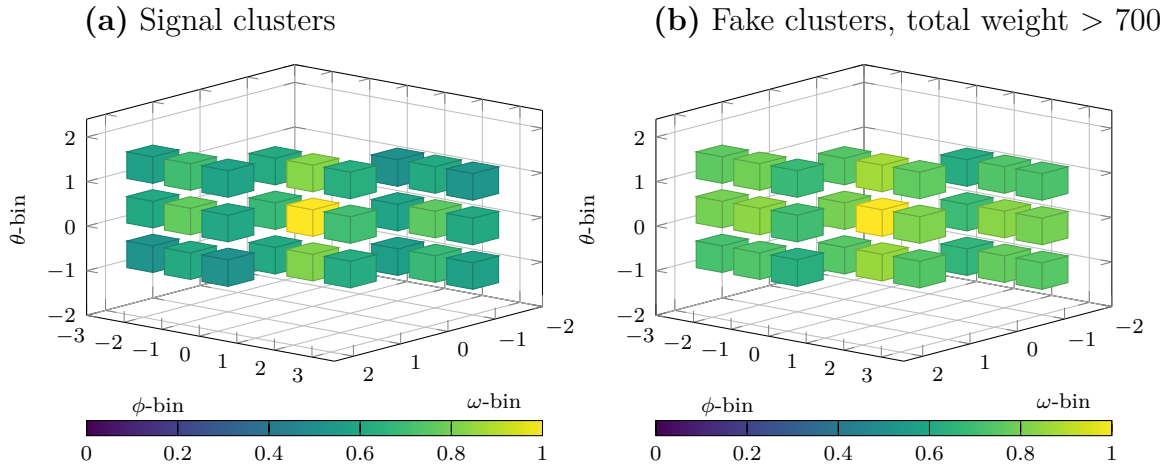


Figure 7.8: Three-dimensional clusters of the relative weights. The weight of each cell is divided by the maximum, added up with all tracks, and then divided by the total number of tracks.

displays the average relative weight of all real clusters without any background, subfigure (b) displays the average relative weight of fake clusters with a total weight of at least 700. All of those fake cells have, on average, a larger relative weight. This motivates a total relative weight, upon which one could introduce a cut that is only active when the total weight is above a certain threshold (e.g., 700). The distribution of the relative weight sums is plotted in Fig. 7.9. In subfigure (a), the 10,000 tracks of the cluster statistics are used. As the green distribution of the fake tracks is larger than the blue one of the real tracks, a cut of around 20 may be possible. However, the real track distribution shifts towards the right when nominal phase-3 background is introduced. This shift is displayed in subfigure (b), where a cut cannot be under 22.5 due to efficiency losses. Hence, the total relative weight of the fixed cluster cells cannot be used to reduce background, as a loss in efficiency is not acceptable.

Since the complete Hough space is accessible during clustering, cells that are outside the fixed cluster shape could be used for relative weight considerations. In Fig. 6.22, 8 cells with a particularly high relative weight difference are analyzed. The 8 yellow cells in subfigure (a) have been motivated by the statistical analysis in Sub. 6.2.3. The sum of those cells is plotted in subfigure (b). Here, the red distribution includes nominal phase-3 background, while the green distribution consists of only the fake tracks within this dataset. As the

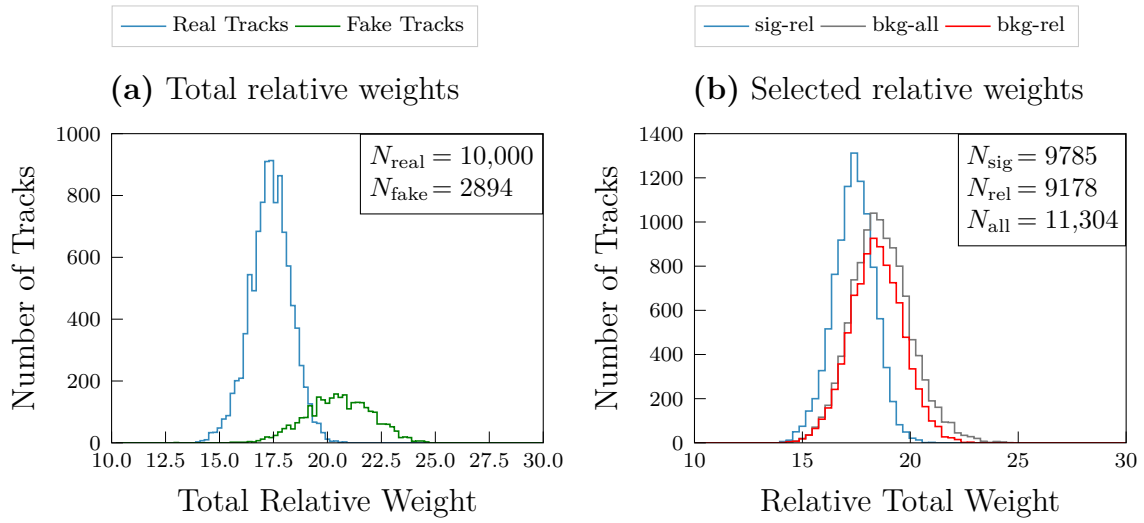


Figure 7.9: The distributions of the total relative weight of real tracks and fake tracks. In subfigure (a), only fake tracks with a total weight of at least 700 are considered. In subfigure (b), nominal phase-3 background is introduced.

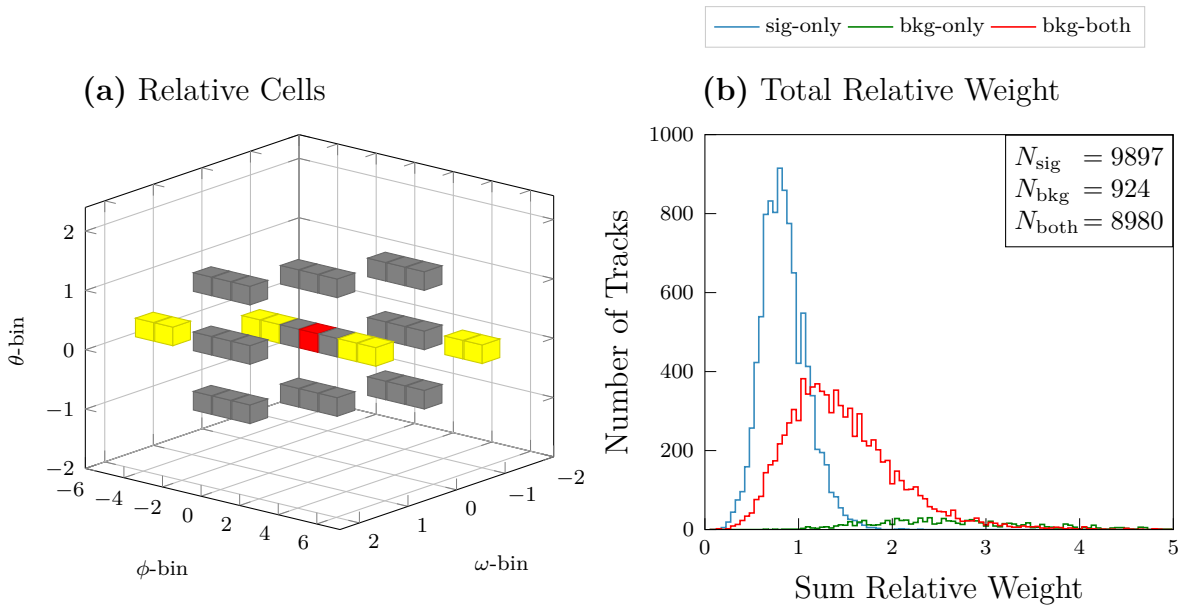


Figure 7.10: The sum of the relative weights in the 8 yellow cells (subfigure (a)) of IP tracks with and without nominal phase-3 background.

inclusion of background shifts this distribution considerably, such a cut seems not to be usable as well.

The impracticability of total relative weight cuts can be explained by the randomness of the background track segments in the Hough space. A significant number of real clusters are observed to be above such background track segments (see Fig. 8.10 (a)). This results in completely different relative weight distributions that cannot be differentiated from the fake clusters.

7.3 Analysis of the Track Segment Distributions

While the `mintotalweight` and `minpeakweight` parameters already filter out some fake clusters, many background clusters can still meet these thresholds. The most promising parameter is `minhits`, a cut on the total number the track segments, as analyzed in Sec. 5.5. However, duplicate track segments in the same super layer make it impossible to have a strict cut on the number of total track segments related to a 3DFinder cluster. Therefore, the number of hit super layers has to be investigated. For this purpose, the datasets of 10,000 single particle gun tracks originating from the IP with and without nominal phase-3 background are used again. Fig. 7.11 compares the track segment frequencies for all track segments (blue distribution) and the corresponding unique track segments (orange distribution). The vast majority of real tracks in subfigure (a) cross 8 or 9 super layers. In

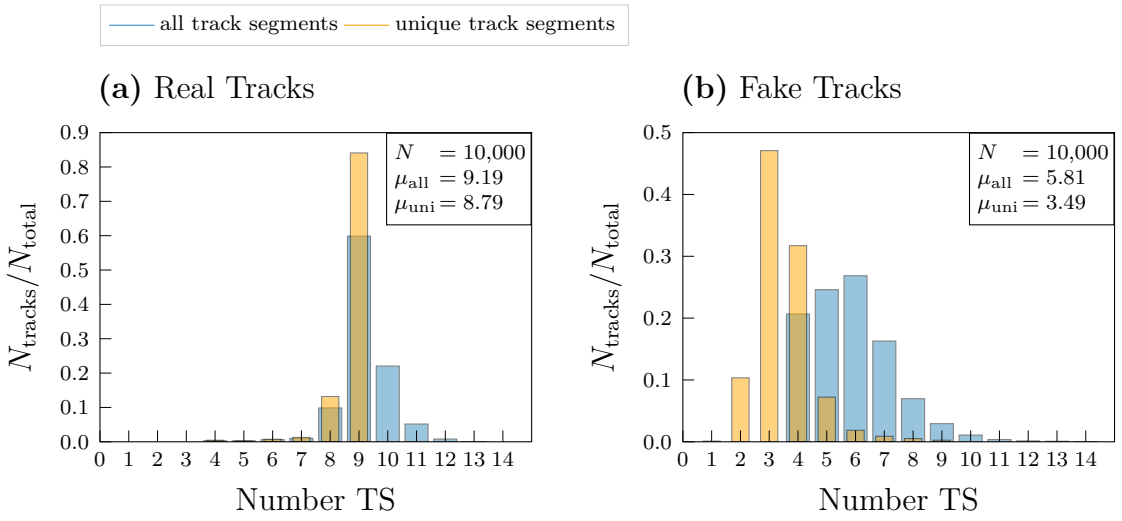


Figure 7.11: Track segment distributions of the 3DFinder using the new fixed-volume clustering algorithm for real and fake tracks. Note that unique track segments are only allowed to have at most one track segment per super layer. Hence, all duplicate track segments are removed.

subfigure (b), the fake tracks are displayed. While the `minhits` parameter visibly restricts the total number of track segments to 4, most fake tracks include actually less than 4 super layers, as can be seen in the orange distribution. Thus, the `minhits` parameter cannot be used as an efficient cut to reduce the fake rate.

For this purpose, a new 3DFinder parameter named `minsuper` is introduced. After estab-

lishing the hit-to-cluster association using the confusion matrix (see Sub. 5.1.2 or Sub. 8.2.2), the super layer number of each track segment belonging to a cluster is determined. Only if at least `minsuper` different super layers have been hit, the track is accepted by the 3DFinder.

When track segments get lost for real IP tracks, those are most likely in the outer super layers due to either shallow emission angles or large crossing angles due to low transverse momentum. In order to increase the efficiency of such tracks, the `minsuper` cut could be used only on the inner super layers. For this purpose, the number of unique track segments within the first 6 super layers is displayed for signal, fake, shallow-θ, and low-momentum tracks in Fig. 7.12. Although those boundary tracks in subfigures (c) and (d) are biased

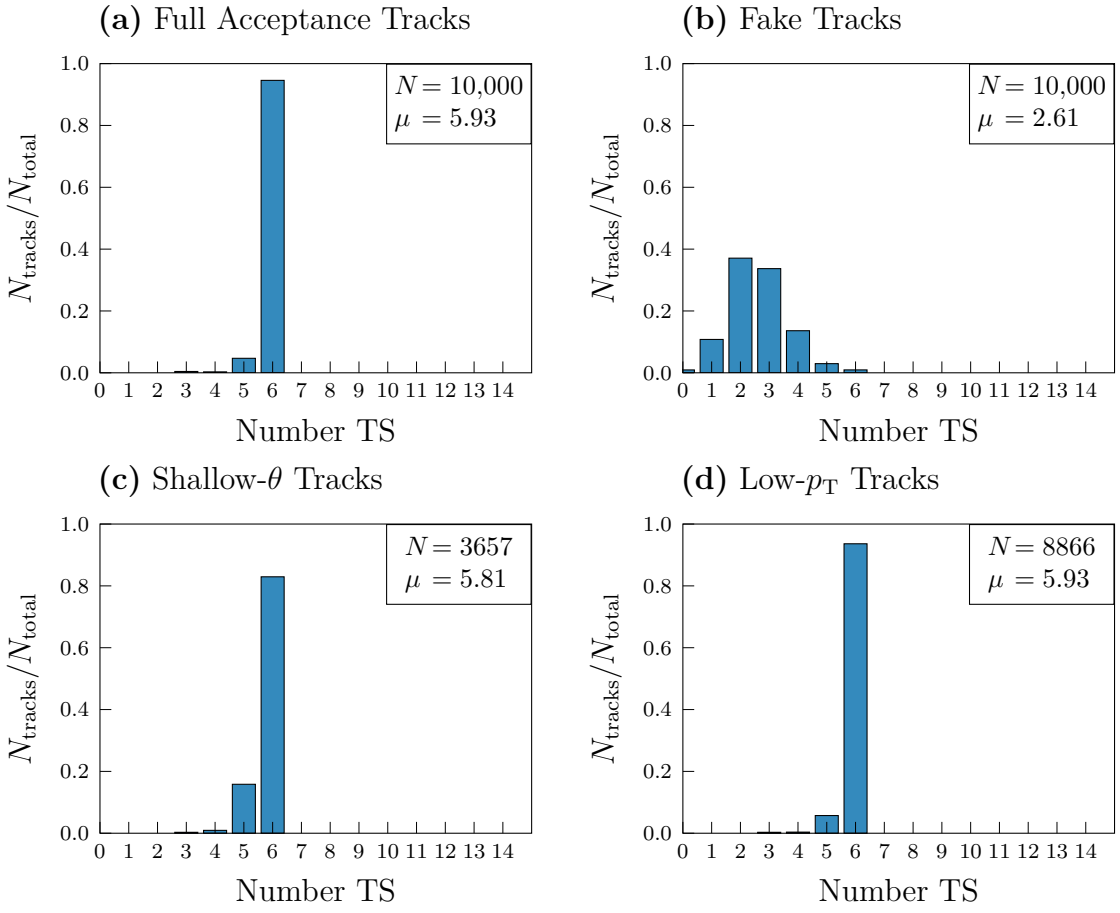


Figure 7.12: Number of how often track segments in the first six super layers contributed to a 3DFinder cluster using the new fixed-volume clustering algorithm for real and fake tracks. Note that all duplicate track segments have been removed.

in the regard that the 3DFinder did not find all tracks, nearly all tracks have at least hit 5 of the first 6 super layers. In contrast, more than 96% of all the fake tracks in subfigure (a) have less than 4 hits in the first 6 super layers.

Consequently, one can define the new `minsuper` parameter as the minimum number of super layers that have to be hit within the first 6 super layers.

7.4 Multiple tracks

Thus far, all datasets consist of single particle gun tracks. However, the efficiency of the new fixed-volume clustering method needs to be evaluated for higher multiplicities. For this purpose, 6 tracks with random parameters from Tab. 5.7 are generated for each of the 5000 events. The parameters used for the 3DFinder are listed in Tab. 7.2. With

Table 7.2: Used clustering parameters for the fixed Hough space volume clustering for the multiple track events.

Parameter	Value
`iterations`	5
`mintotalweight`	730
`minpeakweight`	43
`omegatrim`	5
`phitrim`	5
`thetatrim`	5
`minsuper`	6

`iterations` = 5, at most 5 tracks can be found per event. The maximum possible number of found tracks is therefore $5000 \cdot 5 = 25{,}000$. In this default implementation, a $11 \times 11 \times 11$ "cuboid" cut-out is used to delete a found cluster before the next search is started.

This is compared to a refined cluster volume that is more similar to the average cluster shape. Here, the three `trim` parameters from Tab. 7.2 are still used but have a slightly different meaning. In Fig. 7.13 (b), the new shape is illustrated in a fixed θ-bin, where the red cells are set to zero. This red volume is defined as follows: The center of the shape is defined by the peak cell. The parameter `phitrim` defines the number of cells in each

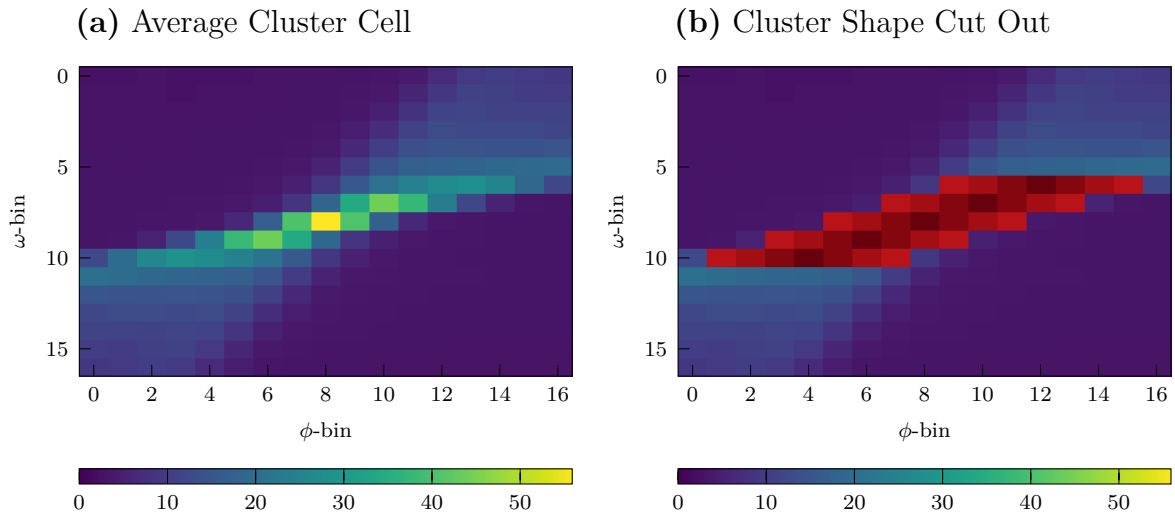


Figure 7.13: The average Hough space weights around the cluster peak before and after the deletion of the cells in the Hough space. Every red cell gets set to 0. Note that the `comp` hit representations with `minweight` = 24 were used.

ϕ-direction relative to the central ω-bin. In this example, `phitrim` = 3, which means a total of 7 bins in every ω-bin, i.e., the center bin and 3 to the left and right. Similarly, `omegatrim` defines how far the cluster should be extended in each ω-direction. Note that the centers, defined by the `omegatrim` parameter, are always shifted 2 ϕ-positions to the right for each smaller ω-bin value and to the left for a larger ω-bin value. This allows for a shape that is similar to the average cluster. Hence, an increase in efficiency is expected when considering multiple tracks because nearby clusters are less likely to be deleted by either one. The `thetatrim` parameter defines by how many θ-layers above and below the peak weight the cluster shape is extended. In this analysis, `phitrim` = 3, `omegatrim` = 2, and `thetatrim` = 2.

In Tab. 7.3, the number of found tracks for each multiplicity is displayed, comparing the two mentioned cluster shapes. As anticipated, the refined cluster shape cut-out is more

Table 7.3: The number of found 3DFinder tracks for two different cluster deletion methods using fixed clustering of 5000 signal events with 6 tracks each. Note that these are the counts of the tracks that have been related to a primary reconstructed track.

11×11×11 Cut-Out		
Multiplicity	**Events**	**Tracks**
5 tracks	4039	20195
4 tracks	887	3548
3 tracks	71	213
2 tracks	3	6
1 track	0	0
0 tracks	0	0
Total	5000	23962

Cluster Shape Cut-Out		
Multiplicity	**Events**	**Tracks**
5 tracks	4399	21995
4 tracks	486	1944
3 tracks	102	306
2 tracks	12	24
1 track	0	0
0 tracks	1	0
Total	5000	24269

efficient since, in the iterative search, smaller regions of the Hough space are cleared. Thus, other clusters are less likely to be deleted, leading to a total efficiency of 97.1%. In 88.0% of all events, all possible 5 tracks were detected. However, for the cuboid cut-out, 5 tracks were found in only 80.8% of all events.

The z-resolutions of the multiple tracks are depicted in Fig. 7.14. The resolution of both distributions can be fitted with a single Gaussian, and the standard deviation of 3.03 cm is comparable to the single track standard deviation of 2.92 cm in Fig. 7.2 (e). As a result, the new fixed-volume clustering is working properly for events with a higher multiplicity. The new cluster shape cut-out will be used in the next section. However, an even better shape is implemented when the hit-to-cluster association algorithm is redone in Chap. 8.

7.5 Reduction of Nominal Phase-3 Background

The new fixed-volume clustering algorithm can now be studied on nominal phase-3 background. As depicted in Fig. 5.18, the default 3DFinder implementation exhibited a very high fake rate of 3.26 : 1. With the new cuts on the peak weight, the total cluster weight,

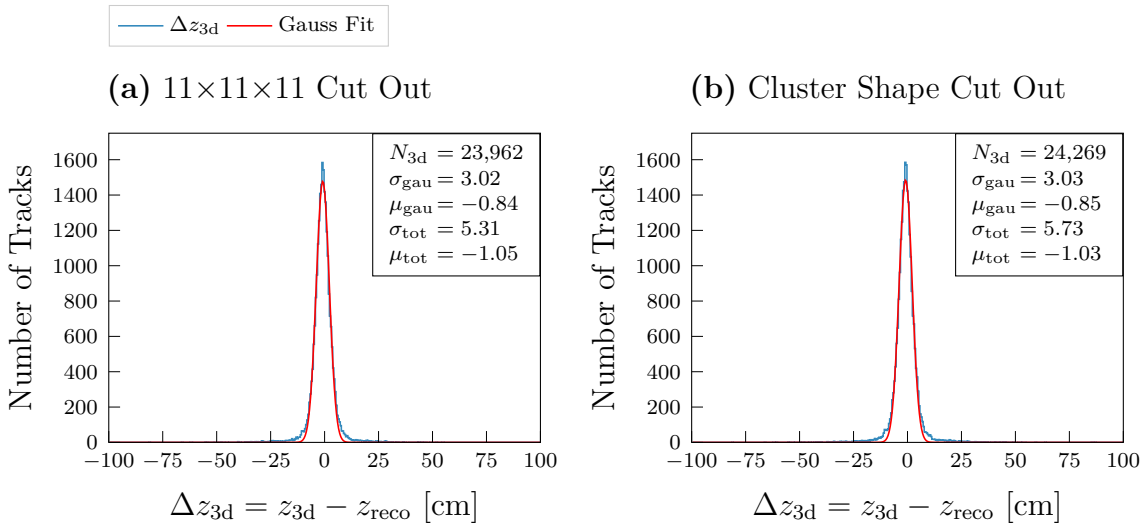


Figure 7.14: The z-resolutions of multiple track events using the new fixed clustering algorithm and IP tracks. Here, two different methods for cluster deletion are used.

and especially the new `minsuper` parameter, a successful suppression of fake tracks is anticipated.

7.5.1 Large Datasets with Different Settings

As in Chap. 5, a large particle gun single event dataset is generated with and without nominal phase-3 background. The parameters for the tracks are listed in Tab. 7.4.

Table 7.4: Parameters of the particle gun single $\mu^{\pm}$ tracks for the big dataset.

Parameter	Value	Distribution
z	$[-100, 100]$ cm	uniform
θ	$[19, 140]°$	uniform in $\cos(\theta)$
ϕ	$[-180, 180]°$	uniform
p_T	$[0.35, 6]$ GeV/c	uniform

To evaluate the performance of the updated 3DFinder, three distinct parameter sets are created, which are listed in Tab. 7.5.

Set 1 does not use the `minsuper` parameter, relying solely on the previous `minhits` parameter for cuts on the track segment numbers. In Chapter 5, the `minhits` parameter was set to the default value 4, resulting in a high fake rate with the nominal phase-3 background. Consequently, it is raised to 6. To further diminish the fake tracks, strict cuts are imposed on the total cluster weight and peak weight.

Set 2 relaxes those weight cuts according to Fig. 7.6 and introduces the `minsuper` parameter with a value of 6. Here, of all 9 super layers, at least 6 different super layers must be hit.

Table 7.5: Three 3DFinder parameter sets for the fixed Hough space volume clustering of the big dataset.

Parameter	Set 1	Set 2	Set 3
`iterations`	5	5	5
`mintotalweight`	730	450	450
`minpeakweight`	43	27	27
`omegatrim`	2	2	2
`phitrim`	3	3	3
`thetatrim`	2	2	2
`minhits`	6	6	5
`minsuper`	0	6	5

Set 3 reduces only the `minsuper` parameter to 5 compared to Set 2, but with a different implementation. A track will now only be accepted when at least 5 of the first 6 super layers contain a related track segment each.

In Fig. 7.15, the large dataset without any background was used. The performance of the three 3DFinder parameter sets can be assessed through both a z-distribution plot and a z-correlation plot. The desired reduction in track-finding efficiency for displaced tracks is most prominent in Set 3. Furthermore, the z-correlation of Set 3 is considerably better compared to the other two parameter sets. The high number of displaced tracks observed in Fig. 5.12 with the DBSCAN algorithm is drastically reduced. The efficiency around the IP is only slightly above the efficiency of the 2DFinder, as the algorithm is more background-resistant with the stricter cuts. Subfigure (c) suggests that Set 2 has a higher efficiency around the IP. When considering subfigure (d), however, it becomes clear that this efficiency gain can be mostly attributed to the feed-down effect. It has to be noted again that the neural network used in this chapter has only been trained with 2DFinder track candidates using low background data. The bad resolutions observed in the correlation plots may disappear when a new neural network is trained with 3DFinder input.

When considering the track segment counts of those three settings, super layer 1, i.e., the first stereo super layer, is missing in nearly half of all 3DFinder tracks. An analysis of this anomaly is conducted in Appendix A.2 for Set 3. While the conclusion is that no problem exists, the insights gained from the algorithm are nonetheless valuable.

The introduction of nominal phase-3 background is depicted in Fig. 7.16. It is immediately evident that parameter Set 3 displays the lowest fake rate in subfigure (f). Using the DBSCAN clustering algorithm with a `minhits` cut of 4 in Sub. 5.3.3, a total of 195,513 tracks were identified, whereas only 40,902 tracks were detected when using `minsuper` = 5. The track-finding efficiency around the IP is still slightly higher than when using the 2D-Finder. In addition to exhibiting the highest fake rate, Set 1 also demonstrates considerable feed-down, as observed in subfigure (a). Hence, solely relying on the weight cuts is not sufficient to deal with such high backgrounds. Instead, a cut on the hit super layers is by far the better option for a robust algorithm. Note that only the reconstructed tracks of primary Mote Carlo particles are plotted in the related subfigures, while in subfigures (b),

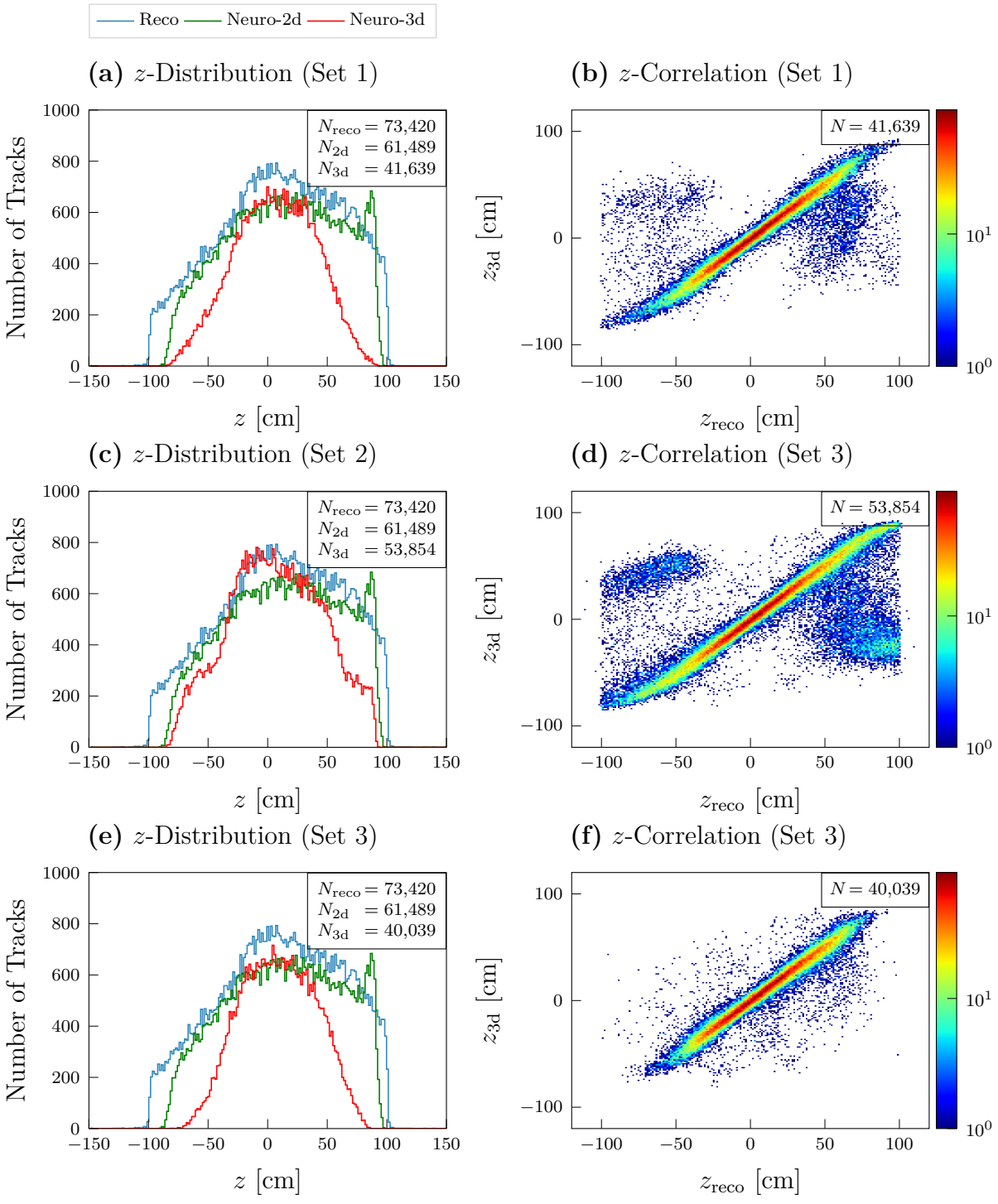


Figure 7.15: The complete z-distributions of the reconstructed, neuro-2d, and neuro-3d tracks without any background. Note that the new fixed-volume clustering algorithm was used with three different settings.

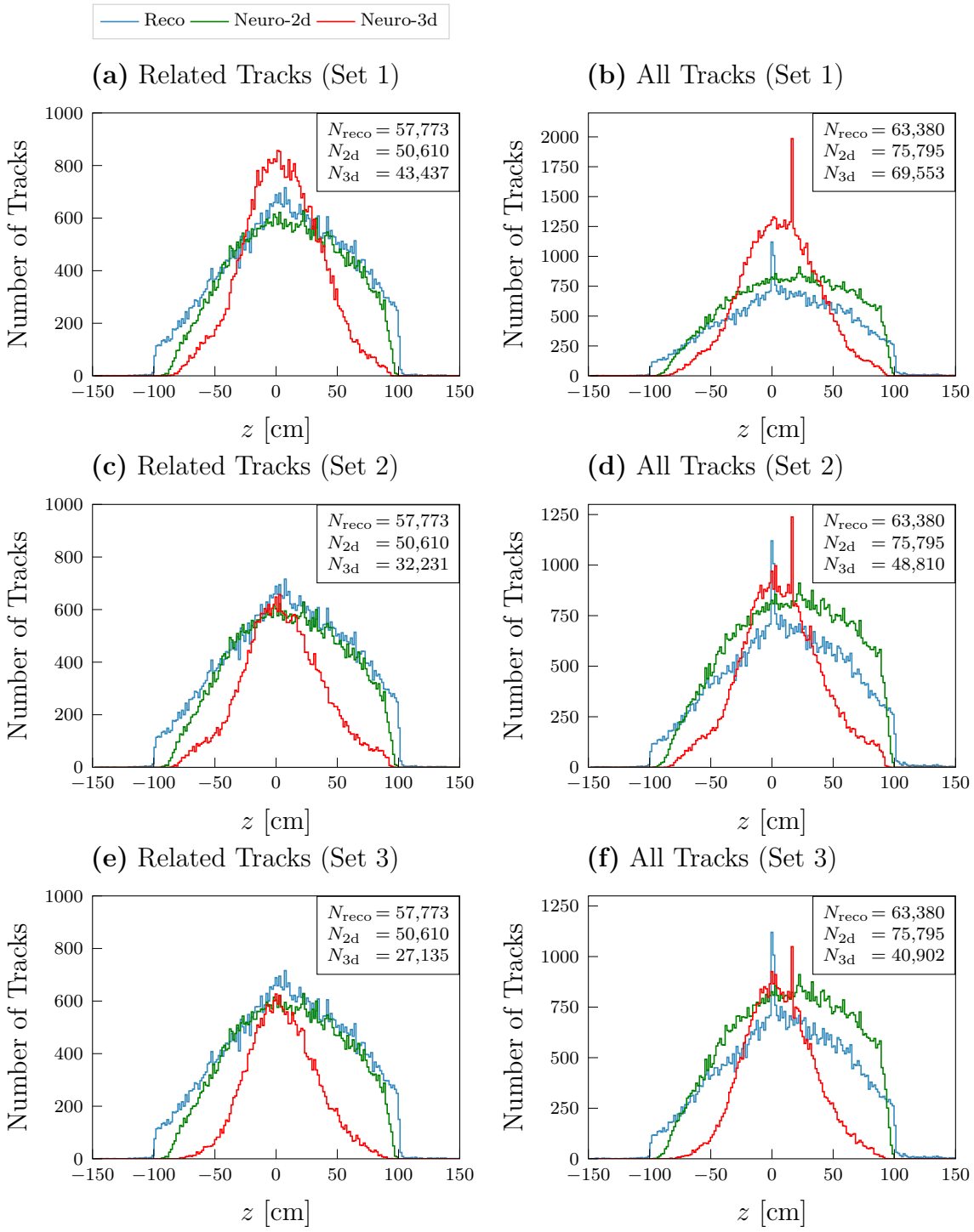


Figure 7.16: The complete z-distributions of the reconstructed, neuro-2d, and neuro-3d tracks with nominal phase-3 background. Here, the 3DFinder is using the new clustering algorithm with three different settings.

(d), and (f), all reconstructed tracks with at least 3 track segments are displayed. This can include tracks from secondary Monte Carlo particles.

The resolution of the neuro-3d tracks using Set 1 is compared to the neuro-2d resolution in Fig. 7.17. This is a notable difference when comparing this to Fig. 5.20, despite using

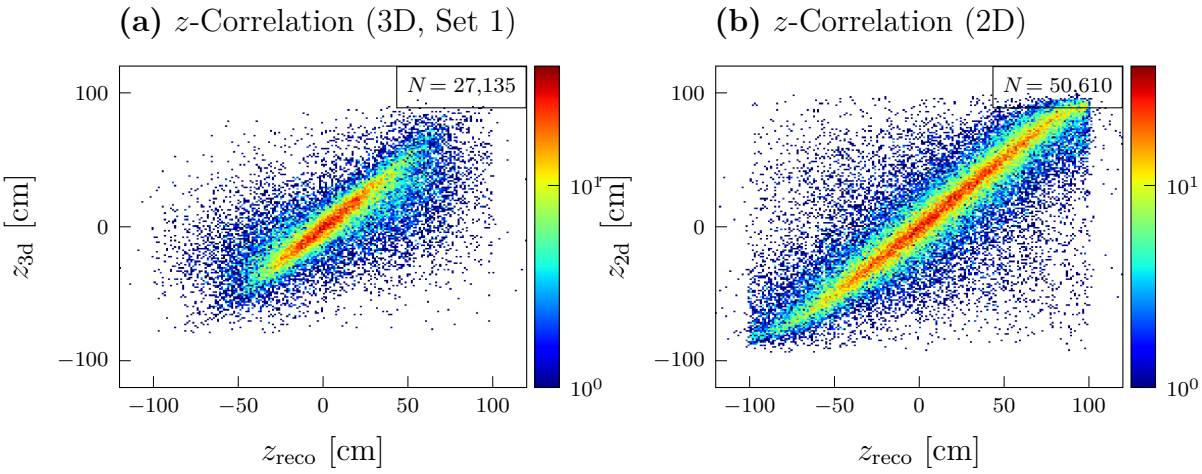


Figure 7.17: Logarithmic heatmaps of the reconstructed z against the predicted z of the neural network with 3DFinder or 2DFinder input. Here, 3DFinder is using the new clustering algorithm with parameter Set 3.

the same neural network. When retraining the neural network with high background data, this resolution should be considerably better.

Up to this point, the momentum resolutions of the neural network tracks with either 2D-Finder or 3DFinder input have not been compared. This resolution serves as an additional measure of the track candidate's quality. In Fig. 7.18, the neural network prediction of the transverse momentum using 2DFinder and 3DFinder input is plotted against the corresponding reconstructed value. Note that the parameter Set 3 was used for the 3D-Finder, with the result that the neural network prediction of p_T is considerably better. This can be explained by the increase in the total number of ω-bins and the better track segment selection. As expected, the resolution drastically decreases when considering tracks with a momentum above $3\,\mathrm{GeV}/c$, as the track is nearly a straight line.

To check the effect of the `mintotalweight` cut of 450 and the `minpeakweight` cut of 27, the weight distributions of parameter Set 2 are plotted in Fig. 7.19 as those settings were the most efficient. The blue distribution includes all tracks found with nominal phase-3 background, while the green distribution only includes the subset of tracks that were successfully related to reconstructed tracks. The red distribution consists only of tracks from the signal dataset where no simulated background was present. Both weight cuts do not have an impact on track-finding efficiency. While the two parameters may not seem important when using the `minsuper` cut, they are crucial for determining the creation of clusters in the first place. The fewer clusters are created and the faster the clustering algorithm terminates, the fewer clusters have to be related to the track segments

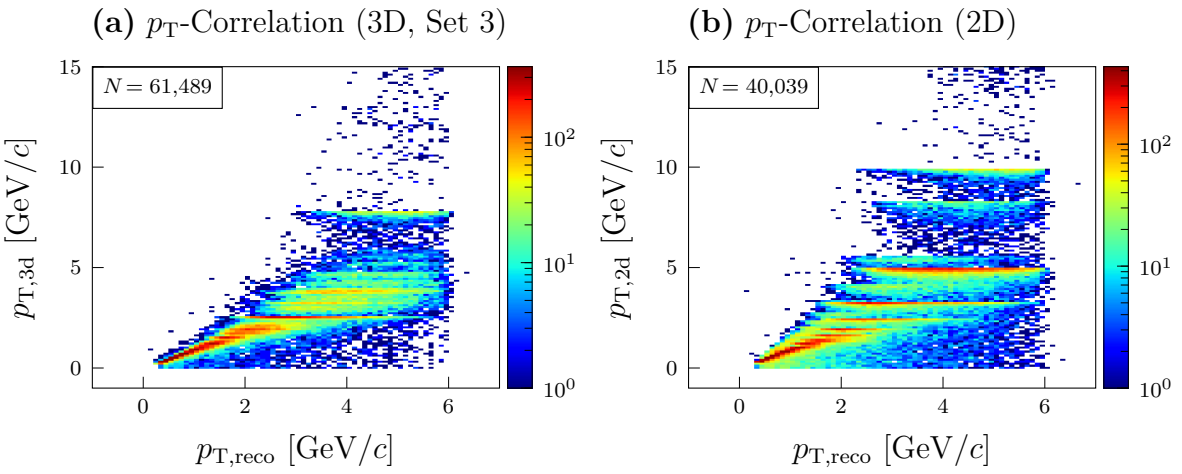


Figure 7.18: Logarithmic heatmaps of the reconstructed p_T against the predicted p_T of the neural network with 3DFinder (parameter Set 3) and 2DFinder input.

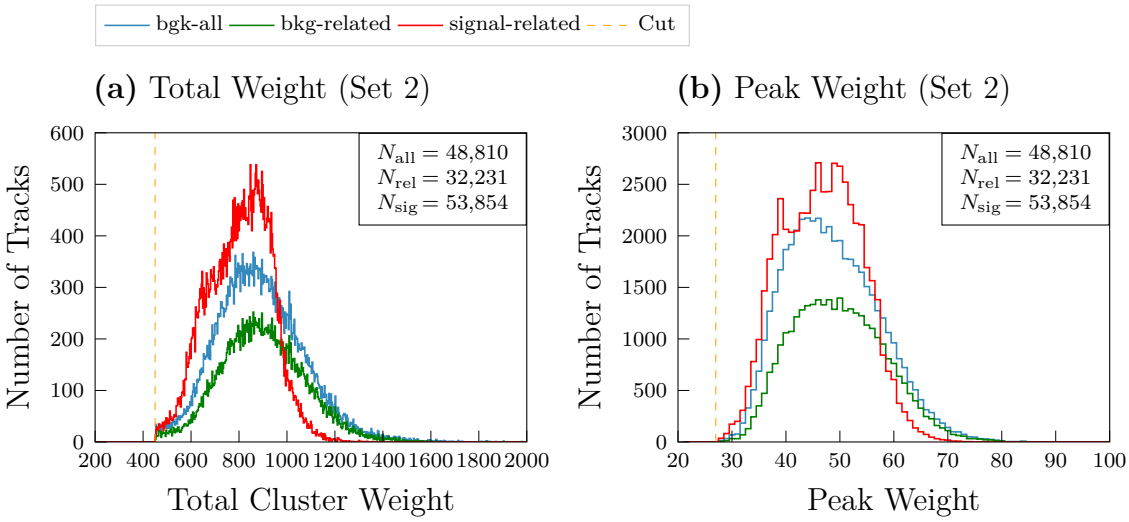


Figure 7.19: The distributions of the relative weights of real tracks and fake tracks with at least a total weight of 700. The weight of each cell is divided by the maximum, and afterwards all cells are added together.

afterwards. Furthermore, when no real cluster is present, all the small background clusters are immediately rejected, which cannot be observed in Fig. 7.19.

7.5.2 Efficiencies and Background Rates of IP Tracks

The final Monte Carlo study conducted before the new clustering algorithm is applied to real data is a large statistic of fake rates and track counts of the 3DFinder with different settings. Since the neural network is not retrained on high-background data and consists of only a single hidden layer, percentages are not provided as they would lack any meaning. Only the total track counts found by the finding algorithms are used for an efficiency estimation. In Chap. 8, the final percentages of real data single track events are provided for a neural network with three hidden layers that has been trained on the most recent experiment 26 data. In this subsection, three datasets are considered: IP tracks without background, IP tracks with nominal phase-3 background, and a sample of pure nominal phase-3 background with no real tracks.

Using Tab. 5.7, a total of 10,000 muons were generated from the IP within the full CDC acceptance range. The first generation was performed without background. In Tab. 7.6, the count of all found tracks, N_{all}, and the count of all tracks successfully related to a reconstructed track, N_{rel}, are listed for 7 different 3DFinder settings. For comparison, the

Table 7.6: Background rates and efficiencies of 10,000 single particle gun tracks originating from the IP within the full CDC acceptance range for different 3DFinder settings. No simulated background was added to this dataset.

Clustering	`mh`	`ms`	`mpw`	`mtw`	N_{all}	N_{rel}
DBSCAN	4	0	—	—	9984	9918
DBSCAN	6	0	—	—	9946	9882
Fixed-Volume	4	0	0	500	9975	9909
Fixed-Volume	6	0	43	730	9803	9748
Fixed-Volume	6	6	43	730	9803	9748
Fixed-Volume	6	6	0	500	9961	9897
Fixed-Volume	5	5	0	500	9948	9888
Neuro-2d					9868	9834
Reconstruction					10,033	9995

counts of the neuro-2d tracks and the offline reconstruction counts for the same dataset are added to the table. The abbreviations `mh` for `minhits`, `ms` for `minsuper`, `mpw` for `minpeakweight`, and `mtw` for `mintotalweight` are used. Here, DBSCAN is again compared with the new fixed-volume clustering algorithm. Since there is no background in the dataset, nearly all efficiencies are at approximately 99%. Only the efficiency of the harsh weight cut settings experiences a slight reduction. In this case, the introduction of the `minsuper` cut does not change the number of found tracks. As before, `minsuper` = 6 considers all 9 super layers, while `minsuper` = 5 only applies to the first 6 super layers.

While high efficiencies are expected, the fake rates and track counts in the presence of

background, which are listed in Tab. 7.7, are much more interesting. When considering

Table 7.7: Background rates and efficiencies of 10,000 single particle gun tracks from the IP within the full CDC acceptance range for different 3DFinder settings. Nominal phase-3 background was added to this dataset.

Clustering	`mh`	`ms`	`mpw`	`mtw`	N_{all}	N_{rel}
DBSCAN	4	0	—	—	41,524	9290
DBSCAN	6	0	—	—	19,250	7908
Fixed-Volume	4	0	0	500	24,997	9385
Fixed-Volume	6	0	43	730	11,304	9178
Fixed-Volume	6	6	43	730	10,253	9164
Fixed-Volume	6	6	0	500	10,409	8980
Fixed-Volume	5	5	0	500	8900	7750
Neuro-2d					11,051	9142
Reconstruction					12,981	9970

N_{all}, which includes fake tracks as well, it becomes evident that the `minsuper` cuts significantly reduce the number of fake tracks. The fixed clustering with the total weight cut of 500 alone reduces the number of fake tracks significantly when compared with the `minhits` = 4 DBSCAN clustering algorithm. However, only 7750 tracks could be related to the reconstructed tracks for the last parameter set. As the module for track-relation compares the track segments of a trigger track with the reconstructed track segments in the current event, an unsuccessful relation strongly indicates that the trigger track is indeed a fake track, as such a fake track consists of totally different track segments. Comparing this with the neuro-2d rate of 9142 reveals an issue with the `minsuper` = 5 cut. When considering the high efficiency in Tab. 7.6, the tracks cannot have been rejected by the `mintotalweight` threshold, as background only increases the potential total cluster weights in the Hough space. Thus, the `minsuper` cut of 5 hits in the first 6 super layers may be too harsh of a threshold in this Monte Carlo dataset. The super layer and weight cuts will be optimized on real data in Chap. 8.

To investigate the fake rates without any signal track present, a sample with only nominal phase-3 background was produced[1]. As some real tracks are present in the background simulation, they should, of course, be found by the full offline reconstruction. These tracks should not be considered fake tracks when found by the 3DFinder. The z-distribution of the reconstructed tracks from pure nominal phase-3 background with at least three different super layers related to a track is displayed in Fig. 7.20 (b). Note that most reconstructed tracks do not have any related track segments as they have been reconstructed by the VXD (see subfigure (a) and Sec. 5.4). For this reason, neurotracks that have been successfully related to any reconstructed tracks with at least 3 track segments are listed as N_{rel} in Tab. 7.8. For the `minsuper` cut, only a fake rate of 6% is observed, which is a big improvement compared with the 290% observed when using the original 3DFinder with default parameters. This improved fake rate is comparable with the neuro-2d tracks.

[1]This dataset is created by generating 10,000 neutrinos with the particle gun instead of muons, which do not interact with the detector.

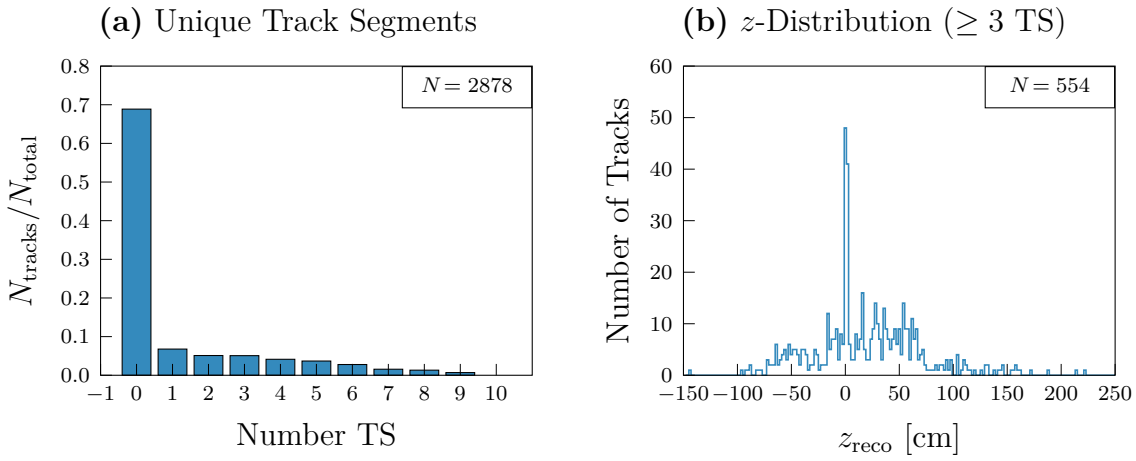


Figure 7.20: In subfigure (a), the number of hit super layers of the reconstructed tracks where only nominal phase-3 background was present in the event generation is displayed. In subfigure (b), the z-distribution of all reconstructed tracks with at least 3 unique track segments is plotted.

Table 7.8: Background rates and efficiencies using only nominal phase-3 background for different 3DFinder settings.

Clustering	`mh`	`ms`	`mpw`	`mtw`	N_{all}	N_{rel}
DBSCAN	4	0	—	—	29,424	428
DBSCAN	6	0	—	—	11,350	306
Fixed-Volume	4	0	0	500	19,156	402
Fixed-Volume	6	0	43	730	1365	178
Fixed-Volume	6	6	43	730	569	166
Fixed-Volume	6	6	0	500	924	254
Fixed-Volume	5	5	0	500	783	186
Neuro-2d					914	260
Reconstruction					2878	554

In Fig. 7.21, the z-distributions of the neuro-3d tracks with fixed clustering (subfigure (a)) are compared to the neuro-2d distributions (subfigure (b)). Given that the tracks originate

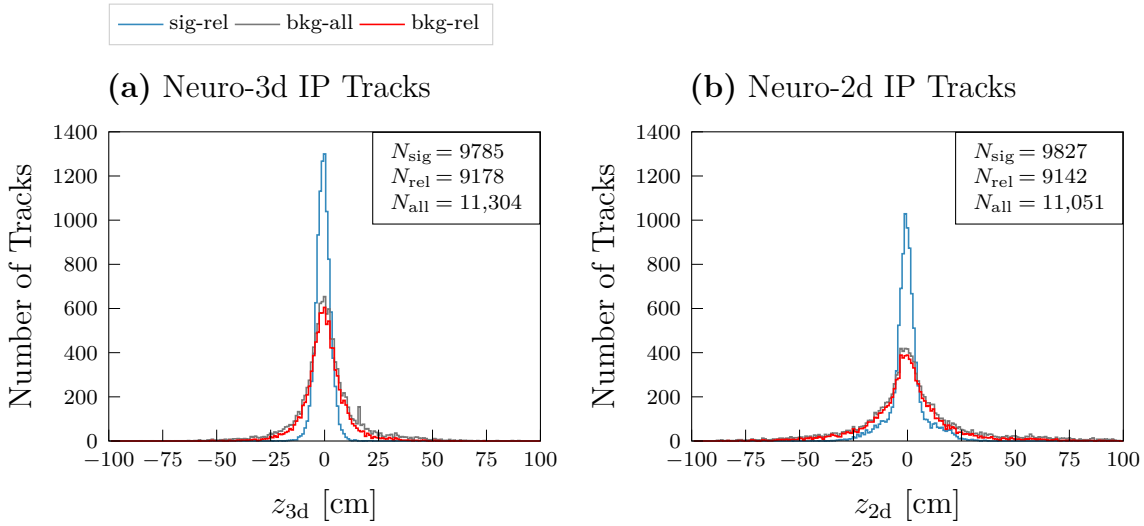


Figure 7.21: The z-distribution of the new clustering algorithm for tracks from the IP ($z_{\text{MC}} \in [-1, 1]\,\text{cm}$, full CDC) compared to the neuro-2d distribution. Note that sig-rel means tracks without any background related to reconstructed tracks, while bkg-all/rel are the complete/related distributions with nominal phase-3 background present.

from the IP, the z-distribution closely approximates the z-resolution. As expected, the z-distribution, and therefore the z-resolution, deteriorate in the presence of background. It is important to note that the resolution of the neuro-3d tracks is considerably better than that of the neuro-2d tracks. This was not the case in Sub. 5.3.3 with the original 3DFinder implementation.

Chapter 8

Optimization on Real Data

While the Monte Carlo studies were suitable to assess the basic capabilities of the 3DFinder algorithm in a controlled environment, real data studies are ultimately the only way to optimize the new 3DFinder. Any inadequacies that may appear in the Monte Carlo simulation are not present in the real data samples of the Belle II detector. Furthermore, the distributions of the track parameters and the background levels are very different from the Monte Carlo simulations. In this chapter, the last runs in experiment 26 are used for the last parameter optimization of the 3DFinder, which are characterized by high background contributions. This includes a new volume for the fixed cluster deletion shape. Additionally, the hit-to-cluster association algorithm is simplified and redone by analyzing single track events in the real data.

8.1 Fixed-Volume Clustering on Real Data

To get an overview of the real data, the first data sample consists of the last 10 runs of experiment 26. Throughout this chapter, the "neuroskim" data is used, i.e., the events with a positive L1 trigger decision. Hence, the data is biased as a lot of background events have already been filtered out since the z-vertex prediction of the Neuro Trigger was active with a cut on $z \in [-15, 15]\,\mathrm{cm}$. The 10 runs contain 49,789 events where 43,568 reconstructed tracks with at least 3 track segments were found. The z-distribution of these tracks is plotted in Fig. 8.1. Approximately 65% of the tracks originate from the IP as a result of the active L1 trigger, dominated by the Neuro Trigger. When applying a cut based on the requirement of a minimum of 5 track segments, a total of 31,102 reconstructed tracks remain. In Fig. 8.2, the p_T- and θ-distributions of the reconstructed tracks are displayed. The majority of tracks have a transverse momentum below $1\,\mathrm{GeV}/c$. While the momentum resolution for the 3DFinder is optimal in this domain, some tracks may be lost due to high crossing angles α in the outer super layers. When considering the θ-distribution, more reconstructed tracks for lower angles are observed. This is expected since those angles describe the forward direction of the detector, through which the particles are more likely to fly as the $7\,\mathrm{GeV}$ electrons are boosted in the positive z-direction.

To initiate the analysis, the z-distributions of four different 3DFinder settings, fed through the standard neural network, are compared. Here, the dataset described above is used (see Fig. 8.1). In Fig. 8.3, these distributions are compared with the neural network predictions when 2DFinder input is used. Note that only neurotracks that are related to a reconstructed track are plotted. The number of identified related tracks and the total track counts for the different settings are listed in Tab. 8.1. This data offers a few valuable insights. First, the 3DFinder efficiency seems to be below the 2DFinder efficiency for each setting. When using `minsuper` = 5 and `iterations` = 5, nearly twice as many 2DFind-

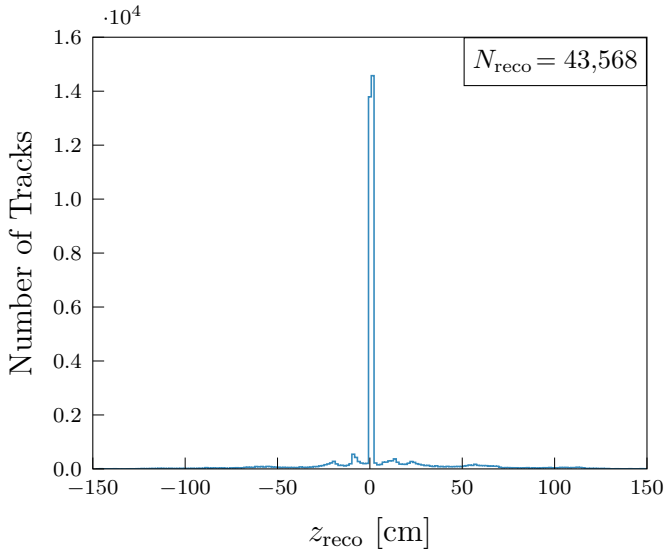


Figure 8.1: The z-distribution of the reconstructed tracks with at least 3 track segments in the real data sample.

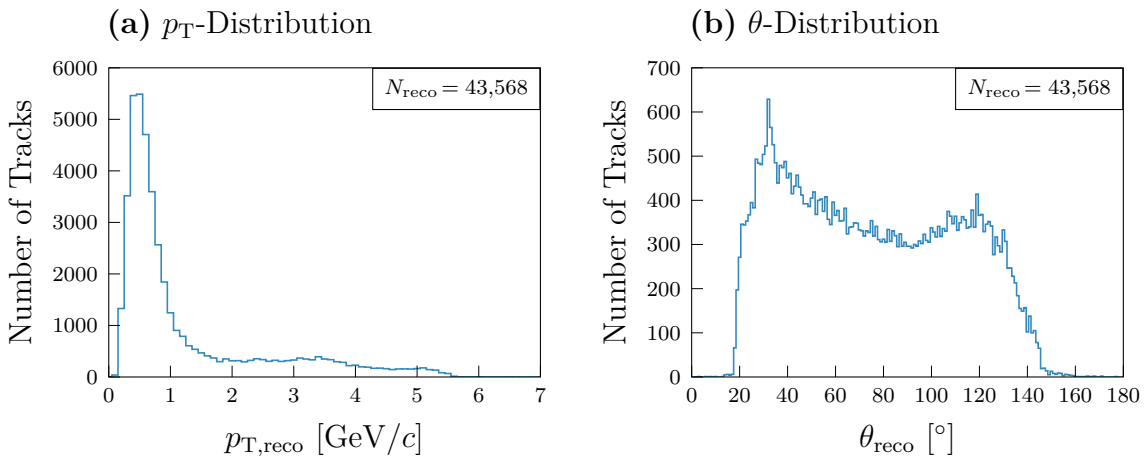


Figure 8.2: The p_{T}- and the θ-distribution of reconstructed track with at least 3 track segments in the real data sample.

Table 8.1: Comparison of the related and total track counts for four different 3DFinder settings.

Algorithm	**Clustering**	`mw`	`ms`	`mpw`	`mtw`	`iter`	N_{3ts}	N_{all}
3DFinder	DBSCAN	26	5	—	—	—	17,926	30,723
3DFinder	Fixed-Volume	—	5	27	450	5	16,368	26,158
3DFinder	Fixed-Volume	—	4	27	450	5	20,429	55,010
3DFinder	Fixed-Volume	—	5	27	450	10	17,458	29,202
2DFinder	—	—	—	—	—	—	29,632	60,377
Reco	—	—	—	—	—	—	43,568	195,537

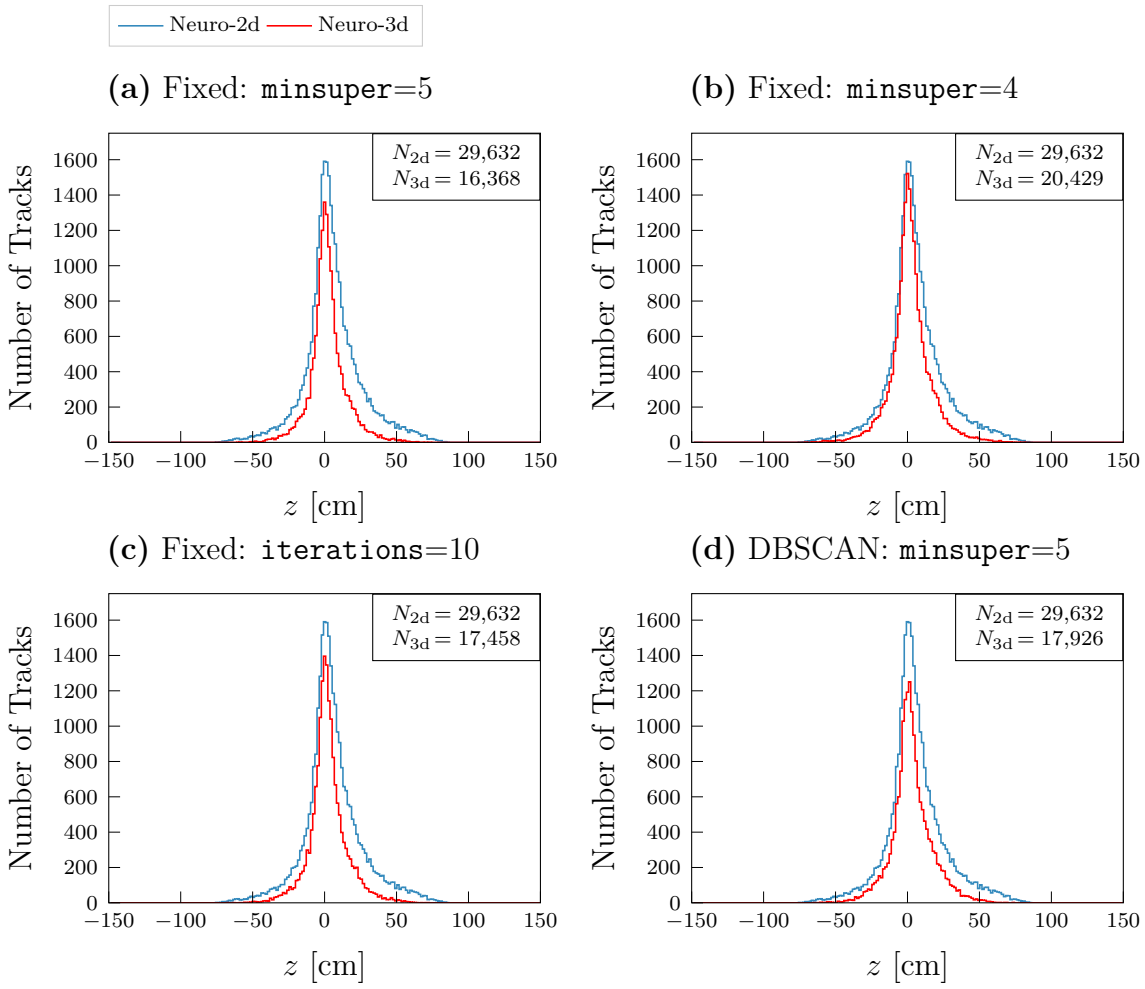


Figure 8.3: The z-distribution of the neuro-2d and neuro-3d tracks that are related to a reconstructed track with at least 3 track segments in the real data sample.

er tracks were found. While the 3DFinder strongly suppresses tracks that have a large displacement in z, there are too few of such tracks in the data sample to explain the large difference. While at most 5 tracks can be found per event with `iterations` = 5, the lack of efficiency may be attributed to high multiplicity events. This turns out to be an incorrect assumption, as an increase to `iterations` = 10 only slightly increases the number of found tracks. Moreover, when using the DBSCAN algorithm, which does not have an upper limit for cluster candidates, the efficiency is not much better. When reducing the `minsuper` cut to 4, the efficiency increases the most, but as a consequence, the fake rate more than doubles (see Tab. 8.1).

To further address this apparent efficiency loss, the kinematic distributions of the neuro-2d and neuro-3d tracks are compared. Fig. 8.4 (a) shows the θ-distribution of the tracks. Note that only the `minhits` = 5 settings of the 3DFinder are displayed. It is evident that the 3DFinder loses tracks across the entire polar angle range. The same is the case when considering the transverse momentum distribution in subfigure (b). This strongly indicates that the lack of track-finding efficiency is a global problem for the 3DFinder.

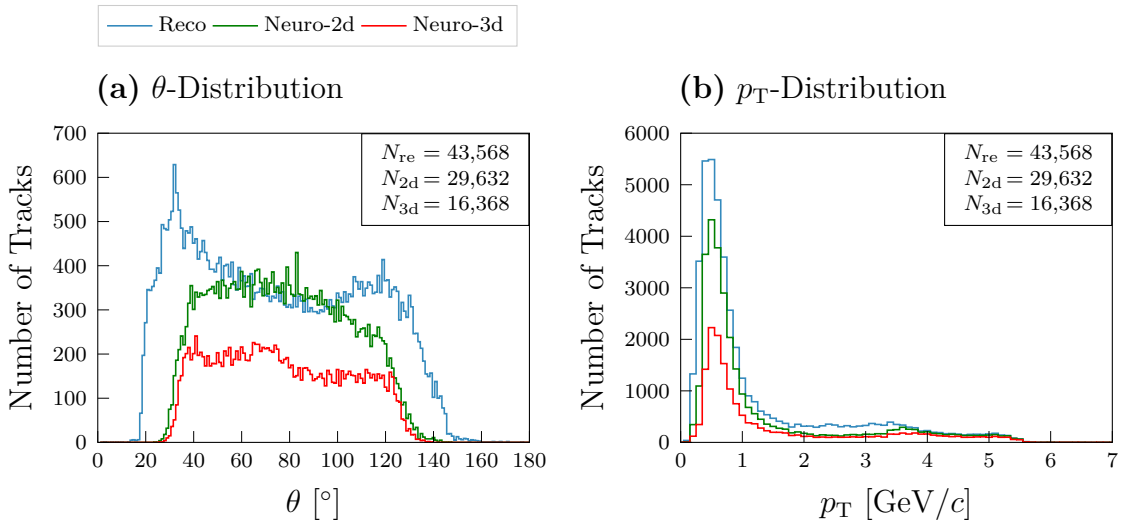


Figure 8.4: The θ- and p_T-distribution of the neuro-2d and neuro-3d tracks compared with the reconstructed distributions. Note that the 3DFinder is using the `minsuper` = 5 settings.

A possible explanation could be overly aggressive cuts on the cluster weights, leading to many rejected clusters. In Fig. 8.5, the total cluster weight and the peak weight distributions are displayed for the `minsuper` = 5 setting. Hence, the cuts do not cause the

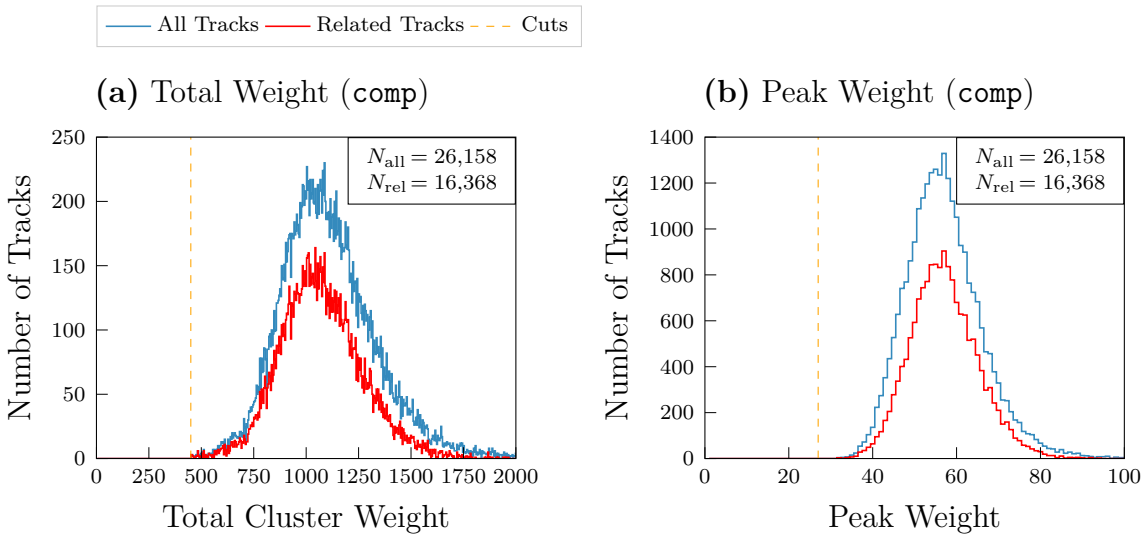


Figure 8.5: The total weight and peak weight distributions of the 3DFinder clusters with their corresponding cuts. Here `minsuper` = 5 was used.

poor efficiency. The peak weight cut may even be increased from 27 to 32. Lastly, the different super layer numbers of each track are investigated. Fig. 8.6 shows the distribution of all track segments and the unique track segments of the 3DFinder in subfigure (a). Nearly all super layers are hit, while the outermost are slightly reduced. This is expected as a lot of tracks have low transverse momentum and shallow θ-angles. Thus, the inefficiency of the 3DFinder cannot be easily explained. For comparison, the neuro-2d track segments

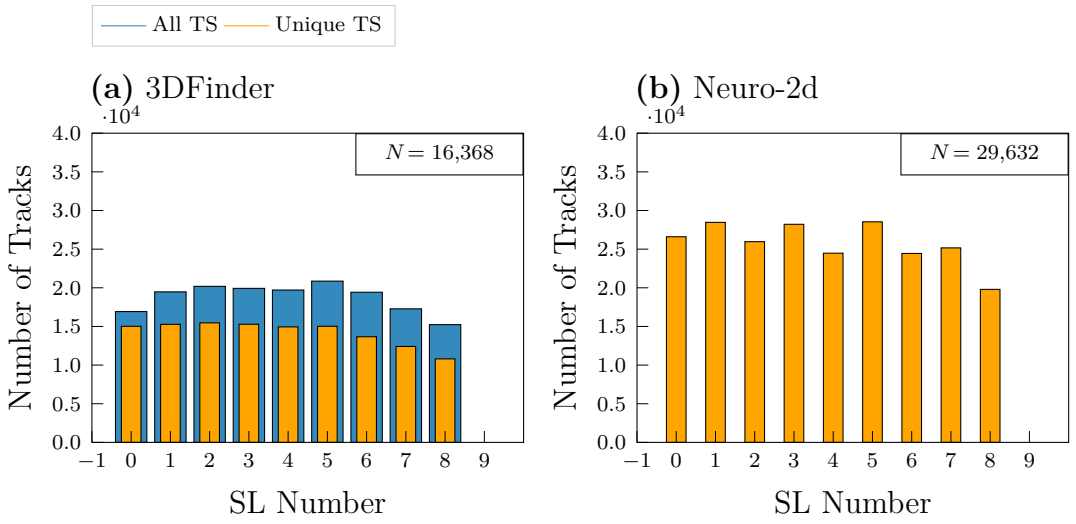


Figure 8.6: How often each super layer occurred in a neuro-3d and neuro-2d track related to a reconstructed track with at least 3 track segments in the real data sample.

are displayed in subfigure (b), which are trivially unique. Note that the stereo super layers 1, 3, 5, and 7 are overrepresented as the Neuro Trigger demands at least 3 stereo track segments as input when using the 2DFinder.

Since the neural network trigger should be most efficient for IP tracks, only tracks related to reconstructed tracks with $p_\mathrm{T} \geq 250\,\mathrm{MeV}/c$ and $z_\mathrm{reco} \in [-1, 1]\,\mathrm{cm}$ are considered next. For this purpose, the dataset is extended to the last 50 runs of experiment 26. In Fig. 8.7 (a), the z-distribution of the IP tracks is displayed. Considerably fewer IP tracks were found by the 3DFinder (273,701) compared with the 2DFinder (392,566), although the resolution of the neural network is better when using 3DFinder input. However, as the

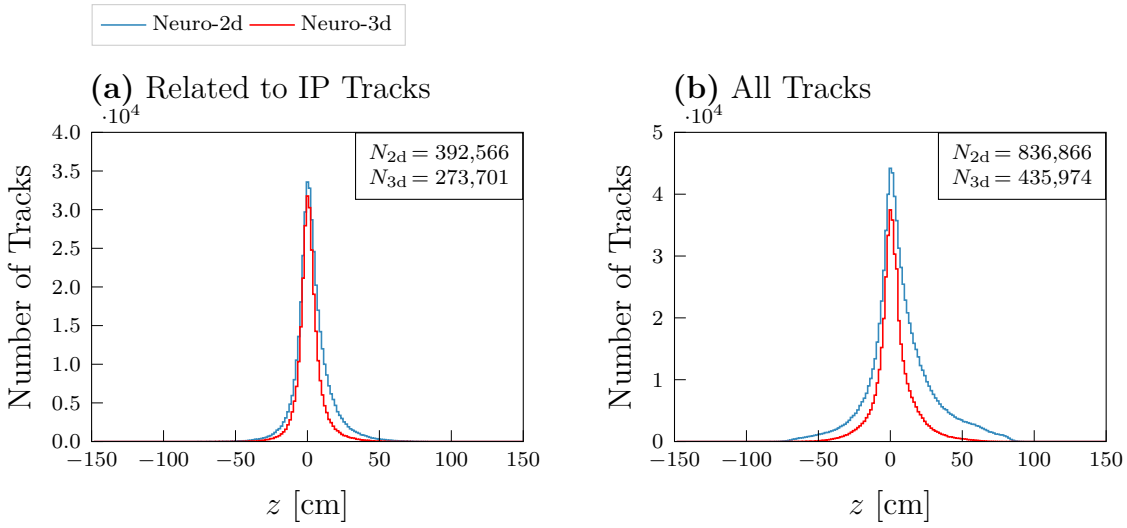


Figure 8.7: The z-distributions of the neuro-3d and neuro-2d tracks of the last 50 runs of experiment 26. In subfigure (a), only IP tracks are included, while in subfigure (b), all found tracks are plotted.

resolution will increase when a better network architecture, retrained with newer data, is used, the efficiency is considerably more important than the resolution. In subfigure (b), the z-predictions of all neurotracks are displayed. The neuro-3d tracks fall off considerably faster and include fewer fake tracks.

To analyze the background suppression of displaced tracks, i.e., all tracks originating from $z \notin [-1, 1]$, the reconstructed tracks that the 3DFinder and 2DFinder were capable of finding are plotted in Fig. 8.8, respectively. Note that the shape of the reconstructed z-

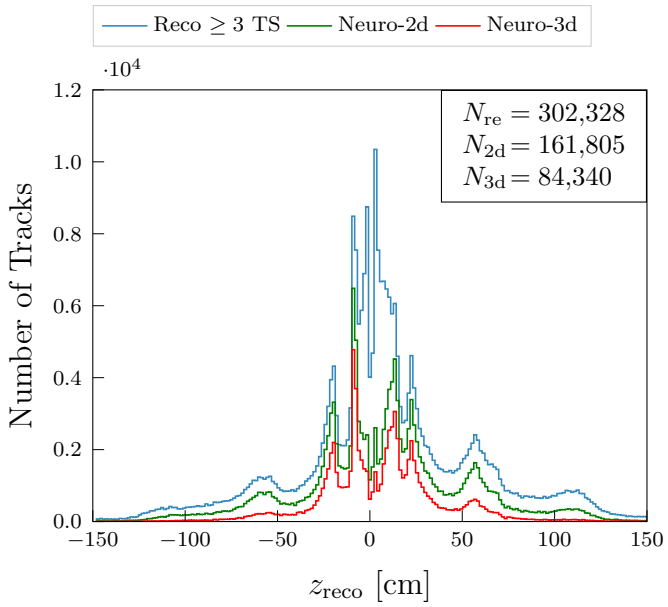


Figure 8.8: The z-distributions of the reconstructed tracks that the 3DFinder and the 2DFinder actually found in the last 50 runs of experiment 26. Note that all tracks from $z_{\text{reco}} \in [-1, 1]\,\text{cm}$ are excluded for better visibility.

distribution is dictated by the machine layout. The final focusing magnets extend very close to the IP ($\approx 50\,\text{cm}$) [5], causing the characteristic peaks in the background z-distribution. At $z \approx \pm 50$, already half of the tracks the 2DFinder detected are not found by the 3DFinder. The further displaced the background tracks are, the stronger the suppression from the 3DFinder is. This confirms that the suppression observed in the Monte Carlo study applies to real data as well. It is important to note here again that the tracks are biased as the Neuro Trigger was active during the data collection. To properly assess the background suppression, an unbiased data sample is necessary.

In conclusion, while the resolution and the background suppression are satisfactory, the efficiency of the 3DFinder seems to be very poor compared to the 2DFinder. This has to be addressed by studying single track events in the real data.

8.2 The New Hit-To-Cluster Association

In the previous section, the apparent efficiency loss of the 3DFinder could neither be fully explained by high multiplicity events (see Fig. 8.3 (c)) nor by too harsh weight cuts (see Fig. 8.5). An important part of the 3DFinder algorithm has not been considered up to this point.

After every possible cluster is found in an event, the hit-to-cluster association algorithm assigns track segments to each cluster. Only when those track segments pass the `minsuper` cut will the track be accepted. For this reason, the hit-to-cluster association has to be analyzed in depth when faced with an efficiency problem. While the original algorithm has already been briefly mentioned in Sub. 5.1.2, a more thorough examination is conducted in this section. Note that the hit-to-cluster association algorithm is independent of the clustering algorithm. Whether the clusters have been created by the DBSCAN algorithm or the new fixed-volume clustering is irrelevant. Before performing the final optimization of the 3DFinder, however, a retrained neural network with a deep learning architecture is introduced.

8.2.1 The Retrained Neural Network

To get a more realistic estimate of the efficiencies, a new neural network is used. This network consists of 3 hidden layers with 100 nodes each (3HL) and was trained on the experiment 26 data [21]. When comparing this to the standard neural network with only one hidden layer with 81 nodes that was trained on old data with lower luminosity, a big improvement in the z-resolution is expected. It is very important to mention here that, like the standard neural network, the new network has only been trained with 2DFinder track candidates. Since the optimization of the 3DFinder is incomplete and an efficiency problem was indicated in the previous section, it was not utilized for the training[1]. This means that the network is slightly biased in favor of the 2DFinder input. In Fig. 8.9, the z-resolution using IP tracks of the last 50 runs of experiment 26 of the standard neural network is compared with that of the new neural network. All y-axes are scaled to the same values so that the distributions are comparable. As in Chap. 5, a double Gaussian is fitted to the data. It is immediately evident that the resolution is considerably better when using the 3HL network. The core Gaussian standard deviation of the neural network prediction with 3DFinder input gets reduced from 3.33 cm to only 0.87 cm (subfigures (a) and (b)). Similarly, the standard deviation of the side Gaussian is only 3.48 cm, compared to 9.24 cm when using this new network. While the resolution of the neuro-2d tracks increases just as much (subfigures (c) and (d)), the z-resolution of the neuro-3d tracks is still better despite the network being only trained on 2DFinder track candidates. This indicates that the track candidates of the 3DFinder are better in a fundamental sense. The track segment selection and the parameter calculation are better than in the 2DFinder case, where the stereo track segments are selected separately.

For the standard neural network, the cut position of the z-prediction implemented at the L1 trigger was ± 15 cm. The new network now allows for an efficiency calculation with a cut position of ± 10 cm, as the z-resolution for both inputs is considerably better. For the remainder of this thesis, only this 3HL network will be used.

8.2.2 Single Track Event Analysis

In this section, a single track event is an event where exactly one reconstructed track, originating from the IP ($z \in [-1, 1]$ cm) with a sufficient number of track segments, was

[1] The performance of networks specifically trained with the input of the final 3DFinder can be found in [21].

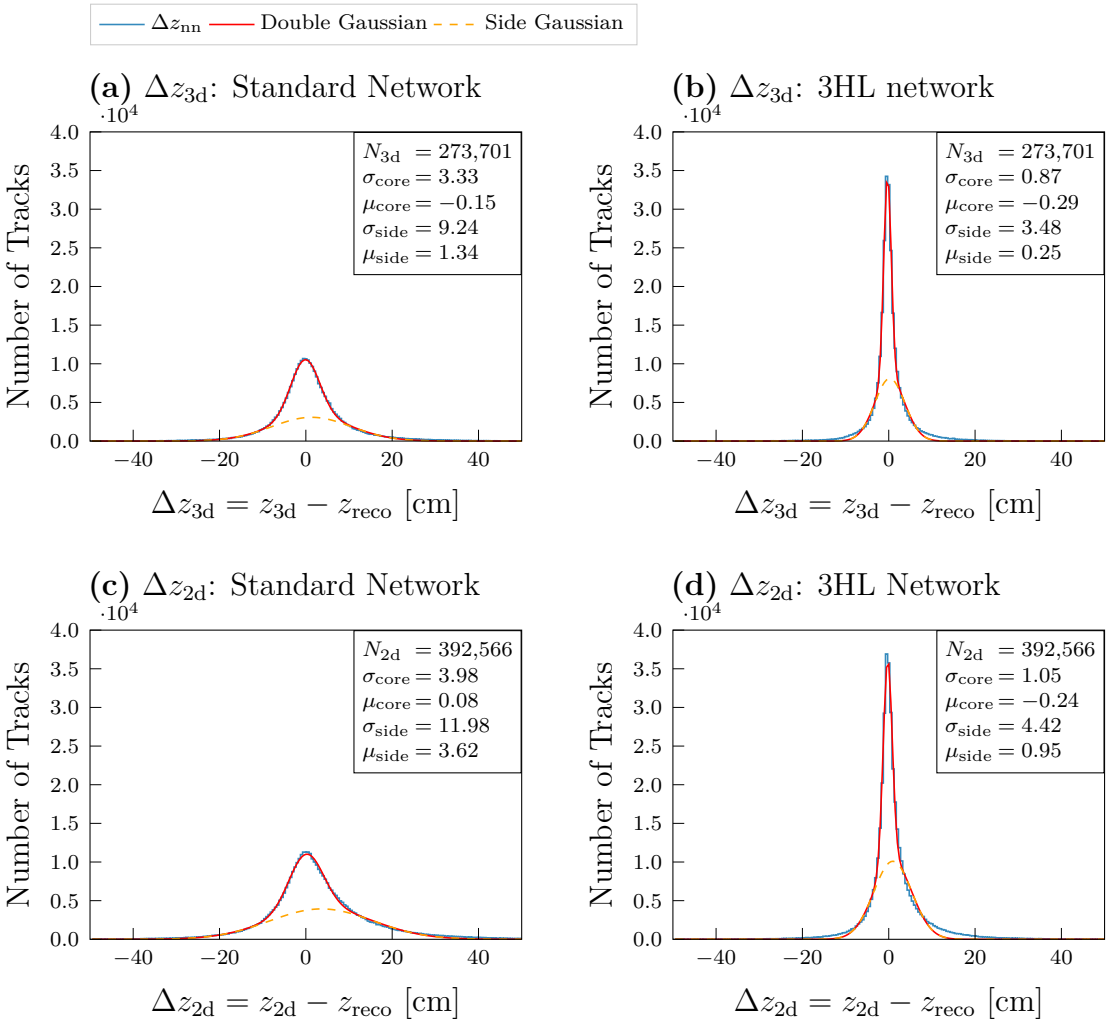


Figure 8.9: The z- and θ-resolutions of the neuro-2d and neuro-3d tracks that are related to a reconstructed track with at least $p_{\text{T}} \geq 250\,\text{MeV}/c$ and $z_{\text{reco}} \in [-1, 1]\,\text{cm}$. The data used is from the last 50 runs of experiment 26. The new neural network with three hidden layers is compared with the old neural network.

found. This means that the reconstructed track has at least 4 unique axial track segments and 3 unique stereo track segments to match the requirements of the track finders. To get an idea of the observed background levels in experiment 26, two arbitrary single track event Hough spaces are displayed in Fig. 8.10. When comparing the number of fake track

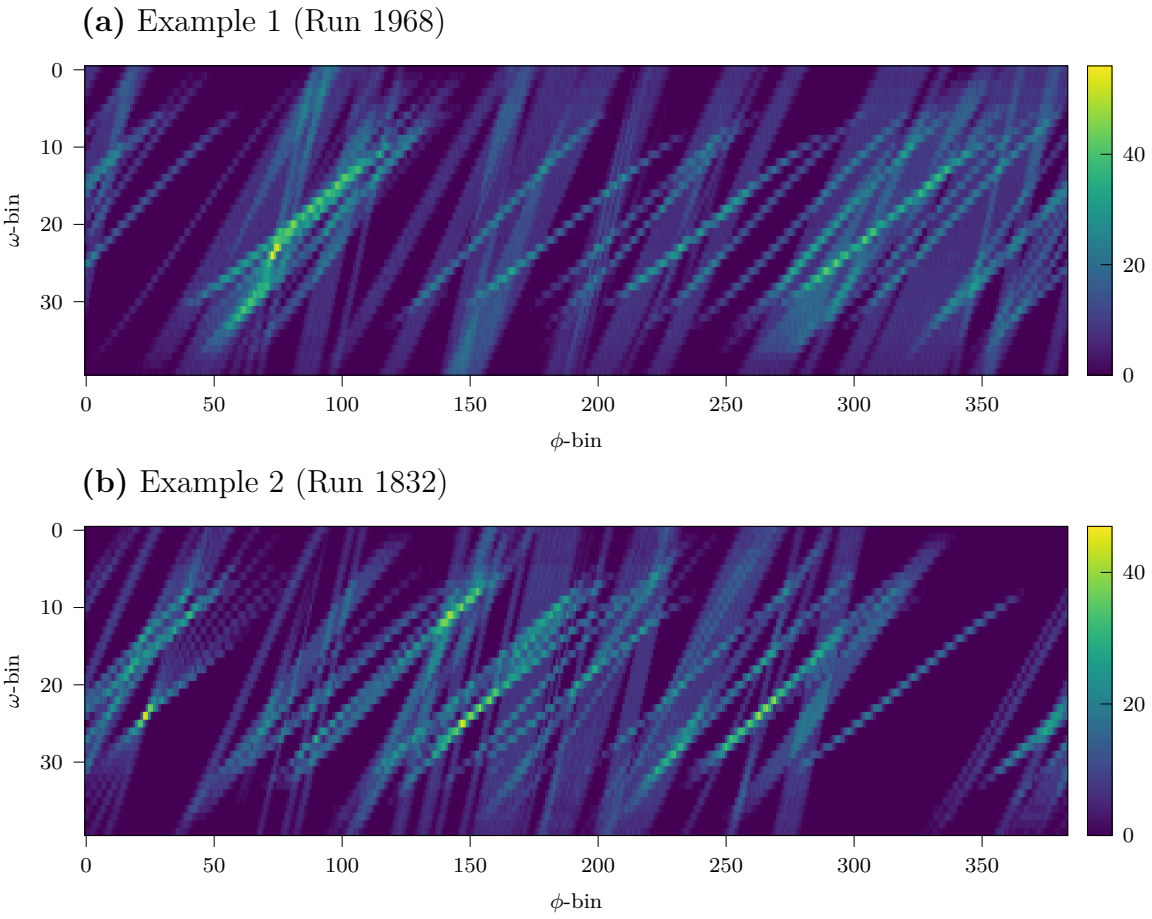


Figure 8.10: Exemplary Hough spaces from two different experiment 26 single track events.

segments with the simulated nominal phase-3 background in Fig. 6.6, a significantly worse background is observed in the real data. The two exemplary Hough spaces yield several significant insights. In subfigure (a), the global maximum created by the real single track is approximately at $(\phi, \omega) = (75, 25)$. This real cluster is directly above a lot of fake background track segments. As the Hough space is filled throughout the whole ϕ- and ω-range with track segments mainly from background the likelihood of such an overlay is very high. Hence, it is impossible to use the cluster shape, e.g., relative weights, to determine whether a cluster is signal or background. In subfigure (b), the real track can be found at $(\phi, \omega) = (25, 25)$, because of the characteristic "fanning" of the track segments that originate from multiple super layers. However, the global maximum in this Hough space is not caused by this real cluster but rather by a cluster formed from background hits ("fake cluster") at approximately $(\phi, \omega) = (150, 25)$. It is very important to note here that the fake cluster does not consist of track segments with different slopes like those observed in the real cluster. Most of the track segments must therefore be confined

to a single super layer. To get the fake rate as low as possible while still ensuring high efficiency for real tracks, the `iterations` parameter must be large enough such that the clustering algorithm does not terminate after a fake global maximum. Thus, the parameter `iterations` is kept at 5. Moreover, the super layer cuts have to be very strict when dealing with such high background levels. For this reason, the `minsuper` parameter is replaced with two new super layer cut parameters, `minsuper_axial` and `minsuper_stereo`. As the name suggests, `minsuper_axial` sets a threshold for the minimum number of axial super layers that have to be hit by a real track. Since there are 5 axial super layers, this parameter is set to 4. Similarly, the `minsuper_stereo` cut is set to 3, implying that 3 of the 4 stereo super layers must contain a hit for a real track.

All the single track events in the last 50 runs of experiment 26 are used to ultimately determine the 3DFinder efficiency. In Fig. 8.11, the z-distributions estimated by the new 3HL neural network with 3DFinder and 2DFinder input are compared. Especially when

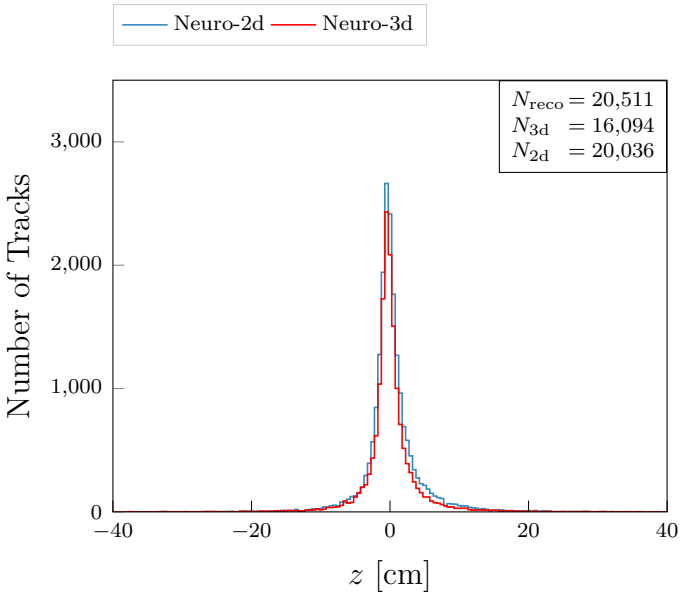


Figure 8.11: The z-distribution of the original hit-to-cluster association algorithm for all single track events of the last 50 runs of experiment 26.

considering the total track counts, the 3DFinder efficiency is significantly worse, which has already been strongly suggested in the previous section. The efficiencies after applying the z-cut of $\pm 10\,\text{cm}$ are listed in Tab. 8.2. The efficiency of the 3DFinder is 75.8%, which is completely unacceptable. The 2DFinder efficiency is better by more than 18%.

Table 8.2: Comparison of the different single track event efficiencies of the last 50 runs of experiment 26 using the original hit-to-cluster association. A found track is accepted if $z \in [-10, 10]\,\text{cm}$.

Efficiency-3D	Efficiency-2D
75.8%	94.0%

To find the reason for this unexpected problem, the single events have to be investigated in more detail. The Hough space of a single track event is displayed in Fig. 8.12. Note that

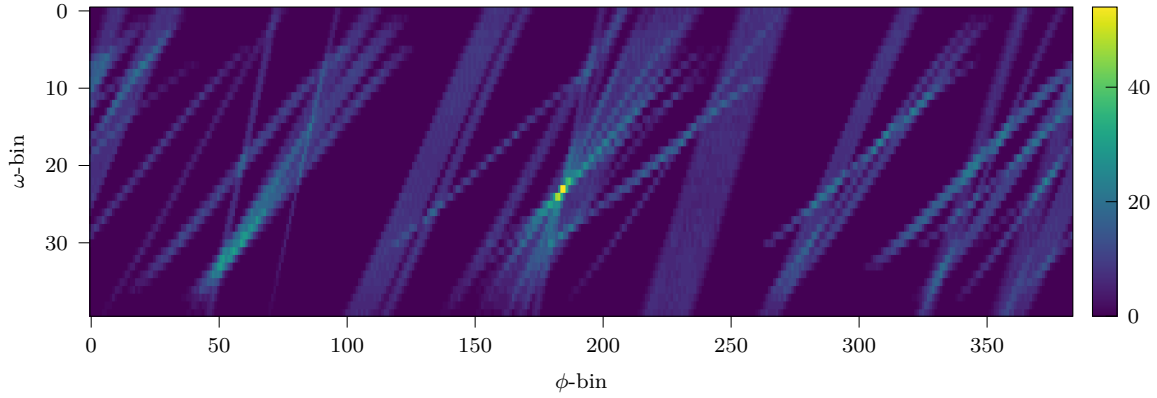


Figure 8.12: The Hough space of event 460 in experiment 26, run 1832, HLT 1, and f00001. Here θ-bin 1 is being displayed. The background does not create a higher peak than the single real track in this event.

the global maximum peak cell in this Hough space has a weight of 54 and was caused by the real single track. Here, the 2DFinder successfully found a track, creating a neurotrack with a track segment from 8 of the 9 super layers:

```
Found track segment IDs: [78, 233, 415, 631, 864, 1131, 1443, 2150]
Super layer numbers of the track segments: [0, 1, 2, 3, 4, 5, 6, 8]
```

The 3DFinder, however, did not find the single track in this event, despite the global maximum in the Hough space being caused by a real track. For this reason, the `iterations` parameter is reduced from 5 to 1. As this only creates a single cluster on the global maximum of the real track, the 3DFinder successfully finds the track with the correct track segments. Now the `iterations` parameter is slowly increased. Surprisingly, after setting this parameter to 2, the track disappeared again. The second iteration generates only one new cluster in the Hough space, centered around a different cell. Hence, this new cluster results in the first cluster failing to produce a track, despite the first cluster having already produced the correct track in the `iterations` $= 1$ case. This can only be explained by a problem in the hit-to-cluster association algorithm. In this example, the cluster is found by the clustering algorithm, but is rejected afterwards by the super layer cuts due to an incorrect association of the track segments to the cluster when multiple clusters are present.

To understand the cause of this incorrect association, the original hit-to-cluster algorithm has to be investigated. After all clusters have been created in the Hough space, a so-called "confusion matrix" is created [4]. It is important to note here again that the hit-to-cluster association is independent of the clustering algorithm. Both DBSCAN and fixed-volume clustering return a set of clusters for which the track segment association has to be determined. The confusion matrix is a two-dimensional matrix where each track segment in the Hough space gets assigned a row and every cluster a column. The corresponding weight contribution $w_{i,j} \in [0, 1, \ldots, 7]$ of the track segment i to the cluster j is written into this matrix. Now, the algorithm loops over all the track segments in this Hough space,

and tries to associate the track segments to the clusters. Suppose there are two clusters k and l in the Hough space and the track segment m is currently considered. When the hit m is associated with one and only one cluster, the hit will be associated with this cluster. However, if the hit m is present in two or more clusters, like k and l in this example, it is only associated with cluster k if

$$\frac{w_{m,k} - w_{m,l}}{w_{m,k}} > \texttt{minassign} \tag{8.1}$$

is satisfied. This means that a hit is only related to a cluster when the relative weight contribution to other clusters is not greater than the parameter `minassign`. As `minassign` was chosen to be 0.2 in the original algorithm, a hit that has the same weight contribution to more than one cluster will not get related to any cluster. Even if $w_{m,k} = 7$ and $w_{m,l} = 6$, neither cluster will get an association to the hit m. This loop is conducted over all track segments in the Hough space, attempting to associate each hit with the best cluster. All leftover track segments can, however, still be associated with a cluster. A new iteration determines small clusters[2], i.e., clusters that, up to this point, have very few associated track segments and deletes them. This not only sets the track segments that have already been associated with those small clusters free to be reassigned, but track segments that have not been associated due to these small clusters now have a second chance. After such a deletion, a new attempt is made to relate the remaining hits and the new hits of the deleted clusters to the remaining clusters. The idea here was to remove small clusters and use their track segments for different larger clusters. However, this fails when two clusters are too close to each other. When using the fixed-volume clustering algorithm on the Hough space in Fig. 8.12, the following hit-to-cluster output in Tab. 8.3 is observed. Due to `iterations` = 2, two clusters are found very close to each other as there is only

Table 8.3: Exemplary hit-to-cluster output when using `iterations` = 2 for a single track event. The green track segments were successfully related to the corresponding cluster, while the red ones were rejected.

Cluster 1			Cluster 2		
SL Number	**TS Number**	**Weight**	**SL Number**	**TS Number**	**Weight**
0	78	7	0	78	7
1	233	6			
2	415	5	2	415	5
3	628	4	3	626	7
3	629	4	3	627	2
4	864	7	4	864	7
5	1131	2			
6	1442	7	6	1442	7
6	1443	5	6	1443	5
8	2150	7	8	2150	7

[2]For this purpose, a second loop of length `minhits` − 1 is conducted. For each iteration n in this loop, clusters with less than n + `minhits` − 2 related hits are deleted, and immediately afterwards a new association is attempted.

a single track in the Hough space and no large background cluster. The green track segments have been directly associated with the respective cluster as they are unique. In contrast, the red track segments were not successfully associated as they had an equal weight distribution in both clusters. Since the `minhits` cut is replaced with a stricter `minsuper_axial` and `minsuper_stereo` cut, cluster 1 with only 4 related track segments will be deleted immediately in the first step, along with cluster 2. Consequently, no track is found.

This problem is especially prevalent for the fixed-volume clustering algorithm. Since the cluster deletion may not be large enough to delete the surroundings of the intersection, the next global maximum may be very close to the old maximum, including many of the old track segments. This is more likely to happen when high background conditions create large clusters in proximity to the real cluster. Especially single track events, where no second large peak is observed in the Hough space, can cause problems. However, when using the DBSCAN algorithm, this incorrect association is possible as well. As DBSCAN only stops clustering when all possible candidate cells have been considered, multiple clusters in proximity are unlikely. Nevertheless, when high background conditions create a considerable cluster on the tails of a real intersection, an incorrect assignment is still possible. As can be seen in Fig. 8.3 (d), the z-distribution using the DBSCAN algorithm indicated an efficiency loss as well. It has to be mentioned here that low background conditions have been used in the development of the original DBSCAN algorithm that are not comparable to the background observed in experiment 26 [4].

It is evident that such a complicated hit-to-cluster algorithm is inadequate. Not only are tracks lost when two clusters are close to each other, but the algorithm first has to find all the clusters in the Hough space. As the hardware implementation on an FPGA board is as pipe-lined as possible, waiting until the clustering of the Hough space is complete is unacceptable. Therefore, a new hit-to-cluster association algorithm is proposed. The C++ source code is provided in Appendix A.1.

In this algorithm, every cluster is considered individually[3]. For each cluster, a simple matrix is created, where the weight contribution of each hit along with its super layer number and drift time are listed. For each super layer, the track segment with the highest weight contribution is directly associated with the cluster. For example, the track segment `1442` is assigned to cluster 1 in Tab. 8.3 instead of `1443`. If multiple hits have the same weight contribution in a single super layer (in this example, `628` and `629`), the shortest drift time is used to decide which hit to use. Hence, the 3DFinder has at most a single track segment per super layer as output and does not consider any other clusters. Therefore, the Neuro Trigger does not need to make any selection of which track segments to use. With the old hit-to-cluster association, many duplicate track segments got passed to the Neuro Trigger module (see, e.g., Fig. 7.11 (a)).

Moreover, the fixed volume for the cluster deletion is refined. A new "butterfly-shape" is used to more accurately delete a cluster after it has been found. Note that the fixed volume that defines the cluster cells for the center of gravity calculation and track segment selection still stays the same (see Fig. 7.1). In Fig. 8.13, the old diagonal cluster shape

[3]In the basf2 simulation, the clustering is executed first, as there is no latency requirement. In the final hardware implementation, this new association will be done immediately after identifying a single cluster.

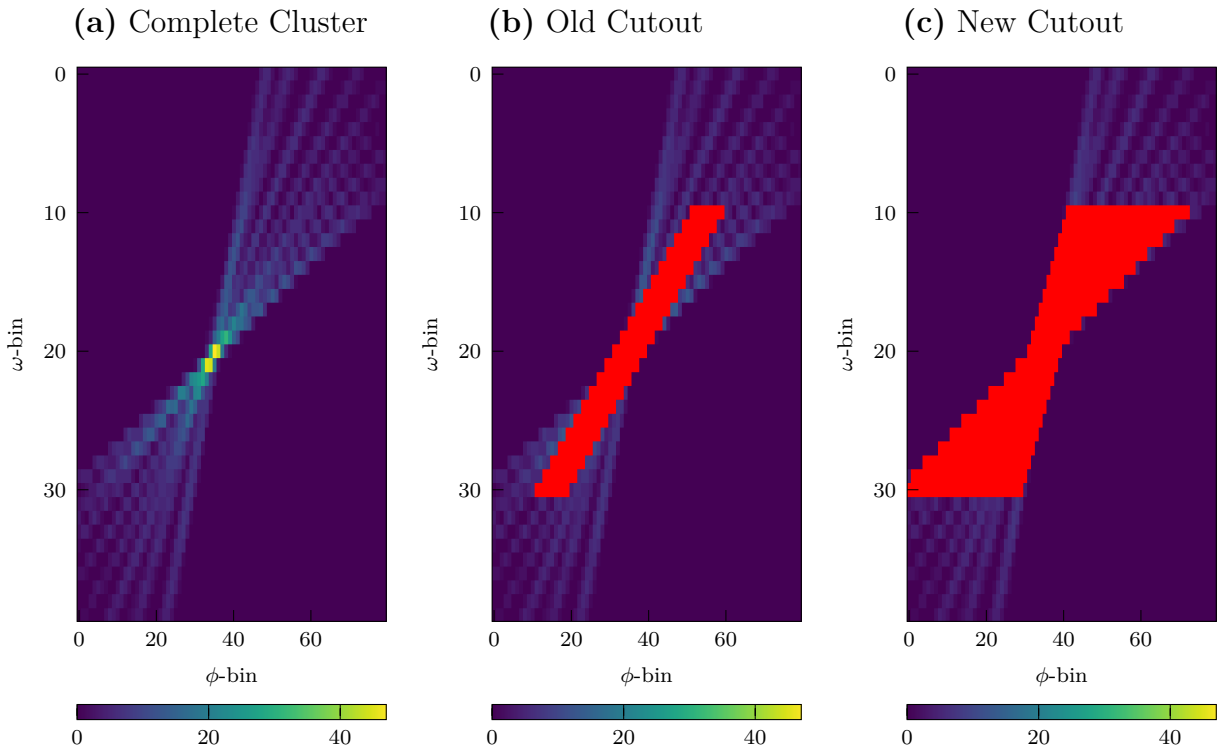


Figure 8.13: The improved cluster shape deletion around the global maximum. Here, a single particle gun track was used to create the cluster. Note that the track contains all nine super layers and `omegatrim` = 10, `phitrim` = 4.

is compared with the new shape. Note that the meanings of the three `trim` parameters remain unchanged. After the given base value, only the `phitrim` value is automatically adjusted for each ω-bin in order to delete according to the average shape in subfigure (a). In this exemplary Monte Carlo cluster, all 9 super layers have been hit, i.e., all the different potential track segment slopes are respected.

8.2.3 The 3DFinder Efficiency after Improving the Hit Association

Using the same single tracks as before, the z-distribution for the new hit-to-cluster association algorithm is displayed in Fig. 8.14. The neuro-3d peak is now considerably higher than the neuro-2d peak, although the distribution is also slimmer. After applying a z-cut of $\pm 10\,\mathrm{cm}$, the track-finding efficiency of using the 3DFinder has strongly improved and is now very slightly better than the one of the 2DFinder. It is important to note again that the neural network has only been trained with 2DFinder input, which makes the efficiency slightly biased in favor of using 2DFinder tracks as input. The efficiencies for the single track events are listed in Tab. 8.4. However, it is important to mention that the total track counts of the 3DFinder (19,802) are slightly smaller than the track count of the 2DFinder (20,227). The better efficiency is therefore a result of the better z-resolution when using 3DFinder input. The difference in total track counts may be explained by the high

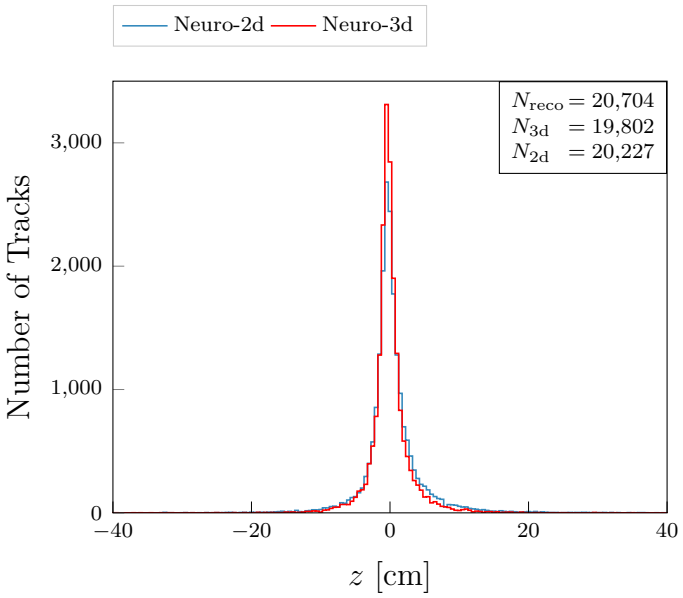


Figure 8.14: The z-distribution of the new hit-to-cluster association algorithm for all single track events of the last 50 runs of experiment 26. Note that all single tracks originate from the IP.

Table 8.4: Comparison of the different single track event efficiencies of the last 50 runs of experiment 26 using the new hit-to-cluster association. A found track is accepted if $z \in [-10, 10]$ cm.

Efficiency-3D	Efficiency-2D
94.1%	94.0%

background levels of the dataset. When a background peak is higher than the real peak, it is possible that the cluster deletion of the background cluster accidentally deletes the real cluster as well.

8.3 Cut on the ADC Count and Final Settings

8.3.1 The ADC Count

As earlier studies have suggested [25], a cut on the ADC count of every individual wire in the CDC can be very useful to reduce the number of background wire hits and therefore also the number of track segments dominated by background. In the CDC front-end electronics supplying the wires to the trigger system (see Sec. 4.4), the analog signal of the sense wire is converted to a digital signal called the ADC count. This integer is proportional to the deposited charge on this wire. In Fig. 8.15, the ADC counts of the wires on the track segments of a random run in experiment 26 for real IP tracks and fake tracks are plotted. Note that the plot is cut off at 300, as the ADC count can get very large. In subfigure (b), the fake wire hits are scaled such that the quantities of the fake and real tracks are the same. The large peak at approximately 50 is due to real minimum ionizing particles.

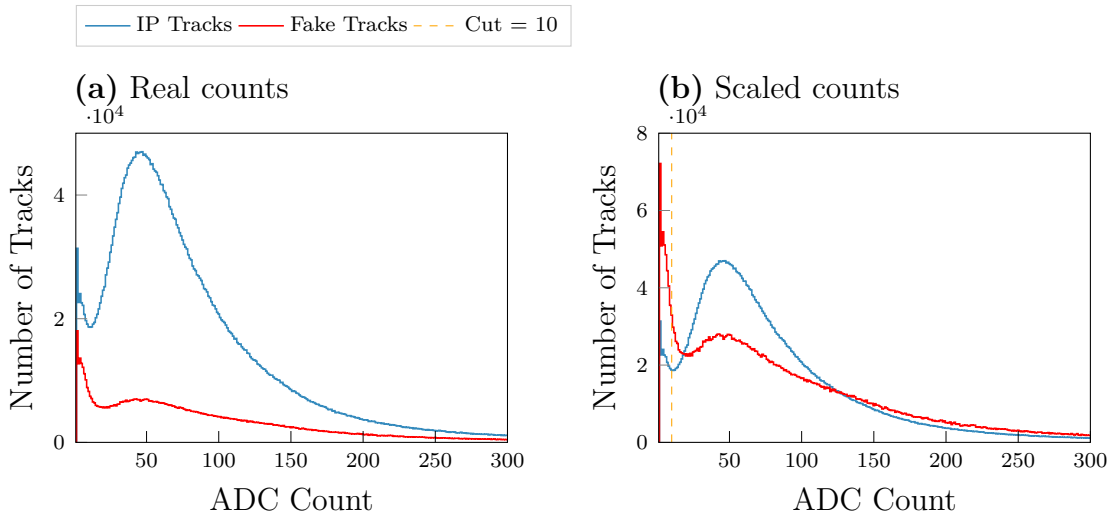


Figure 8.15: The ADC counts of the wires in all track segments in run 91 of experiment 26.

This peak is observed in the fake ADC count distribution as well, since fake tracks can also contain real track segment wires. When considering low ADC counts, the fake wires have a significant peak between 0 and 20. These small counts are likely produced by electronic crosstalk at the CDC end plates or from synchrotron photons. The small peak of the real IP tracks may be due to noisy wires in the real track segments that get used by the reconstruction, as they can be very close to the trajectory. This motivates a cut between 10 and 15 in order to reduce fake track segments. This `adccut` will be implemented at the hardware level of the CDC trigger, i.e., every sense wire with an ADC count of below `adccut` will not be considered by the TSF and will therefore not create track segments.

In Fig. 8.16, the effect of an `adccut`, implemented at the TSF module, is illustrated by a random single track Hough space. In subfigure (a), no `adccut` is active, while an `adccut` of 10 is used in subfigure (b). The number of fake track segments reduces significantly, while the track segments of the real track remain unchanged due to their high ADC count. As the implementation of such a cut is now possible at the L1 trigger, an `adccut` of 10 is also considered in the final efficiency analysis.

8.3.2 The Final Efficiencies and Settings

The final set of 3DFinder parameters and their corresponding values are listed in Tab. 8.5. Note that the new "butterfly-shape" cluster deletion is used where the `trim` parameters are defined in Sec. 7.4. The only parameter that is still kept from the original 3DFinder is `thresh`, requiring every cluster cell to have at least 85% of the peak weight to be included in the center of gravity calculation. The weight cuts are chosen according to the average weight contributions in the real clusters (see Fig. 8.5). The new super layer cuts allow for one missing axial layer and one missing stereo layer. This is a strict cut appropriate for the extremely high background observed in experiment 26 (see Fig. 8.10), which may be relaxed in the future after a cut on the ADC count is introduced at the L1 trigger hardware. As for the standard neural network trigger, four FPGA boards, one for each CDC quadrant, are

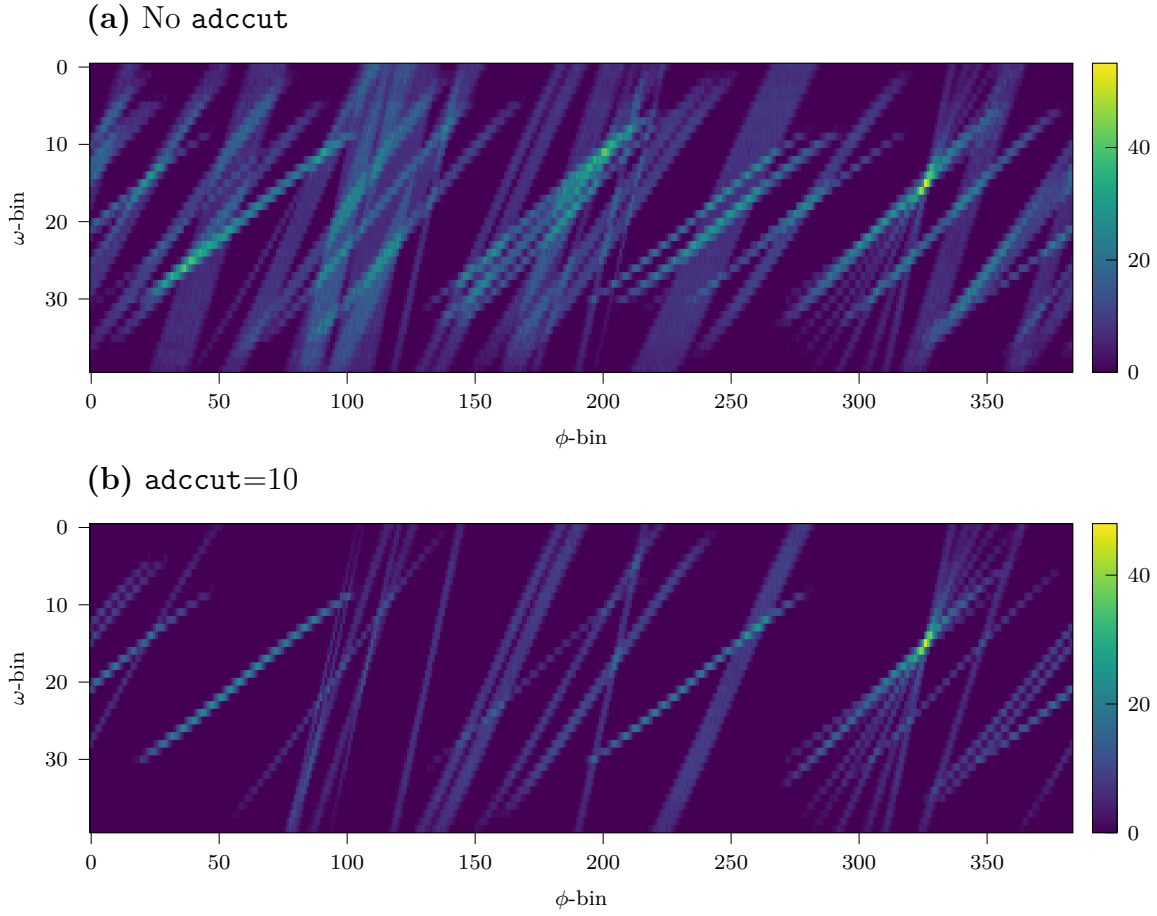


Figure 8.16: The Hough space of event 263 in experiment 26, run 1832, HLT 1, and f00005. No `adccut` with an `adccut` of 10 are compared.

Table 8.5: Final 3DFinder parameters.

Parameter	Datatype	Value
`dbscanning`	`bool`	`False`
`iterations`	`int`	5
`minpeakweight`	`int`	32
`mintotalweight`	`int`	450
`omegatrim`	`int`	5
`phitrim`	`int`	4
`thetatrim`	`int`	4
`minsuper_axial`	`int`	4
`minsuper_stereo`	`int`	3
`thresh`	`float`	0.85

available for the new Neuro Trigger with 3DFinder input. The `iterations` parameter has to be adjusted as only one quarter of the Hough space is considered for the track candidate search. The parameter `iterations` is limited by the latency requirement of the L1 trigger. It is important that the `minpeakweight`, `mintotalweight`, and both `minsuper` cuts can be updated in between future runs in order to react to the current background levels. Thus, the 3DFinder can either be made more efficient when low background is observed[4] or more robust against fake tracks when the trigger rate gets too high by properly adjusting these parameters.

Using the parameters shown in Tab. 8.5, Tab. 8.6 lists the final efficiencies achieved by the neurotracks, both with and without an `adccut` of 10, based on input from 2DFinder and 3DFinder. Note that the newly trained 3HL neural network with a cut of $\pm 10\,\text{cm}$ is

Table 8.6: Comparison of the efficiencies, the fake rates, and the feed-down under different ADC cuts of the last 50 runs in experiment 26 using the final 3DFinder settings.

Algorithm	`adccut`	**Efficiency**	**Fake Rate**	**Feed-Down**	$z \in [-10, 10]\,\text{cm}$
2DFinder	-1	94.0%	31.6%	9.8%	76.6%
3DFinder	-1	94.1%	13.1%	11.3%	86.7%
2DFinder	10	95.3%	13.5%	7.6%	77.3%
3DFinder	10	96.3%	5.8%	9.2%	86.4%

used again, which was trained with 2DFinder track candidates. The 3DFinder is better at utilizing the `adccut`, since the efficiency increases to 96.3%, while the 2DFinder only achieves 95.3%. In Fig. 8.17, the single track event z-distribution is displayed again for this `adccut`. Note that the total track count of the 3DFinder (19,296) is slightly larger than when using the 2DFinder (19,237). Now, with the `adccut` of 10, background clusters are considerably more unlikely to have a peak larger than a real cluster in the event, preventing accidental deletion. Fewer reconstructed tracks are observed as well. Without an `adccut`, a total of 20,704 reconstructed tracks were observed, while now only 19,816 tracks remain. It is important to remind here again that at least 4 axial and at least 3 stereo super layers are required for single track events. With an `adccut` of 10, some fake track segments that previously allowed some reconstructed tracks to pass the super layer threshold are removed.

The efficiency was calculated using the single track events only, all of which originate from the IP, while the other percentages contain all tracks. The fake rate, defined as

$$\text{fake-rate} = \frac{\text{total-tracks} - \text{related-tracks}}{\text{total-tracks}}\,, \tag{8.2}$$

where "total-tracks" is the total amount of found tracks by the corresponding finder and "related-tracks" all neural tracks that were successfully related to a reconstructed track, is listed as well. Here, the 2DFinder exhibits a fake rate more than double that of the 3DFinder. Only the feed-down, defined as

$$\text{feed-down} = 1 - \frac{(\text{track} \in [-10, 10]\,\text{cm}) \wedge (\text{related-reco} \in [-10, 10]\,\text{cm})}{\text{track} \in [-10, 10]\,\text{cm}}\,, \tag{8.3}$$

[4]This could, for example, be the case when an `adccut` is applied for the L1 trigger.

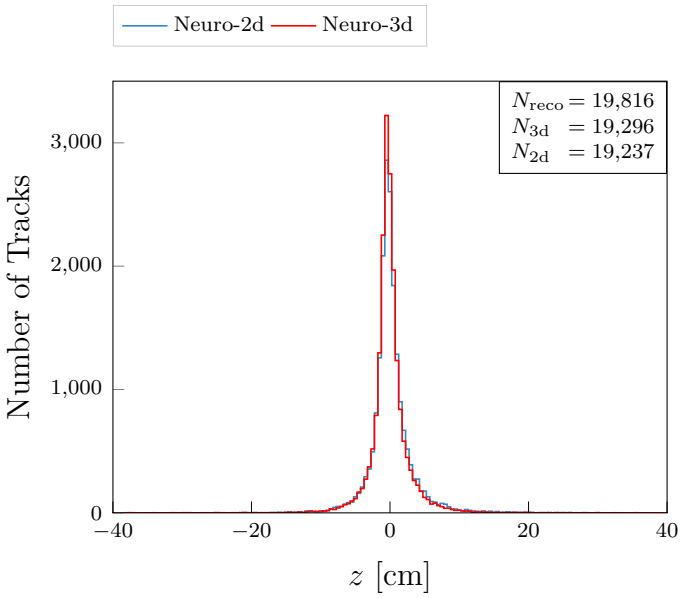


Figure 8.17: The z-distribution of the new hit-to-cluster association algorithm for all single track events of the last 50 runs of experiment 26. Here an `adccut` of 10 is applied.

is slightly higher with 3DFinder input. However, the neural network has only been trained on 2DFinder track candidates. With specialized training using 3DFinder input, the feed-down should diminish. In the last column, all neurotracks that are within $[-10, 10]$ cm are counted. As anticipated, the 3DFinder rejects more tracks from the outside, resulting in a higher relative density in the trigger region.

In Fig. 8.18, the z-distributions of the last 50 runs in experiment 26 with and without an `adccut` are displayed, comparing neuro-3d with neuro-2d predictions. These are the tracks used in Tab. 8.6 for the fake rate and feed-down percentages. The blue distribution depicts all found tracks, the green distribution includes all tracks that have been related to a reconstructed track, and the red distribution includes only the IP tracks. Note that the difference between related and all tracks is considerably smaller for the 3DFinder, i.e., fewer fake tracks are being found. The introduction of the `adccut` of 10 only reduces the number of IP tracks for the 3DFinder by 9423 tracks. These may even be tracks that were dependent on background track segments in the first place. Since `iterations` was set to 5, only a few tracks could be found per event, especially when a lot of high background peaks in the Hough space prevented the detection of more tracks. This is why the absolute number of tracks found by the 3DFinder is lower than that of the 2DFinder. For the trigger, this is not important as only a single track is necessary for a positive trigger decision (see Sub. 4.6.1). With an `adccut` of 10, the difference diminishes. As the L1 trigger does not have to find all tracks per event, this is not a problem. It should be noted as well that the resolution of the IP tracks is better for the 3DFinder, leading to a better rejection of background since the z-cut can be tightened, e.g., ± 10 cm or lower, in contrast to the present cut of 15 cm.

To investigate the suppression of non-IP tracks, the reconstructed tracks of these datasets that have been found by the 3DFinder and the 2DFinder are plotted in Fig. 8.19. In subfigure (a), no `adccut` is used. While the 3DFinder does indeed find fewer tracks than

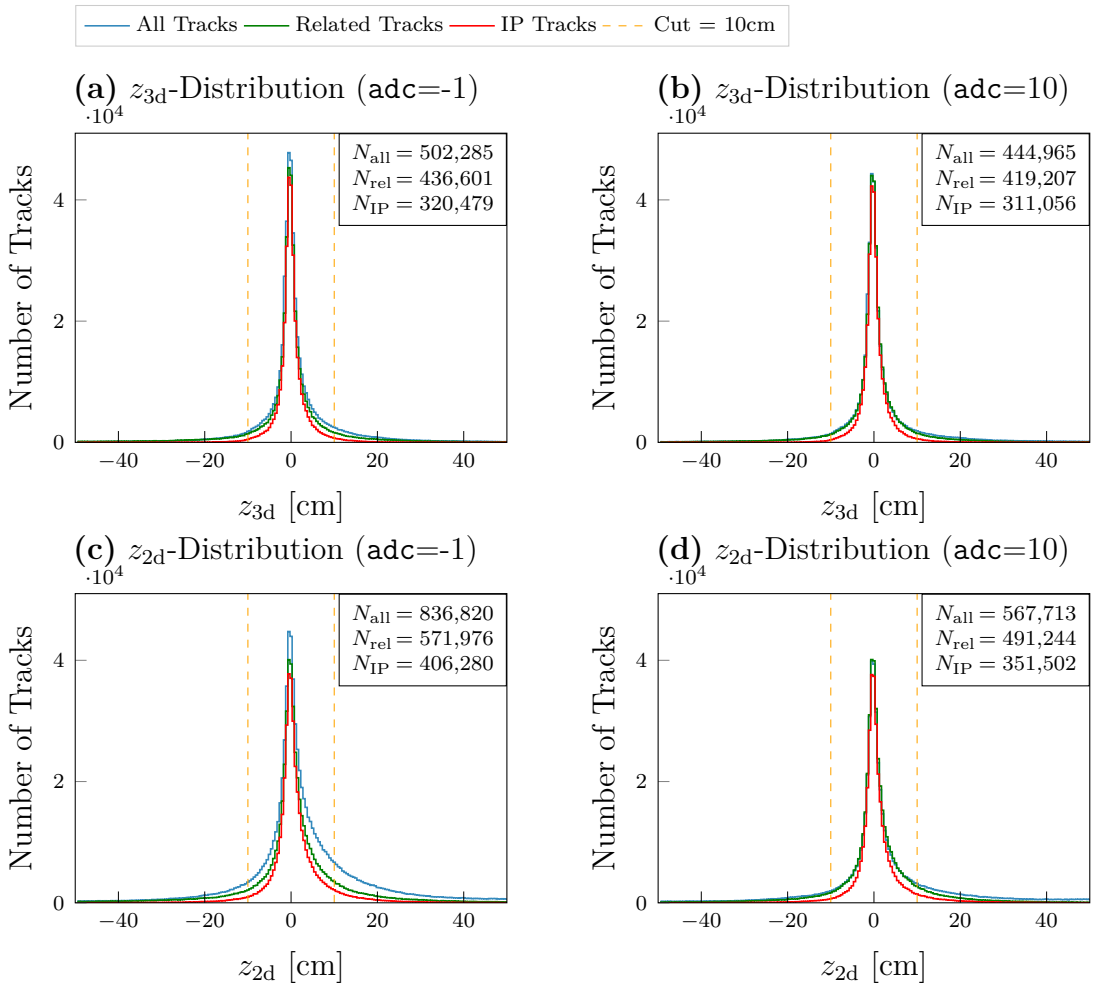


Figure 8.18: Comparison of the total, related, and vertex z-predictions under different ADC cuts of the last 50 runs in experiment 26.

the 2DFinder on the outside, the difference is not as large as expected from the Monte Carlo studies. The new fixed-volume clustering is apparently still efficient for displaced tracks. The same is observed with an `adccut` of 10 in subfigure (b). It is important to note that the data is biased, as the events were already accepted by the L1 trigger. The suppression of the 3DFinder has to be analyzed with unbiased data in the future.

As a last step, the total cluster weight and peak weight distributions are considered with an `adccut` of 10 in order to assess the cut positions. In Fig. 8.20, these distributions are compared with no `adccut`. Unsurprisingly, the average weight gets reduced when the `adccut` is present. Nevertheless, the `mintotalweight` and `minpeakweight` thresholds are still viable.

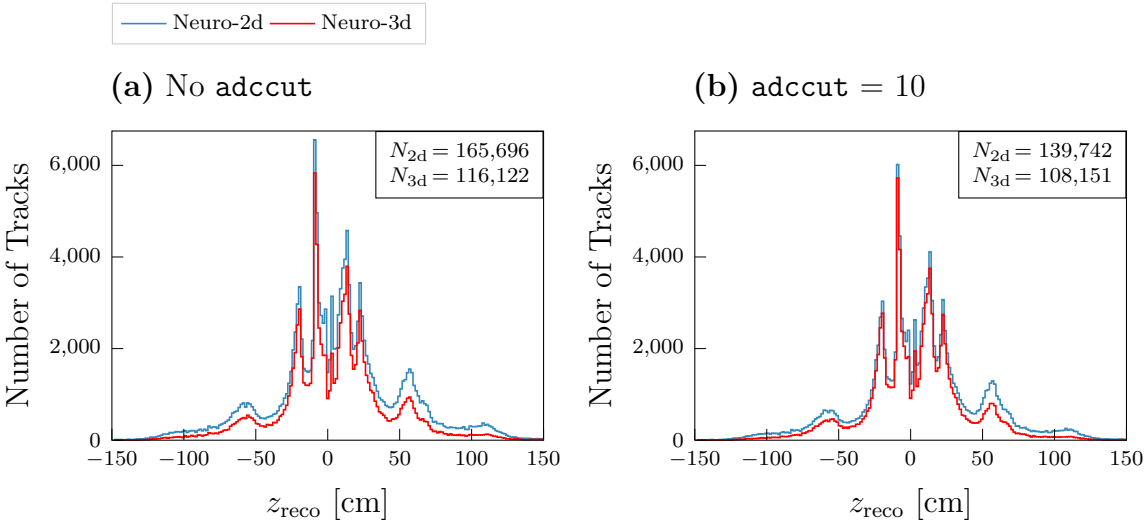


Figure 8.19: The z-distributions of the reconstructed tracks that the 3DFinder and the 2DFinder actually found in the last 50 runs of experiment 26. Note that the 3D-Finder is using the new hit-to-cluster association with the final settings. All tracks from $z_{\text{reco}} \in [-1, 1]\,\text{cm}$ are excluded for better visibility.

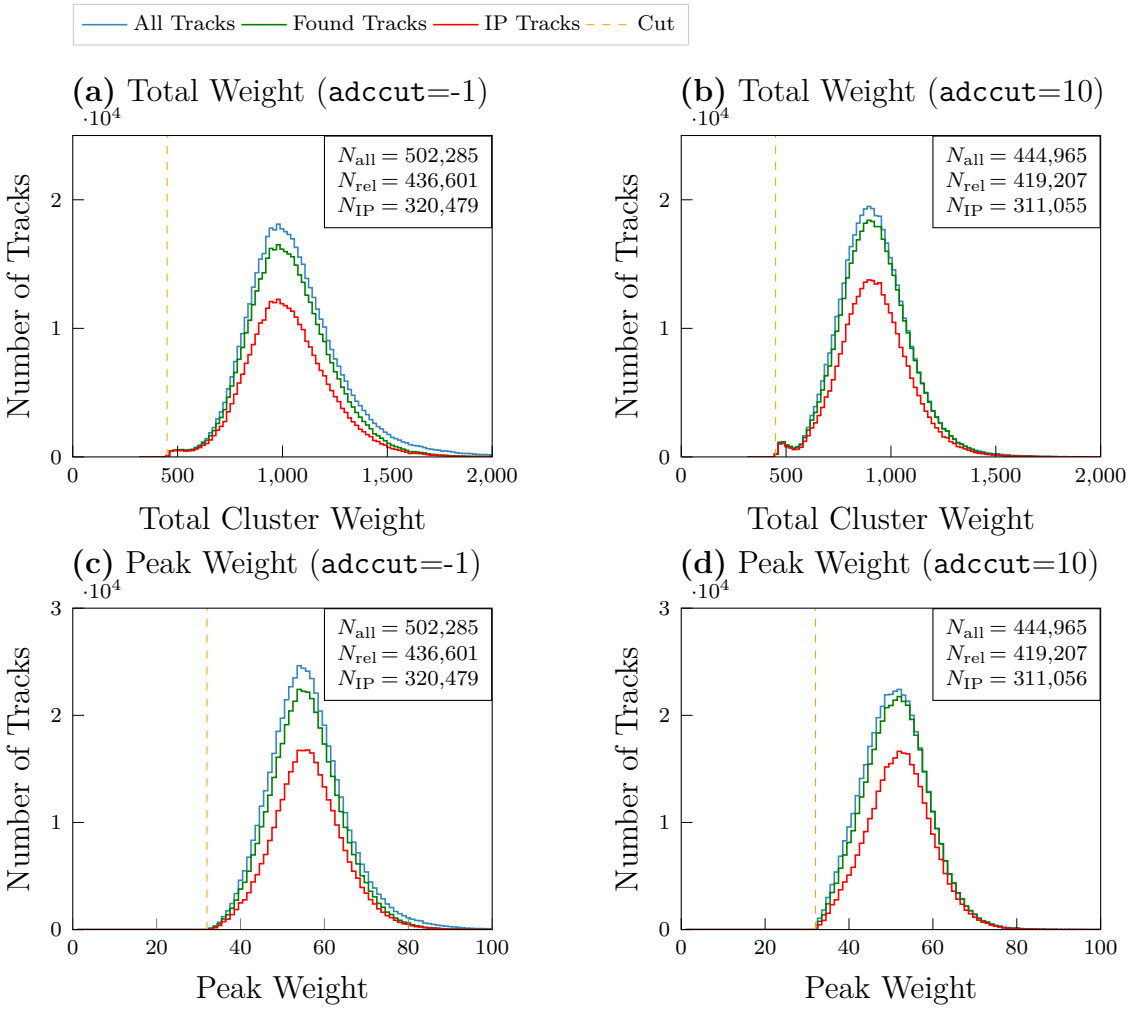


Figure 8.20: The total weight and peak weight distributions of the 3DFinder clusters with their corresponding cuts. Here the final implementation of the clustering algorithm is used.

Chapter 9

Conclusion and Outlook

9.1 Conclusion

Currently, 2DFinder tracks are used as track candidates for the L1 trigger at Belle II. As those candidates do not possess any z-information about the vertex origin, a lot of background tracks can be accidentally triggered. Furthermore, fake tracks are more likely to be produced as only 5 super layers are used for the track candidates. When upgrading to three-dimensional track-finding (3DFinder), this problem can be solved since a vertex assumption is made. The goal of this thesis was to make the 3DFinder algorithm operational.

In conclusion, the upgrade of the original 3DFinder was successful. The weaknesses of the original 3DFinder were identified in the studies and addressed. A new fixed-volume clustering algorithm was implemented, resulting in a better resolution of the neural network. The fixed cluster shape was determined by the average weight distribution of signal clusters. As this clustering algorithm is very fast and of deterministic length, its implementability on FPGA boards is ensured. With the weight cuts on the peak weight and the total cluster weight, some fake clusters can be rejected right at the creation. To reduce the high fake rates even further, new super layer cuts were implemented as a result of track segment studies. These cuts turned out to be by far the most effective way to reject background clusters. After studying the efficiencies on real data with very high background contributions, a new hit-to-cluster association was implemented. Now, every cluster gets the track segments assigned independently. This new algorithm considers the individual weight contributions of the track segment and assigns at most one track segment per super layer to the track. Hence, the 3DFinder has a fixed output of not more than 9 track segments in accordance with the neural network architecture. The cluster deletion shape ("butterfly-shape") was adjusted to delete the cluster in a more realistic way.

When considering single IP tracks from the real data, the efficiency of the neural network with 3DFinder input is comparable with the efficiency of using 2DFinder input. The resolution is always better with 3DFinder input, despite only using neural networks that have been trained with 2DFinder input in this thesis. The fake rate of the 3DFinder is below 15% and half that of the 2DFinder. After introducing a cut on the ADC count of the sense wires, an efficiency of 96.3% was determined.

9.2 Outlook

Some improvements to the 3DFinder are still possible, however. One could conduct a cluster statistic using real data in order to determine a new average cluster shape for the

fixed-volume clustering algorithm. However, this may be difficult due to the large background contributions. Adjustment of the 3DFinder parameters according to the observed background conditions in future runs is still necessary. Here, the trade-off between high efficiency and a low fake rate has to be considered. Furthermore, it may be useful to train new hit representations with the new clustering algorithm in mind. This may improve the suppression of displaced tracks.

Especially the neural network architecture and the available datasets for training are important to improve the z-distribution and reduce feed-down. For this purpose, extended inputs, multiple hidden layers, and 3DFinder input training could be used [21]. Unbiased training and analysis data are very important for future improvements. Here, a so-called "f"-stream could be implemented at the trigger to store some unbiased events that are not required to be accepted by the L1 trigger. With the high background rates in this data, the feed-down and rejection of the neural network should improve significantly.

The θ-prediction of the 3DFinder may be used as a new input parameter for the neural network, or expert networks could be trained for different θ-ranges. For non-IP tracks, the θ-prediction is either overestimating or underestimating the real polar emission angle since the 3DFinder track is fixed to the IP. This may be useful to differentiate between IP and background tracks. As suggested in Sec. 8.3, an implementation of the ADC cut into the CDC hardware seems to be very important for the L1 trigger when being faced with high background conditions in future experiments.

In the more distant future, the "traditional" Hough method for track-finding could be replaced by modern network architectures like Graph Neural Networks (GNN) [26]. Instead of using track segments, i.e., only the priority wire, the full information, including the drift time and ADC count, of the entire set of sense wires in the CDC could be used. For this, however, new and more powerful FPGA boards ("UT5") are necessary. Considering the expected backgrounds at the final design luminosity in several years, such methods may be the only way to guarantee an efficient track trigger for Belle II.

Appendix A

Appendix

A.1 Code of the New Clustering Algorithm

In this section, the C++ source code for the new fixed-volume clustering algorithm and hit-to-cluster association are listed. A detailed explanation of the clustering algorithms is given in Sec. 7.1 and of the hit-to-cluster association in Sub. 8.2.2.

Apart from some standard libraries like `vector` and `set`, a few custom types are used for the Hough space [4]. These are listed in List. A.1.

```
#include "boost/multi_array.hpp"
typedef unsigned short c3elem;
typedef boost::multi_array<c3elem, 3> c3array;
typedef c3array::index c3index;
typedef std::vector<c3index> cell_index;
```

Listing A.1: Definition of the custom data types.

In the following, the functions used for the new fixed-volume clustering algorithm are listed. Here, `m_houghVals` is a global, three-dimensional matrix representing the Hough space of the current event. Note that all configuration parameters of the 3DFinder are saved in the struct `m_params`. The class `SimpleCluster` stores information about the member cells and the associated track segments of a cluster. For example, methods like `append` can be used to add a Hough cell to this cluster data type, while `add_hit` can be used to add a track segment hit to the cluster. Note that these data types and methods were developed by Sebastian Skambraks in [4].

In List. A.2, the main function of the fixed-volume clustering algorithm is displayed.

```
std::vector<SimpleCluster> Clusterizend::makeClusters()
{
  std::vector<SimpleCluster> candidates;
  for (unsigned long iter = 0; iter < m_params.iterations; iter++) {
    auto [globalmax, peakweight] = getGlobalMax();
    if (peakweight < m_params.minpeakweight || peakweight == 0) {
      break;
    }
    auto [new_cluster, totalweight] = createCluster(globalmax);
    if (totalweight >= m_params.mintotalweight) {
      candidates.push_back(new_cluster);
    }
    deleteMax(globalmax);
  }
  return candidates;
}
```

Listing A.2: The main function of the clustering algorithm.

This function iterates over the `iterations` parameter, yielding a vector of clusters called `candidates`. It applies the weight cuts and calls the other clustering functions.

The first function that is called is `getGlobalMax`. In List. A.3, a simple iteration over the complete Hough space determines the global weight and its position.

```
std::pair<cell_index, unsigned long> Clusterizend::getGlobalMax()
{
  unsigned long maxValue = 0;
  cell_index max_index = {0, 0, 0};
  for (c3index iom = 0; iom < 40; iom++) {
    for (c3index iph = 0; iph < 384; iph++) {
      for (c3index ith = 0; ith < 9; ith++) {
        if ((*m_houghVals)[iom][iph][ith] > maxValue) {
          maxValue = (*m_houghVals)[iom][iph][ith];
          max_index = {iom, iph, ith};
        }
      }
    }
  }
  return {max_index, maxValue};
}
```

Listing A.3: The global maximum search.

In List. A.4, the fixed cluster shape is put around the maximum index determined by the `getGlobalMax` function.

```
std::pair<SimpleCluster, unsigned long> Clusterizend::createCluster(
    cell_index max_index)
{
  SimpleCluster fixedCluster;
  c3index omIndex = max_index[0];
  c3index phIndex = max_index[1];
  c3index thIndex = max_index[2];
  unsigned long totalClusterWeight = 0;

  for (c3index ith = std::max<int>(0, thIndex - 1); ith < std::min<int
      >(9, thIndex + 2); ith++) {
    for (c3index iph = phIndex - 1; iph < phIndex + 2; iph++) {
      c3index iphMod = (iph + 384) % 384;
      cell_index newMemberIndex = {omIndex, iphMod, ith};
      fixedCluster.append(newMemberIndex);
      totalClusterWeight += (*m_houghVals)[omIndex][iphMod][ith];
    }
  }
  if (omIndex - 1 >= 0) {
    for (c3index ith = std::max<int>(0, thIndex - 1); ith < std::min<int
        >(9, thIndex + 2); ith++) {
      for (c3index iph = phIndex + 1; iph < phIndex + 4; iph++) {
        c3index iphMod = (iph + 384) % 384;
        cell_index newMemberIndex = {omIndex - 1, iphMod, ith};
```

```
22          fixedCluster.append(newMemberIndex);
23          totalClusterWeight += (*m_houghVals)[omIndex - 1][iphMod][ith];
24        }
25      }
26    }
27    if (omIndex + 1 < 40) {
28      for (c3index ith = std::max<int>(0, thIndex - 1); ith < std::min<int
          >(9, thIndex + 2); ith++) {
29        for (c3index iph = phIndex - 3; iph < phIndex; iph++) {
30          c3index iphMod = (iph + 384) % 384;
31          cell_index newMemberIndex = {omIndex + 1, iphMod, ith};
32          fixedCluster.append(newMemberIndex);
33          totalClusterWeight += (*m_houghVals)[omIndex + 1][iphMod][ith];
34        }
35      }
36    }
37    return {fixedCluster, totalClusterWeight};
38  }
```

Listing A.4: Cluster creation around the maximum.

In this function, modulo operators are used in order to respect the boundaries of the Hough space.

The last function that is called in `makeClusters` is listed in List. A.5.

```
1  void Clusterizend::deleteMax(cell_index max_index)
2  {
3    c3index omIndex = max_index[0];
4    c3index phIndex = max_index[1];
5    c3index thIndex = max_index[2];
6    for (c3index ith = std::max<int>(0, thIndex - m_params.thetatrim); ith
       < std::min<int>(9, thIndex + m_params.thetatrim + 1); ith++) {
7      for (c3index iom = std::max<int>(0, omIndex - m_params.omegatrim);
         iom < std::min<int>(40, omIndex + m_params.omegatrim + 1); iom++)
          {
8        c3index phiIndex = phIndex + omIndex - iom;
9        c3index relativePhi = phiIndex - phIndex;
10       if (relativePhi > 0) {
11         for (c3index iph = phiIndex - m_params.phitrim; iph < phiIndex +
             m_params.phitrim + std::floor(2.4*relativePhi); iph++) {
12           c3index iphMod = (iph + 384) % 384;
13           (*m_houghVals)[iom][iphMod][ith] = 0;
14         }
15       } else if (relativePhi < 0) {
16         for (c3index iph = phiIndex - m_params.phitrim + std::ceil(2.4*
             relativePhi); iph < phiIndex + m_params.phitrim + 1; iph++) {
17           c3index iphMod = (iph + 384) % 384;
18           (*m_houghVals)[iom][iphMod][ith] = 0;
19         }
20       } else {
21         for (c3index iph = phiIndex - m_params.phitrim; iph < phiIndex +
             m_params.phitrim + 1; iph++) {
22           c3index iphMod = (iph + 384) % 384;
23           (*m_houghVals)[iom][iphMod][ith] = 0;
24         }
```

```
      }
    }
  }
}
```

Listing A.5: The deletion of the cluster around the global maximum.

This function deletes the "butterfly" shape around the maximum index. As in the cluster function, the boundaries of the Hough space have to be respected.

Given the list of clusters from the `makeClusters` function, a `hitsVsClusters` matrix is created. This is the confusion matrix, where the weight contribution of each track segment for each cluster is listed. The `allHitsToClusters` function listed in List. A.6 is assigning to each cluster at most one track segment per super layer.

```
std::vector<SimpleCluster>
NDFinder::allHitsToClusters(std::vector<std::vector<unsigned short>>&
    hitsVsClusters, std::vector<SimpleCluster>& clusters)
{
  std::vector<SimpleCluster> useclusters;
  if (hitsVsClusters.size() > 0) {
    // Iteration over the number of clusters
    for (unsigned long iclus = 0; iclus < hitsVsClusters.size(); iclus++)
        {
      std::vector<std::vector<long>> super_layer_numbers;
      // Iteration over all track segment hits
      for (unsigned long ihit = 0; ihit < m_hitIds.size(); ihit++) {
        unsigned short contribution = hitsVsClusters[iclus][ihit];
        if (contribution > 0) {
          super_layer_numbers.push_back({static_cast<long>(ihit),
              contribution, m_hitSLIds[ihit], m_prioTime[ihit]});
        }
      }
      // Iteration over all super layers
      for (unsigned short sl = 0; sl < 9; sl++) {
        std::vector<std::vector<long>> one_super_layer_contributions;
        for (unsigned long n_ts = 0; n_ts < super_layer_numbers.size();
            n_ts++) {
          if (super_layer_numbers[n_ts][2] == sl) {
            one_super_layer_contributions.push_back({super_layer_numbers[
                n_ts][0], super_layer_numbers[n_ts][1],
                super_layer_numbers[n_ts][3]});
          }
        }
        // Continue if there are no hits in the current super layer
        if (one_super_layer_contributions.size() == 0) {
          continue;
        }
        // Sorting after the drift times
        struct sortingClass {
          bool operator()(std::vector<long> i, std::vector<long> j) {
              return (i[2] < j[2]);}
        } sortingTimes;
        sort(one_super_layer_contributions.begin(),
            one_super_layer_contributions.end(), sortingTimes);
        long max_hit = one_super_layer_contributions[0][0];
```

```
34          long max_contribution = one_super_layer_contributions[0][1];
35          // Iteration over all track segments in this super layer
36          for (size_t index = 0; index < one_super_layer_contributions.size
                ();  index++) {
37            // The maximum weight contribution gets identified
38            if (one_super_layer_contributions[index][1] > max_contribution)
                  {
39              max_contribution = one_super_layer_contributions[index][1];
40              max_hit = one_super_layer_contributions[index][0];
41            }
42          }
43          clusters[iclus].add_hit(max_hit, max_contribution, m_hitOrients[
                max_hit]);
44        }
45        SimpleCluster& clu = clusters[iclus];
46        // The hits of the current cluster get extracted
47        std::vector<unsigned short> cluster_hits = clu.get_hits();
48        std::vector<unsigned short> cluster_sl_numbers;
49        for (const auto& element : cluster_hits) {
50          cluster_sl_numbers.push_back(m_hitSLIds[element]);
51        }
52        // A cut on the super layer numbers is applied
53        std::set<unsigned short> unique_sl_numbers(cluster_sl_numbers.begin
              (), cluster_sl_numbers.end());
54        size_t n_sl = unique_sl_numbers.size();
55        unique_sl_numbers.insert({0, 2, 4, 6, 8});
56        size_t with_axial_sls = unique_sl_numbers.size();
57        size_t axial_number = 5 - (with_axial_sls - n_sl);
58        size_t stereo_number = n_sl - axial_number;
59        if (axial_number >= m_params.minsuper_axial && stereo_number >=
              m_params.minsuper_stereo) {
60          useclusters.push_back(clusters[iclus]);
61        }
62      }
63    }
64    return useclusters;
65  }
```

Listing A.6: The new hit-to-cluster association.

Note that the super layer cuts are applied in this function. All clusters that survive those cuts are returned and will create a 3DFinder track.

A.2 The Missing Super Layer 1

When considering the large Monte Carlo dataset of signal tracks from Sec. 7.5, the occurrences of each super layer per track were studied. In Fig. A.1, the super layer distribution of the 3DFinder tracks is compared with the reconstructed and neuro-2d tracks. The blue bars represent all found track segments, while the orange bars only include a single track segment per super layer. It is immediate to see that the first stereo layer is missing for the 3DFinder in nearly half of all tracks. The 3DFinder is using the fixed-volume clustering algorithm with `minsuper` = 5. When considering the reconstructed and neuro-2d tracks, this anomaly is not observed. Only the first two super layers are less frequent. However,

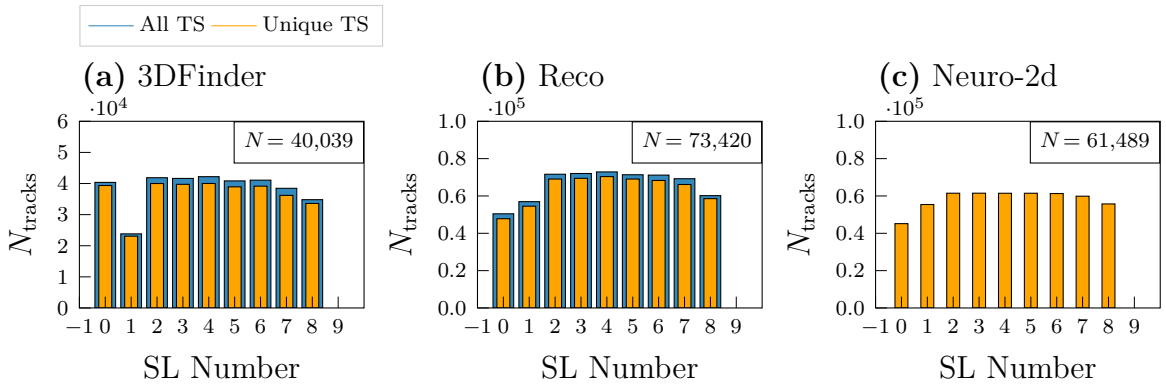


Figure A.1: The absolute track segment frequency for each super layer. Note that the big dataset without any background was used.

this is expected as tracks originating from the outside, which do not get automatically rejected by the reconstruction and the 2DFinder, are less likely to intersect the first two super layers. Note that the first super layer for the 3DFinder tracks is equally common as the other outer ones. This is dictated by the `minsuper` cut, which requires hits on 5 of the first 6 super layers. When super layer 1 is missing, super layer 0 must be present in a track.

In Fig. A.2 (a), the DBSCAN algorithm is used on the same dataset to determine whether this anomaly is caused by the new clustering algorithm. This is apparently not the case,

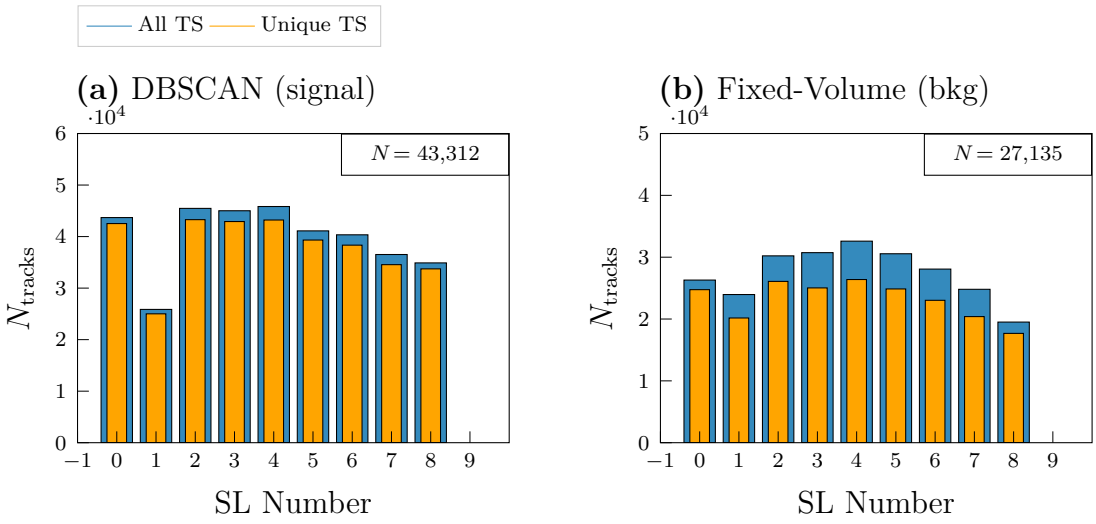


Figure A.2: How often a 3DFinder track contained a track segment from each super layer. In subfigure (a), the DBSCAN algorithm was used for signal tracks, while in subfigure (b), the fixed-volume clustering algorithm is used with nominal phase-3 background.

as super layer 1 is missing just as often. In subfigure (b), the super layer counts for the 3DFinder with nominal phase-3 background are displayed. In this context, super layer 1 is missing less often. This confirms that background track segments can fill up empty spots.

As the 3DFinder has to be efficient on IP tracks only, the tracks originating from $z \in [-10, 10]$ are considered in Fig. A.3. It is interesting that super layer 1 is now present as

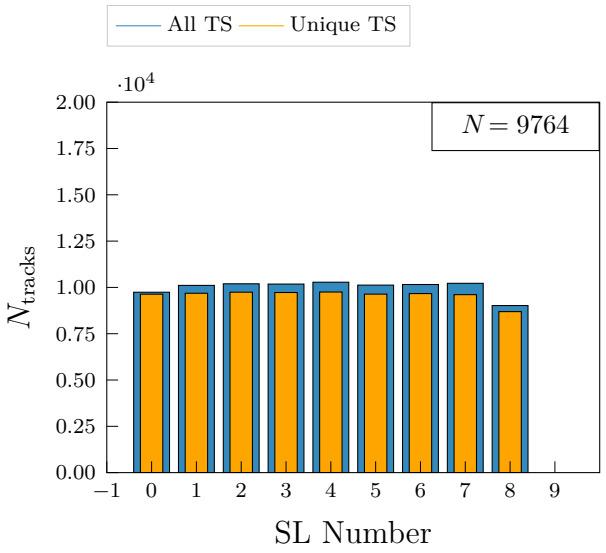


Figure A.3: The absolute frequency of each super layer for track originating from $z \in [-10, 10]$.

frequently as all the other super layers. Hence, the missing track segment is a result of displaced tracks and therefore not a problem.

In Fig. A.4, the neuro-3d z-distribution for all tracks, the tracks that do contain super layer 1, and the tracks without super layer 1 are plotted. Nearly all tracks without super layer 1 are displaced, especially around $z \approx \pm 50\,\text{cm}$. The corresponding z-correlations are depicted in Fig. A.5 in order to assess the z-resolution. This confirms that those tracks do indeed originate from the outside.

In Fig. A.6, three exemplary clusters of tracks originating from $z \in [49, 51]\,\text{cm}$ are illustrated. The single track segment that does not intersect the cluster is from super layer 1. Hence, the clustering algorithm does not find this track. Since super layer 0 consists only of axial wires that do not provide any z-information, the track segments from this super layer are still present in the cluster. As super layer 1 is the first stereo super layer with the most precise z-vertex construction capabilities, such track segments are the first ones incompatible with the IP hypothesis of the 3DFinder.

In conclusion, the missing super layer 1 track segments are due to displaced tracks originating from $z \approx \pm 50\,\text{cm}$ that fail to meet the 3DFinder IP hypothesis as they are the innermost stereo track segments. Since a `minsuper` cut of 5 was used, all other track segments had to be present, resulting in the loss of only the super layer 1 track segment.

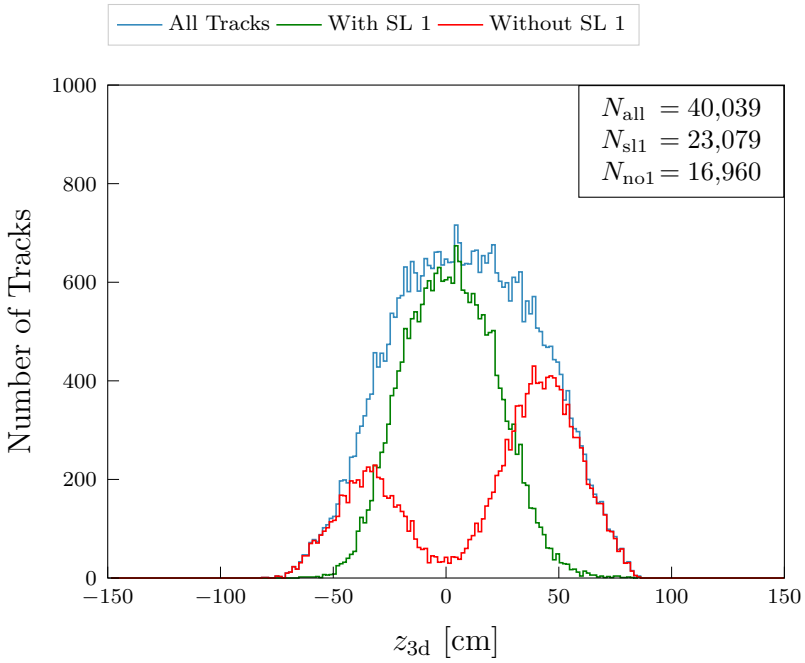


Figure A.4: The complete z-distributions of the reconstructed, neuro-1d and neuro-3d tracks without any background. Here, the z-distribution is split up into tracks that either have or don't have a track segment in super layer 1. Note that the new fixed-volume clustering algorithm with `minsuper` = 5 for the first six super layers.

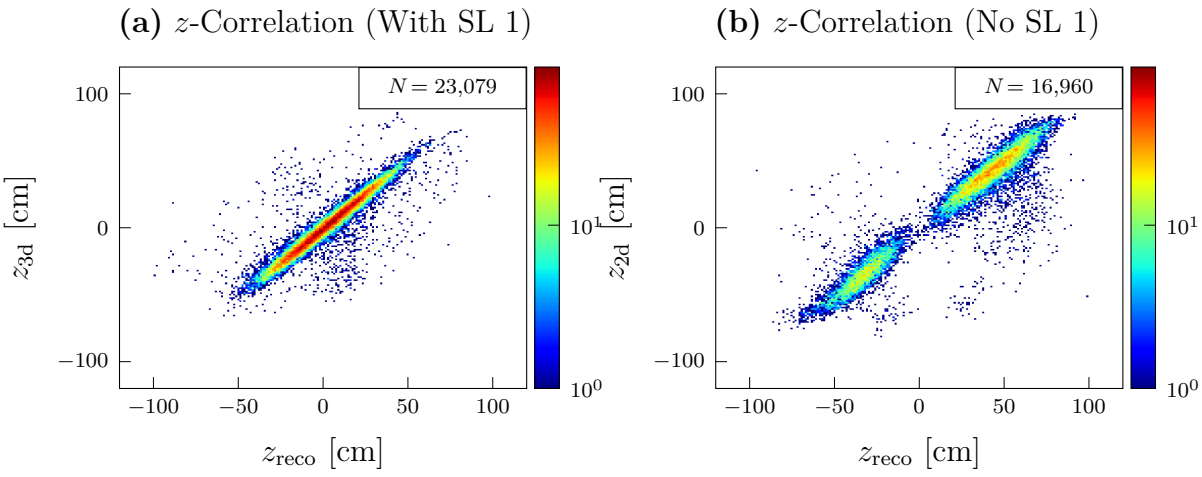


Figure A.5: Logarithmic heatmaps of the reconstructed z against the predicted z of the neural network with 3DFinder input of signal tracks only. In subfigure (a), only tracks that have a track segment in the first super layer are included, while in subfigure (b), no super layer 1 is present.

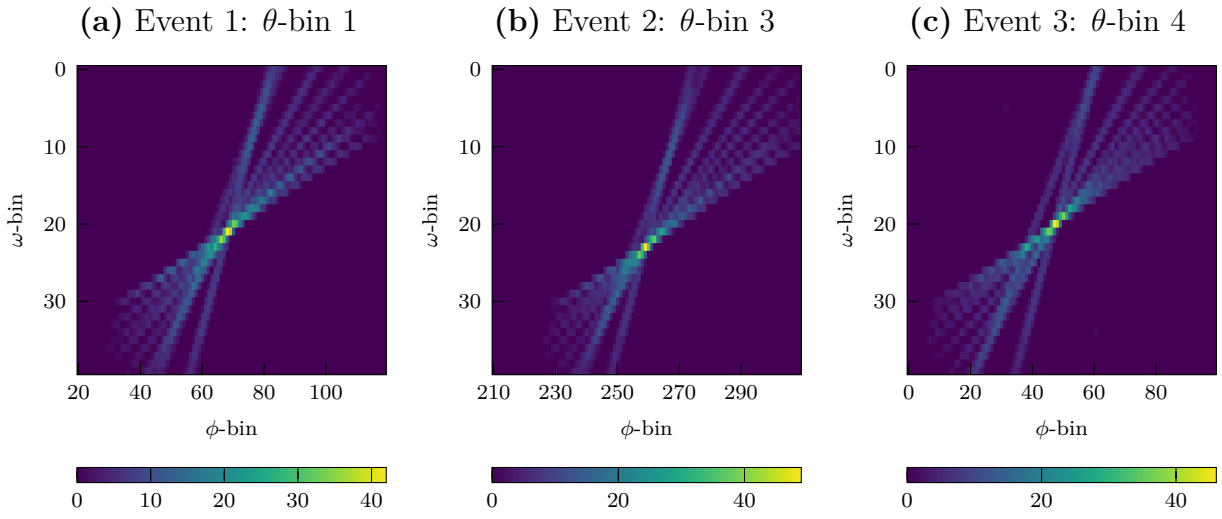


Figure A.6: Three exemplary signal particle gun clusters from $z \in [49, 51]\,\mathrm{cm}$. All three tracks have no association to track segments from super layer 1.

Acknowledgments

First and foremost, I would like to thank my supervisor, Prof. Christian Kiesling, for the opportunity to write this thesis. I'm very grateful for the time he dedicated to discussing new results and ideas with me. His enthusiasm for physics is one of a kind. I have learned a lot over the past year, and I am looking forward to working with him in the future.

I also want to thank Kai Unger for the important discussions about the design and feasibility of the new 3DFinder algorithm. I am especially thankful that Kai will implement my new clustering code on hardware.

Furthermore, I am very thankful for Dr. Sebastian Skambraks for joining our weekly trigger meeting. He helped me a lot with my understanding of the original 3DFinder algorithm and the corresponding source code.

I am very grateful to Felix Meggendorfer for always helping me with problems regarding basf2 and providing valuable ideas in the meetings.

I want to thank my friend Timo Forsthofer for the useful discussions over the past year, as we had very closely related thesis topics.

Last but not least, I want to thank Dr. Hans-Günther Moser, the head of the Belle II group at the Max Planck Institute for Physics, for helping me with the organizational matters.

Bibliography

[1] E Kou et al. "The Belle II Physics Book". In: *Progress of Theoretical and Experimental Physics* (2019). URL: https://arxiv.org/abs/1808.10567.

[2] S Neuhaus et al. "A neural network z-vertex trigger for Belle II". In: *Journal of Physics: Conference Series* 608 (2015), p. 012052. ISSN: 1742-6596. DOI: 10.1088/1742-6596/608/1/012052. URL: http://dx.doi.org/10.1088/1742-6596/608/1/012052.

[3] S. Bähr et al. "The Neural Network First-Level Hardware Track Trigger of the Belle II Experiment". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* (2024). arXiv: 2402.14962. URL: https://arxiv.org/abs/2402.14962.

[4] Sebastian Skambraks. "Efficient Physics Signal Selectors for the First Trigger Level of the Belle II Experiment based on Machine Learning". PhD thesis. Ludwig-Maximilians-University Munich, 2021. URL: https://edoc.ub.uni-muenchen.de/27627.

[5] T. Abe et al. *The Belle II Technical Design Report*. 2010. URL: https://arxiv.org/abs/1011.0352.

[6] Yukiyoshi Ohnishi et al. "Accelerator design at SuperKEKB". In: *Progress of Theoretical and Experimental Physics* (2013). URL: https://doi.org/10.1093/ptep/pts083.

[7] A. Paladino. "Beam background evaluation at SuperKEKB at Belle II". In: *JINST* (2020). URL: https://docs.belle2.org/record/1912/files/BELLE2-CONF-PROC-2020-008_submitted.pdf.

[8] Yoshihito Iwasaki et al. "Level 1 Trigger System for the Belle II Experiment". In: *IEEE Transactions on Nuclear Science* (2011). URL: https://www.phys.hawaii.edu/~idlab/taskAndSchedule/local_DAQ/main_update.pdf.

[9] Sara Pohl. "Track Reconstruction at the First Level Trigger of the Belle II Experiment". PhD thesis. Ludwig-Maximilians-University Munich, 2018. URL: https://edoc.ub.uni-muenchen.de/22085.

[10] A. J. Bevan, B. Golob, T. Mannel, et al. "The Physics of the B Factories". In: *The European Physical Journal C* 71 (2014). URL: https://doi.org/10.1140/epjc/s10052-014-3026-9.

[11] Andrei D Sakharov. "Violation of CP invariance, C asymmetry, and baryon asymmetry of the universe". In: *Soviet Physics Uspekhi* 34.5 (1991), p. 392. DOI: 10.1070/PU1991v034n05ABEH002497. URL: https://dx.doi.org/10.1070/PU1991v034n05ABEH002497.

[12] Jolanta Brodzicka et al. "Physics achievements from the Belle experiment". In: *Progress of Theoretical and Experimental Physics* 2012.1 (2012), p. 04D001. ISSN: 2050-3911. DOI: 10.1093/ptep/pts072. URL: https://doi.org/10.1093/ptep/pts072.

[13] Makoto Kobayashi and Toshihide Maskawa. "CP-Violation in the Renormalizable Theory of Weak Interaction". In: *Progress of Theoretical Physics* 49.2 (1973), pp. 652–657. URL: https://doi.org/10.1143/PTP.49.652.

[14] Kazunori Akai, Kazuro Furukawa, and Haruyo Koiso. "SuperKEKB collider". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spec-*

trometers, Detectors and Associated Equipment (2018). URL: http://dx.doi.org/10.1016/j.nima.2018.08.017.

[15] The Belle II Archive. *Belle II Outline*. URL: https://www.belle2.org/e21595/e21770/infoboxContent25432/BelleII-outline.pdf.

[16] Mariangela Varela. "Slow Pion Identification using the Pixel Detector of Belle II". MA thesis. Ludwig-Maximilians-University Munich, 2023. URL: https://docs.belle2.org/record/4101?ln=en.

[17] T. Kuhr, C. Pulvermacher, M. Ritter, et al. "The Belle II Core Software". In: *Computing and Software for Big Science* (2018). URL: https://doi.org/10.1007/s41781-018-0017-9.

[18] S. Lee et al. "Belle-II High Level Trigger at SuperKEKB". In: *Journal of Physics: Conference Series* (2012). URL: https://iopscience.iop.org/article/10.1088/1742-6596/396/1/012029.

[19] E. Won and K. T. Kim. "Development of track segment finder for central drift chamber based level-1 trigger system in the Belle II experiment". In: *Journal of the Korean Physical Society* (2021). URL: https://doi.org/10.1007/s40042-021-00143-w.

[20] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703. URL: https://arxiv.org/abs/1912.01703.

[21] Timo Forsthofer. "Improving the Belle II Neural Track Trigger with Deep Neural Networks". MA thesis. Ludwig-Maximilians-University Munich, 2024.

[22] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: (1996), pp. 226–231. URL: https://dl.acm.org/doi/10.5555/3001460.3001507.

[23] S. Agostinelli et al. "Geant4 - a simulation toolkit". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303. ISSN: 0168-9002. URL: https://www.sciencedirect.com/science/article/pii/S0168900203013688.

[24] A. Natochii et al. "Measured and projected beam backgrounds in the Belle II experiment at the SuperKEKB collider". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1055 (2023). URL: http://dx.doi.org/10.1016/j.nima.2023.168550.

[25] Yuxin Liu, Akimasa Ishikawa, and Taichiro Koga. "Study of Sudden Beam Losses of SuperKEKB and Development of 3D Track Hardware Trigger using Machine Learning at Belle II". MA thesis. Tsukuba: SOKENDAI, 2023. URL: https://docs.belle2.org/record/3902?ln=en.

[26] Marc Neu et al. "Real-Time Graph Building on FPGAs for Machine Learning Trigger Applications in Particle Physics". In: *Computing and Software for Big Science* 8.1 (2024). ISSN: 2510-2044. DOI: 10.1007/s41781-024-00117-0. URL: http://dx.doi.org/10.1007/s41781-024-00117-0.

Declaration:

I hereby declare that this thesis is my own work, and that I have not used any sources and aids other than those stated in the thesis.

München, den 12.04.2024

_____

Simon Hiesl