Dissertation

submitted to the

Combined Faculty of Mathematics, Engineering and Natural Sciences

of Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

M.Sc. Paul André Günther

born in Kassel

Oral examination: 05.07.2023

Track Reconstruction Development and Commissioning for LHCb's Run 3 Real-time Analysis Trigger

Referees: Prof. Dr. Stephanie Hansmann-Menzemer

Prof. Dr. André Schöning

Abstract

In Run 3 of the LHC, the LHCb experiment faces megahertz-rates of data containing beauty and charm hadron decays. Thus the task of the trigger is not to select any beauty and charm events but to select specific beauty and charm decays interesting for the LHCb physics programme. LHCb has therefore implemented a real-time data analysis strategy to trigger directly on fully-reconstructed events. This is done in two software-trigger stages, the first of which performs a partial event reconstruction on GPUs, while the second stage reconstructs the full event with offline quality on CPUs. This thesis describes LHCb's CPU-based track reconstruction for Run 3, highlighting the Forward tracking, which is the algorithm that reconstructs trajectories of charged particles traversing the entire tracking system. It is shown that using the capabilities of modern CPUs, the event throughput of the Forward tracking is improved by a factor of 3.5 while reaching reconstruction efficiencies of more than 95% for tracks above 5 GeV/c originating from a B meson.

Zusammenfassung

In Run 3 des LHC wird das LHCb-Experiment mit Datenraten von Beauty- und Charm-Hadronzerfällen im Megahertz-Bereich konfrontiert. Die Aufgabe des Triggers ist daher nicht jedes Kollisionsereignis, das solch einen Zerfall erzeugt hat, zu erkennen, sondern nur Beauty- und Charm-Hadronzerfälle zu selektieren, die interessant für LHCbs Forschungsprogramm sind. Um das zu erreichen hat LHCb eine Echtzeitdatenanalysestrategie entwickelt um direkt auf voll rekonstruierte Kollisionsereignisse triggern zu können. Das passiert in zwei Schritten; der erste rekonstruiert Teile des Kollisionsereignis auf GPUs, der zweite rekonstruiert das gesamte Ereignis in voller Qualität auf CPUs. In dieser Doktorarbeit wird LHCbs CPU-basierte Spurrekonstruktion für Run 3 beschrieben, wobei das Forward-Tracking hervorgehoben wird. Das Forward-Tracking rekonstruiert Teilchentrajektorien, die sich durch das gesamte Spurrekonstruktionssystem ziehen. Es wird gezeigt, dass der Ereignisdurchsatz des Forward-Trackings mit Hilfe moderner CPUs um einen Faktor $3,5$ verbessert werden kann, wobei eine Rekonstruktionseffizienz von über 95% für Spuren mit einem Impuls von mehr als 5 GeV/c, und die von einem B-Meson stammen, erreicht wird.

Contents

1 Introduction

Since the emergence of particle physics in the early 20th century, the science of making charged particle trajectories visible and thus measurable has driven the design of experiments that aim at revealing the underlying mechanics of nature. Fundamental discoveries like the existence of the electron [1], the positron [2] and the muon [3], all applied techniques that made it possible to follow a particle's path through the experimental setup. It is this path that forms a bridge between the quantum world and its macroscopic limit, from where we try to catch a glimpse of what is on the other side. Nowadays, it is not expected anymore to directly discover a new particle by the track it left in an apparatus; however, studying the properties and particularly the origins of particle trajectories has led and will still lead to a deeper understanding of the universe. The unbowed timeliness of track reconstruction becomes apparent in the structure of the large contemporary high-energy physics experiments at the Large Hadron Collider (LHC) and other facilities. They all employ various technologies to record the passage of charged particles precisely. While cloud chambers and nuclear emulsion plates showed trajectories visible and interpretable by the eye in the past, today's tracking systems are designed for high spatial track densities and ultra-fast data-taking rates to collect large amounts of statistics. The particle trajectories must be reconstructed from discrete space point measurements with the help of dedicated hardware and computers running specialised track reconstruction software.

The work presented in this dissertation has been conducted with the objective of developing fast and efficient track reconstruction software for the new tracking system of the upgraded LHCb experiment in Run 3 of the LHC and beyond. In its endeavour to precisely measure and compare the rates and topologies of heavy-flavour hadron decays, LHCb, described in Chapter 2, aspires to shine light on physics beyond the Standard Model of particle physics by studying CP symmetry violation [4], discover new particles either directly [5] or indirectly via their occurrence as virtual particles in loop corrections [6], and challenge predictions of the Standard Model such as lepton flavour universality [7]. All these studies have in common that they require vast data sets of reconstructed high-energy proton-proton collisions. Yet because it is impossible to determine beforehand whether a collision produced a process of interest, it is vital to reconstruct the event quickly, decide on the presence of interesting particle decays, and, if applicable, store the information or discard it to spare computing resources. The novel approach LHCb adopts for this, summarised in Chapter 3, is an entirely software-based two-stage trigger system performing a real-time analysis of the collision event. Part of this real-time trigger strategy is the full offline-quality event reconstruction in the

second trigger stage, with the reconstruction of tracks as a central component described in Chapter 4. The author's main work addresses the fast and efficient finding of tracks traversing LHCb's entire tracking system. The corresponding algorithm developed in the course of this dissertation is called the *Forward tracking*, detailed in Chapter 5, together with the Matching algorithm, a neural-network-based approach the author also contributed to. The track reconstruction algorithms were designed and tuned on simulation before the start of Run 3. The first track-reconstruction tests and studies on data were performed by the author during the detector commissioning in 2022 and are highlighted in Chapter 6.

The track reconstruction development in LHCb's real-time trigger software is strongly interconnected with the development of other software components not explicitly documented in this thesis. The author contributed in particular to LHCb's Kalman filter, the framework handling the decay selections, and the components monitoring the quality of the event reconstruction during data taking. The work on the selection framework was conducted along with preparations for the author's Run 3 data analysis, measuring the charm-production cross-section, also mentioned in Chapter 6. Moreover, the author joined the team maintaining the software stack which organises the code review and testing across all projects for the Run 3 software to ensure high code quality.

2 The LHCb Experiment

This chapter overviews the LHCb experiment (the 'b' stands for beauty) with its detector located at Interaction Point 8 of the LHC organised in the Conseil Européen pour la Recherche Nucléaire (CERN). Only the upgraded LHCb detector with its sub-detectors, data acquisition system and software framework as of 2022 during Run 3 of the LHC is discussed. Information about the previous LHCb setup can be found in Ref. [8].

2.1 Large Hadron Collider

The LHC is the world's largest and most powerful particle accelerator located roughly 100 m below ground near Geneva, Switzerland. It is a hadron-hadron collider composed of two superconducting magnet-rings with 26.7 km circumferences storing two counter-rotating particle beams. The particle beams collide in four interaction points around which the four large experiments, ATLAS, CMS, ALICE and LHCb, are built. The LHC was planned for a maximum centre-of-mass energy of $\sqrt{s} = 14\,\text{TeV}$ and instantaneous luminosities of $\mathcal{O}(10^{34}\,\text{cm}^{-2}\text{s}^{-1})$ for proton-proton (pp) collisions. The operation of the LHC is divided into Runs, the first of which took place from 2009 to 2013; Run 2 started in 2015 and ended in 2018, and Run 3 has begun in 2022 and is planned to continue until 2026. During Run 3, the LHC collides protons with a centre-of-mass energy of $\sqrt{s} = 13.6\,\text{TeV}$. While ATLAS and CMS operate at a peak instantaneous luminosity of $\mathcal{L} = 2 \times 10^{34}\,\text{cm}^{-2}\text{s}^{-1}$ decreasing with time during a fill, LHCb uses luminosity levelling [9] to keep the instantaneous luminosity lower but constant. This is done to ease physics analyses by delivering steady conditions, but also because the processes LHCb is designed to study occur at high rates in the forward region of hadron collisions, and it is a challenge to efficiently record them at high luminosity as described in Section 3.1. For Run 3, LHCb aims at an instantaneous luminosity of $\mathcal{L} = 2 \times 10^{33}\,\text{cm}^{-2}\text{s}^{-1}$ which yields about five pp collisions per bunch crossing on average. A bunch crossing is the passing of particle clouds through each other at the interaction points, as sketched in Figure 2.1. Nominally, this happens every 25 ns or with a frequency of 40 MHz. In practice, bunches are grouped in bunch trains separated by more than 25 ns, lowering the average collision frequency. Aside from colliding protons, the LHC delivers collisions of heavy ions, such as lead and xenon, as well as proton-ion collisions.

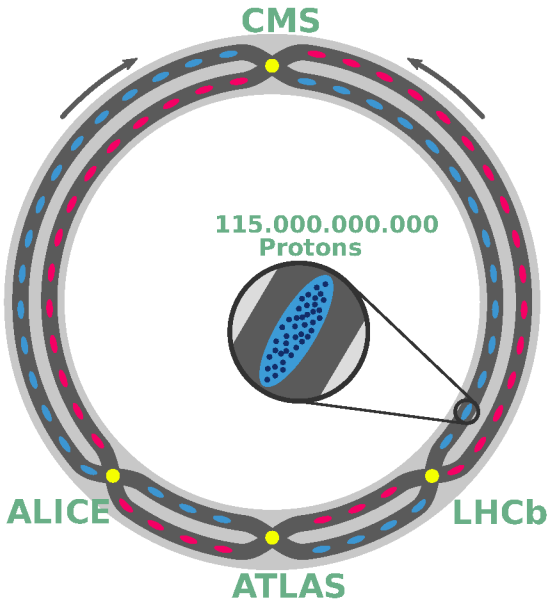


Figure 2.1: Sketch of the LHC with its accelerator ring containing two beam pipes and the four main interaction points with their experiments. The beam pipes contain the particle bunches that cross at the interaction points [10].

2.2 LHCb Detector Overview

The LHCb detector, shown in Figure 2.2, is a single-arm forward spectrometer covering angles from 10 mrad to 300 mrad relative to the beamline in the x-z plane, and up to 250 mrad in the y-z plane, corresponding to a pseudorapidity range of approximately $2 < \eta < 5$. The coordinate system has its origin at the nominal interaction point in the centre of the Vertex Locator (VELO) with the z axis pointing along the beam pipe into the detector (*downstream*), the y axis vertically oriented towards the surface, and the x axis pointing away from the centre of the accelerator ring (into the plane in Figure 2.2). The detector's single-arm forward design is unique among the experiments at the LHC and was chosen because pairs of b quarks are predominantly produced at small angles around the beamline in pp collisions [11]. The resulting hadrons containing heavy quarks, for example B and D mesons, are enormously boosted with respect to the laboratory frame and thus exhibit significant flight distances before they decay. Precisely reconstructing their displaced decay vertex in distinction to the primary pp collision vertices is a key feature of LHCb and is the purpose of the first sub-detector, the VELO. Stable particles, like protons and electrons, or particles with a longer lifetime, like pions, kaons, and muons, further travel through the first Ring-imaging Cherenkov detector (RICH1), the Upstream Tracker (UT), the magnet, the Scintillating Fibre (SciFi) tracker, the second RICH detector (RICH2), the electromagnetic calorimeter (ECAL), and the hadronic calorimeter (HCAL). Muons also reach the four muon stations (M2-M5) at the end of the detector. The RICH detectors, the calorimeters

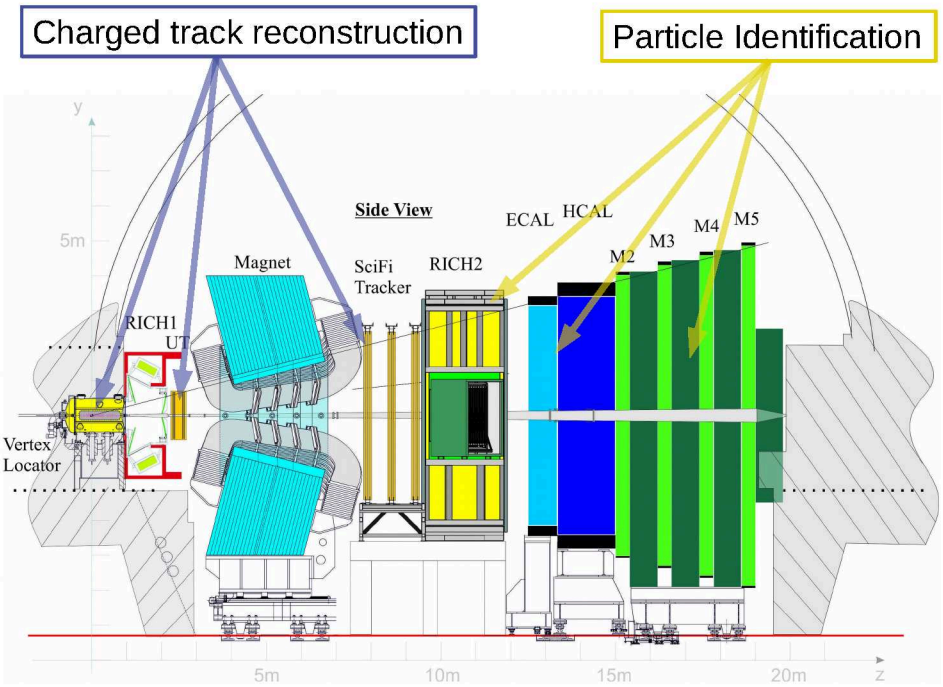


Figure 2.2: Layout of the LHCb spectrometer shown from the side. Particles collide in the Vertex Locator.

and the muon stations make up the particle identification (PID) system, which is only described briefly in Section 2.5 as it plays a minor role in the work presented in this thesis. The VELO, UT and SciFi tracker, together with the magnet, constitute LHCb's tracking system and are described in more detail in the following sections.

LHCb features new tracking detectors in Run 3 to record data at a five times higher instantaneous luminosity than during Run 2. All sub-detectors are upgraded with new electronics allowing for a trigger-less data acquisition system as outlined in Section 2.6.

2.3 Magnet

The spectrometer's magnet consists of two saddle-shaped aluminium coils in a window-frame steel yoke. The magnet poles are tilted towards the interaction point, following the acceptance of LHCb. It is a water-cooled warm magnet with a bending power of $\int B\mathrm{d}l \simeq 4\,\mathrm{T\,m}$ for tracks traversing the entire tracking system [12]. The strength of the individual field components as seen by a charged particle is shown in Figure 2.3. Particle trajectories are bent most by the B_y component, deflecting the particles in the x-z plane. While the B_x component is negligible, the effect of B_z must be accounted

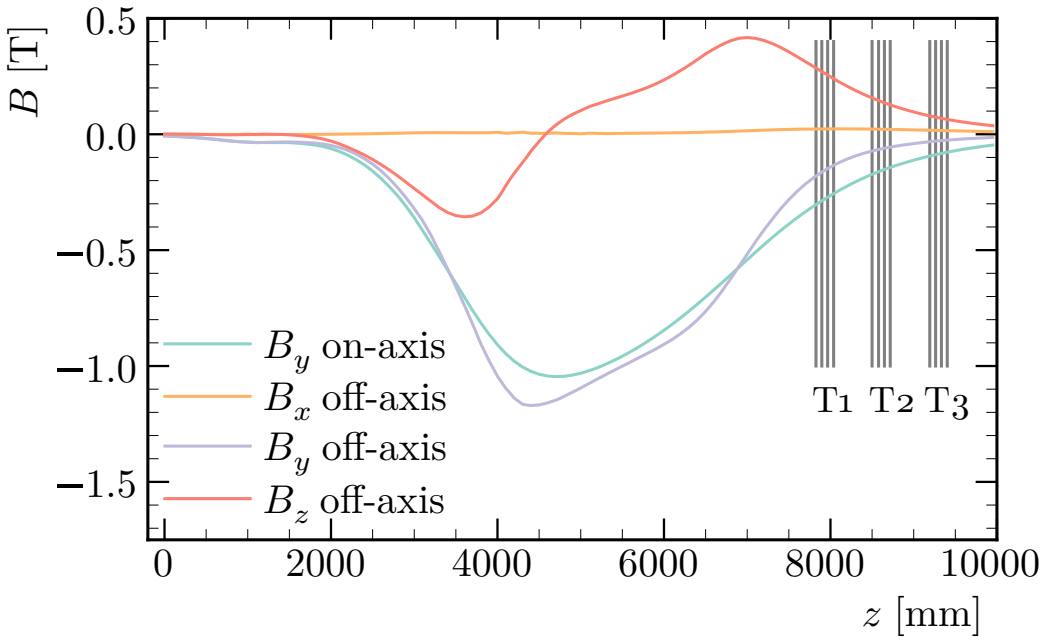


Figure 2.3: Magnetic field strength components in dependence on z position within LHCb's tracking system. The B_y component is shown "on-axis" for a particle travelling along the z axis. The components off-axis are shown for a particle with positive slopes in x and y. The vertical grey lines denoted T1, T2, and T3 represent the position of the three SciFi tracker stations. Adapted from Ref. [13].

for when accurately describing trajectories through the tracking system. The magnetic field is mapped by measuring its strength with 3D Hall probes in the relevant regions of the detector, creating a grid of field values [14]. Simulations of the magnetic field complement the grid in regions where measurements are not feasible. The sign of the current flowing through the coils is reversed every few weeks during data taking such that data sets of roughly equal size for both polarities are recorded. This reduces potential particle detection asymmetries which could otherwise affect measurements of fundamental asymmetries.

2.4 Tracking Detectors

The tracking system employs three different detector types to measure the trajectory of charged particles and to estimate their momentum from the bending caused by the magnet. Each of the three sub-detectors is new and was developed specifically for the conditions during Run 3 and beyond.

2.4.1 Vertex Locator

The VELO is the most vital sub-detector for the LHCb experiment. It detects particles created in or close to the beam collision region and is used to locate the primary pp collision vertices and displaced vertices from particle decays. Furthermore, it seeds the reconstruction of tracks in other sub-detectors. The closer the detector is to the interaction region, the better the vertex resolution. The VELO modules are therefore located 5.1 mm from the beam, only separated from it by the RF boxes located 3.5 mm from the beamline. The RF boxes, made of 150 µm thin aluminium foil, are corrugated enclosures providing the necessary barrier between the beam vacuum and the VELO vacuum and protect the sensors from electromagnetic induction caused by the beam. The VELO consists of 52 L-shaped modules, 26 on each side of the beamline, forming a movable half, as shown in Figure 2.4. The module positions are chosen such that particles within LHCb's acceptance traverse at least four modules. Until the LHC declares stable beams, the two halves are separated by about 50 mm, which is called *open* VELO. During stable beams, the VELO is moved into its *closed* position, depicted in Figure 2.4 on the right.

Each VELO module is composed of four sensors with three VeloPix [16] chips. The chips have an active matrix of 256 × 256 pixels with a size of 55 µm × 55 µm [15]. The entire VELO thus totals almost 41 million channels. Pixels are clustered into hits on the readout cards during data acquisition as explained in Section 2.6. The reached hit resolution is about 12.5 µm in the x and y coordinate [17], which allows the most precise track segment reconstruction of LHCb. However, the position close to the beam with the necessity for the RF foil has the downside of adding a significant material budget in the path of the particles, amounting to a radiation length fraction of $x/X_0 \simeq 21.3\%$,

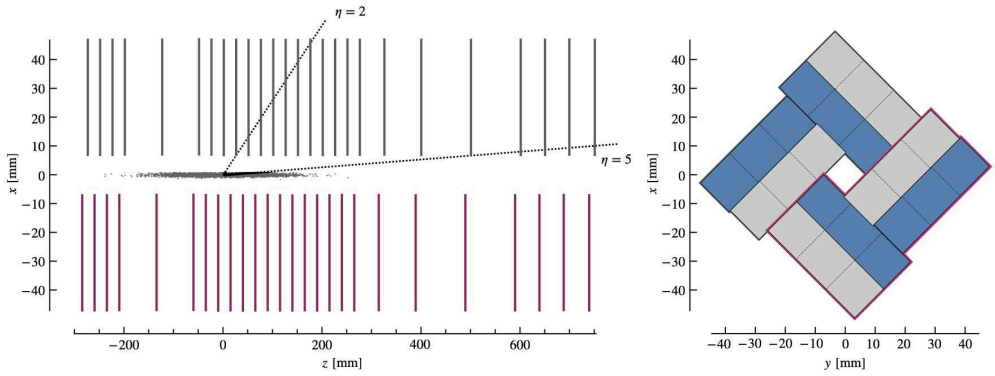


Figure 2.4: The left image shows the schematic top view of the VELO's x-z plane at $y = 0$ with an illustration of the luminous region and the nominal pseudorapidity acceptance in the two halves. On the right, two modules with the nominal layout of the sensors around the z axis in the closed configuration are shown with half of the ASICs facing upstream (grey) and the other half facing downstream (blue) [15].

half of which is attributed to the RF foil [18].

2.4.2 Upstream Tracker

The UT is the second tracking detector particles encounter. It is located around $z = 2485\,\text{mm}$ right in front of the magnet and thus already contains a significant magnetic field, which makes the UT interesting for fast track reconstruction approaches exploiting the momentum estimated from the combination of VELO and UT measurements (see Sections 3.2.1 and 5.3.2). The four layers of the UT are illustrated in Figure 2.5. The UT's silicon micro-strip sensors are shown as coloured boxes in the figure and sit on vertical staves. They vary in length and pitches, matching the expected occupancy, *i.e.* a decreasing particle flux away from the centre around the beam pipe hole. The green sensors reach a hit resolution of about $55\,\mu\text{m}$, while the sensors in the centre resolve hits at around $27\,\mu\text{m}$ [19]. The layers are arranged in a x-u-v-x layout, *i.e.* the first and the last layer only measure the x coordinate of a track, while the two layers in the middle measure a y component with their stereo angle of $\pm 5°$. Because the UT is positioned between the VELO and the SciFi tracker, where scattering of the particles can severely obstruct their reconstruction, it is essential to the track reconstruction that the material budget is kept low. This was considered in the detector design, and the average radiation length fraction in the pseudorapidity region $2 < \eta < 5$ is relatively

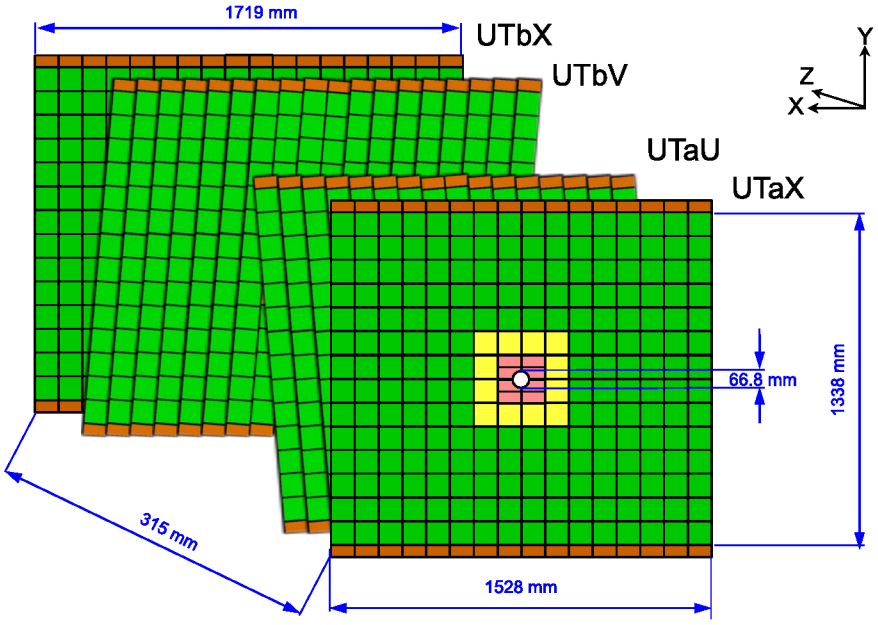


Figure 2.5: Sketch of the four UT silicon planes [19]. Different colours mark sensors with different pitches and lengths: pink sensors in the centre have a pitch of $93.5\,\mu\text{m}$ at a length of roughly $5\,\text{cm}$, yellow sensors have the same pitch, but twice the length and green sensors have a pitch of $187.5\,\mu\text{m}$ and are also twice as long the sensors in the centre.

small with $x/X_0 \simeq 7\%$ [15].

The UT was the last sub-detector to be installed, completed in March 2023 and therefore too late to include its measurements for the data studies performed in this thesis.

2.4.3 Scintillating Fibre Tracker

The SciFi tracker is the largest track reconstruction sub-detector in LHCb and the only one located downstream of the magnet. It thus complements trajectories starting in the VELO or UT and provides measurements to best estimate a particle's momentum. The plastic-scintillator fibres are arranged in multi-layer fibre mats with silicon photo-multipliers (SiPM) at their outer edge to detect the scintillation light. The mats have a length of about 2.4 m with a mirror glued to their end at $y = 0$ to reflect light back to the SiPMs. There are twelve detection layers distributed over three stations (T1, T2, T3), *i.e.* four layers each in an x-u-v-x configuration, as shown in Figure 2.6. The x layers have their fibres oriented vertically, and the stereo layers, u and v, are rotated by $\pm 5^\circ$ to be able to measure the y component of a track. To ensure the mechanical stability of the SciFi tracker, the mats are arranged in modules hinged into the support structure such that gravity pulls longitudinally on them. This introduces a tilt with respect to the y axis of around 0.2° in LHCb's global coordinate system used for track reconstruction.

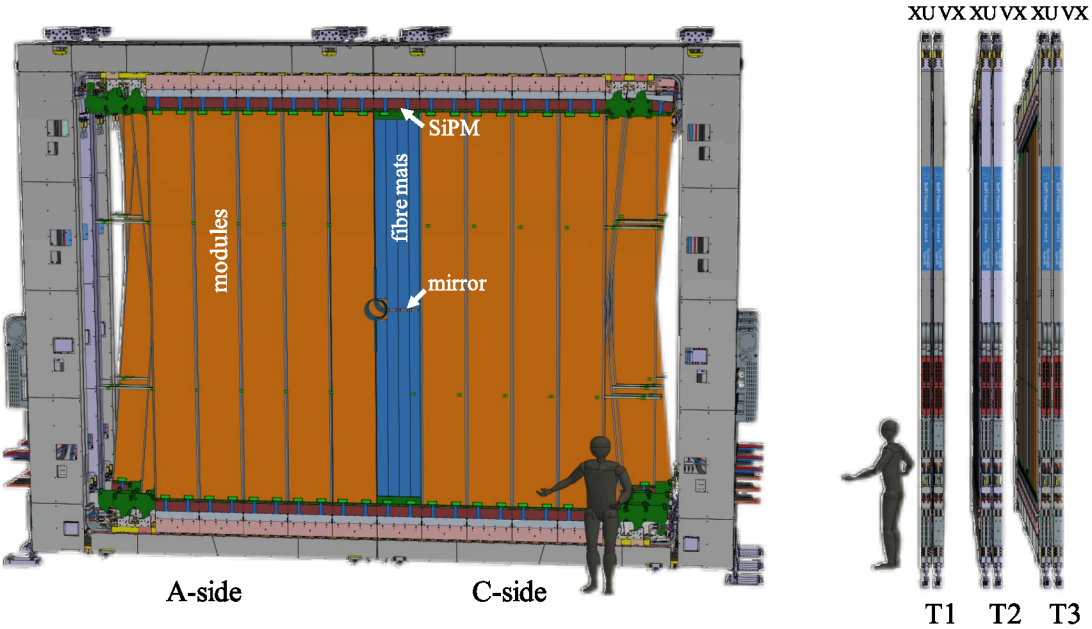


Figure 2.6: Front and side view of the SciFi tracker [15]. The acceptance ranges from around 20 mm from the edge of the beampipe to ± 3186 mm horizontally in the last station, and ± 2425 mm vertically in all stations.

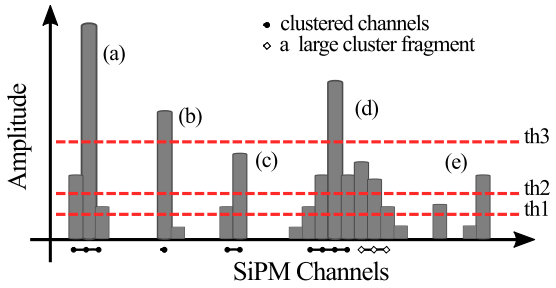


Figure 2.7: Illustration of different cluster types created by the SciFi FPGA clustering algorithm [15].

Gaps between fibre mats and modules lead to a total geometric inefficiency of approximately 1.7% per layer. Excluding these gaps, the SciFi detector's single-hit efficiency is estimated to be larger than 99% with a single-hit position resolution of (64 ± 16)µm for perpendicular tracks estimated during test-beam campaigns [15]. The fibre and readout-channel density and thus the granularity is the same across the entire sub-detector. A particle passing through all SciFi layers traverses material worth at least $x/X_0 = 12.4\%$ radiation length fraction.

The SciFi hits used for the particle tracking are created from clustered SiPM channels. The clustering is performed in the SciFi tracker's front-end electronics on a field-programmable gate array (FPGA) [19]. The FPGA algorithm groups neighbouring SiPM channels into clusters of different sizes according to three comparator thresholds, as shown in Figure 2.7. The cluster size eventually determines the uncertainty on the hit position used by the track reconstruction and currently ranges from 50 µm to 290 µm. The hit position is calculated from a weighted average of all participating channels in the cluster. The data bandwidth limits the number of clusters the FPGAs can send per event. This sets an upper bound of 45568 hits the SciFi tracker can provide for the track reconstruction per event. This upper bound is hardly ever reached in pp collisions[1], yet it poses a measurable restriction for ion-ion collisions.

2.5 Particle Identification

The information from the particle identification system is crucial to reduce the combinatorial background when searching for specific particle decays. It furthermore allows controlling the rates of particle misidentification in data analyses. Charged particle identification relies on tracks reconstructed using the system presented in the previous sections. Three technologies are employed: Ring-imaging Cherenkov detectors [20], calorimeters [20, 21], and multi-wire proportional chambers interleaved with iron absorbers for muon detection [20, 22].

[1]Under nominal Run 3 conditions the average number of SciFi hits is around 6000.

RICH Two RICH detectors are used in LHCb, providing charged particle discrimination between pions, kaons and protons in the $2.6 - 100\,\mathrm{GeV}/c$ momentum range. Both detectors are filled with fluorocarbon gaseous radiators but with different refractive indexes. The Cherenkov photons produced by a charged particle traversing the radiator are reflected by mirrors focusing the ring images on photon detector planes. From the ring images, the opening angle under which the Cherenkov light was emitted is reconstructed, which depends on the momentum, the refractive index of the gas, and the particle's mass. Thus, a momentum known from track reconstruction in combination with a precisely estimated trajectory through the radiator allows distinguishing particles with different masses by their Cherenkov light's angle.

RICH1 is located upstream of the magnet, between the VELO and the UT. It has a radiator with a higher refractive index than RICH2, targeting the identification of particles with momenta below $60\,\mathrm{GeV}/c$. Below $5\,\mathrm{GeV}/c$, RICH1 can also be utilised for electron and muon identification. With its location amidst the tracking system, RICH1's material budget must be as low as possible. The total radiation length fraction amounts to $x/X_0 \simeq 4.8\%$, dominated by the radiator gas and the mirrors. This amount has little impact on the track reconstruction but does not include the beampipe within RICH1. Yet, the beampipe has a conic shape with an edge at the RICH1 exit window [23], which obstructs the track reconstruction in this region as is shown in Section 5.3.1.

RICH2 is positioned between the SciFi tracker and the ECAL. It is designed to provide PID for higher momentum particles, between $15\,\mathrm{GeV}/c$ and $100\,\mathrm{GeV}/c$, using a radiator with a lower refractive index than RICH1. Unlike RICH1, RICH2 does not cover LHCb's whole acceptance, but only angles up to 120 mrad in the x-z plane because particles in the targeted momentum range are produced predominantly at low angles and are hardly bent by the magnet. The material's radiation length fraction of $x/X_0 \simeq 12.4\%$ is irrelevant for the track reconstruction because of RICH2's location downstream of the tracking system. For the matching between tracks and calorimeter clusters, however, the material adds uncertainty when extrapolating the track into the ECAL.

Calorimeters The ECAL is located downstream of RICH2 and is used to identify electrons and photons. It comprises cells with alternating layers of scintillator tiles and lead absorbers to contain and measure the energy and position of electromagnetic showers. In addition to characteristic shower shapes, electrons are distinguished from photons by matching their ECAL clusters to reconstructed tracks.

Behind the ECAL, the HCAL collects the energy from hadronic showers using alternating layers of scintillator tiles and iron absorbers. Because hadrons are identified using the RICH system, the HCAL was mainly used by the hardware triggers during Runs 1 and 2. For Run 3, its role is reduced to shielding the muon stations by stopping hadrons and providing additional PID information only if necessary.

Muon Stations The four muon stations are the detector parts farthest from the interaction point. The first muon station, M2, is located directly downstream of the HCAL and is followed by alternating iron absorbers and stations M3-M5. The large amount of iron stops most particles such that only muons leave signals in all multi-wire proportional chambers. These signals are matched to the trajectory of reconstructed tracks to identify the muons.

2.6 Data Acquisition System and Data Centre

The LHCb data acquisition (DAQ) system, illustrated in Figure 2.8, consists of event builder (EB) servers with AMD EPYC 7502 32-core CPUs hosting custom-made FPGA detector read-out boards (TELL40) and NVIDIA RTX A5000 graphics processing units (GPU) on the Peripheral Component Interconnect Express (PCIe) interface. Because LHCb follows a real-time analysis strategy for Run 3, the DAQ design is trigger-less, *i.e.* the detector is read out at the nominal LHC bunch-crossing frequency of 40 MHz.

Data is transported via optical fibres from the detector front-end electronics in the cavern through a service shaft up to the data centre on the surface (see Figure 2.9) where it is received by the EBs' TELL40s[1]. Each TELL40 only receives data from a single sub-detector and thus has a specific firmware decoding the data, ordering it according to the bunch crossings, building a sub-detector-specific event package and sending it to the network interconnecting the EB servers. To use the TELL40s efficiently, parts of the event data reconstruction for the sub-detectors can be implemented in the FPGA firmware. This is done for the VELO, the pixel data of which are clustered into hits on the TELL40s, as described in Ref. [24]. One EB node collects all sub-

[1]The "40" refers to the detector read-out at 40 MHz. Its predecessor was the TELL1, which readout the detector at 1 MHz.

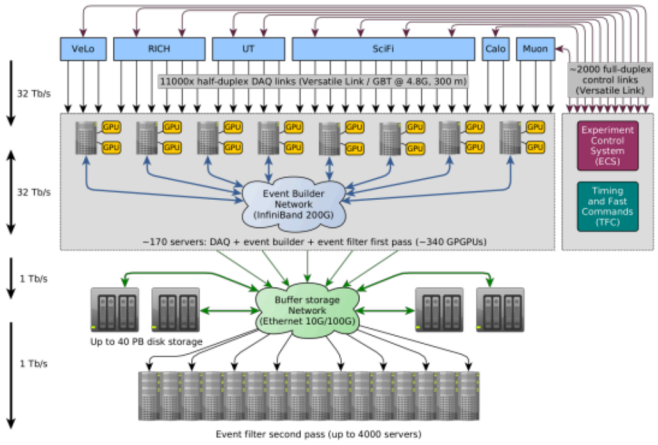


Figure 2.8: Overview of the LHCb data acquisition system [15].

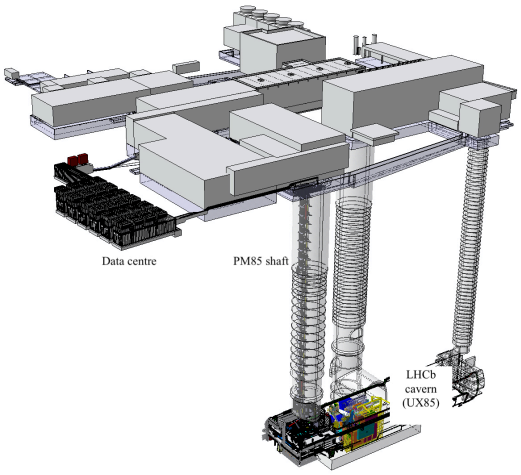


Figure 2.9: Sketch of the LHCb site at Point 8. The readout system is located in the data centre connected with the front-end electronics in the cavern underground through long-distance optical fibres in the PM85 shaft [15].

detector event packages, builds a full event out of them, and stores the full-event packages in a shared memory buffer. From there, the software trigger application Allen (see Section 2.7) performs a partial event reconstruction and selection, outlined in Section 3.2.1, using the GPUs hosted in the EB servers. Events selected by Allen are sent to the buffer storage network where they are kept temporarily until accessed and processed by the alignment and the trigger application Moore, which performs the full event reconstruction and selection, summarised in Section 3.2.2, concluding the real-time analysis of the event. The computing farm running Moore, also called *event-filter farm*, currently consists of more than 3000 general-purpose CPU servers of different kinds and processing power. The farm can be viewed as a computing cloud, with all nodes running the same operating system[1] and no relevant local storage such that each node can be easily replaced. A replacement of the oldest nodes might be reasonable in the future as they have bad performance-to-energy-consumption ratios.

Data processed by the computing farm is subsequently sent to permanent storage, *e.g.* the Worldwide LHC Computing Grid.

2.7 LHCb Software Framework

Most of the LHCb software framework was rewritten and updated to accommodate the changes necessary to run the offline-quality event reconstruction in real-time in the trigger. This goes along with unifying the code base of the trigger and the offline event reconstruction such that offline processing merely becomes a special configuration of the underlying algorithms. The dominant programming language for the back-end

[1]Still CentOS Linux 7, as of writing this thesis.

code is modern C++. The configuration of the components is done in Python. The code targeting CPUs is built on top of the Gaudi framework [25], which is actively developed and used by the LHCb and ATLAS collaborations. The GPU code base is implemented in the cross-architecture Allen framework [26], which can be compiled for execution on CPUs and GPUs. This allows an integration of Allen into the Gaudi framework.

LHCb's real-time analysis strategy risks data loss if the software trigger implementation is erroneous. A code review system, including automated unit and integration tests, is in place to avoid breaking the functionality of the latest software stack [27]. The system is maintained by a team of LHCb developers, in which the author took an active role. The software stack is released and deployed regularly to ensure the reproducibility of physics results.

The LHCb code base is split into several projects, which are versioned and managed independently using the git version control system. The code is publically available in Ref. [28] and distributed under the GNU General Public License v3, except for the code of the Allen project which uses the Apache License v2. The projects relevant to the work presented in this thesis are:

Gauss and Boole To tune and validate reconstruction and selection algorithms, Monte Carlo samples are produced using the Gauss application [29], which interfaces with Pythia 8 and EvtGen for generating the physics processes, and Geant4 for the transport through the LHCb detector. The output of Gauss is processed by Boole to simulate the detector response by converting simulated detector hits into the same format created by the DAQ. Boole, therefore, simulates the electronic response, including noise, cross-talk and dead channels. For the track reconstruction, the effect of radiation damage to the tracking system simulated by Boole is most relevant for efficiency studies. The simulation versions used in this work are given in the text where applicable.

LHCb The LHCb software project contains LHCb-custom implementations of data structures and provides additional framework utilities. In particular high-performance data structures developed and used in this thesis are implemented there. LHCb `v54r3` or later is used.

Rec Code related to the event reconstruction is versioned in the Rec software project. It includes the components for the track reconstruction, reconstruction in the calorimeters, RICH detectors and muon stations, and monitoring of these. Furthermore, the event and decay selections code is located in this repository. Rec `v35r3` or later is used.

Allen The Allen project is a standalone GPU-trigger framework which can also be run within Gaudi to fit into the LHCb software framework. It contains components for the partial event reconstruction and coarse selections. Allen `v3r3` or later is used.

Moore The Moore package is used to configure and run the trigger application.

This includes everything from configuration of the detector hit decoding, track reconstruction and PID to event and decay selections. It is mostly Python-based, easing the development of trigger selections (*trigger lines*) and defining the data flow. Moore `v54r3` or later is used.

3 Analysing LHCb's Data in Real-time

This chapter gives an overview of LHCb's real-time analysis trigger deployed for Run 3 of the LHC. The developments are organised by the Real-time Analysis Project (RTA), in which the author of this thesis took an active developer role with contributions to many of the necessary software components. The main focus, however, lies on HLT2 track reconstruction. This chapter intends to give merely enough information to put the following chapters into a broader context. The reader can follow the references in the next sections for more detailed information. The bigger picture and perspective are informatively drawn in Ref. [30].

3.1 The Need for a Real-time Analysis Trigger

When talking about real-time applications, it is generally good practice to define how the term is used in the application context. For LHCb, real-time is the whole period between a particle-bunch crossing with an inelastic collision and writing the data recorded by the detector to a permanent storage space. This can be a matter of days, a time span most people would not immediately identify as real-time. However, the keyword here is *permanent* storage. The amount of data produced by each of the four large high-energy particle detectors at the LHC is far too enormous to be transported to a remote data centre economically, let alone persisting it on disk or tape. For example, a typical event at LHCb under Run 3 conditions results in around 100 kB of data, which at a pp-collision rate of 40 MHz gives a data rate of 4 TB/s. In 2017, the LHC provided stable beams for 1634 hours [31], so assuming the same availability for Run 3, the amount of data adds up to 18 EB per year. The other three large detectors, covering almost the entire solid angle, produce even more recordable signals, pushing the sum to $\mathcal{O}(100\,\text{EB})$, which is comparable to the monthly global internet traffic [32] and not feasible to store permanently. To cope with this data surge, all large high-energy physics experiments use sophisticated trigger systems that filter out uninteresting collision events. If the processes the experiment wishes to study are sufficiently rare, *e.g.* with a cross-section smaller than 10 nb such as top-quark or Higgs-boson production [33], triggers implemented in hardware performing a fast but coarse classification of the collision event are sufficient to reduce the amount of data to a manageable level and with a high selection efficiency. The processes LHCb was designed to study, *i.e.* production and decays of hadrons containing b or c quarks, have cross-sections of several millibarns in pp collision at $\sqrt{s} = 13.6\,\text{TeV}$. As shown in Figure 3.1, with the foreseen instantaneous luminosity of $\mathcal{L} = 2 \times 10^{33}\,\text{cm}^{-2}\text{s}^{-1}$ (or

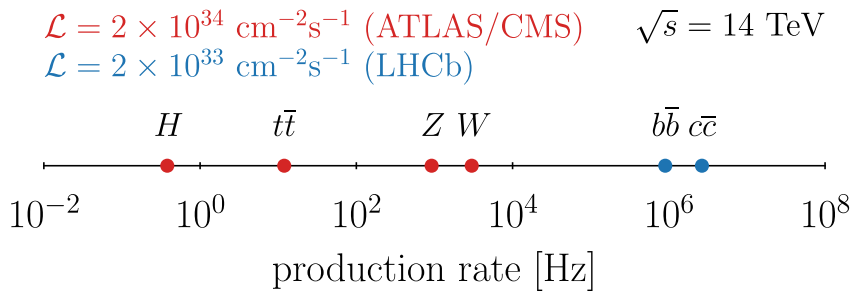


Figure 3.1: Production rates of main particles under study by LHCb and ATLAS/CMS [34].

$\mathcal{L} = 2\,\text{nb}^{-1}\text{s}^{-1}$) provided to LHCb during Run 3, this means that at least every tenth bunch crossing produces an event with a particle decay LHCb would like to study. This comes with two issues for the conventional hardware triggers LHCb employed during Runs 1 and 2. First, their output rate is limited to 1 MHz at which the LHCb detector could be read out. This implies that some interesting decays do not fit into the output bandwidth even with perfect hardware-trigger signal efficiency. This was already observed during Run 2 for hadronic final states from heavy flavour decays as shown in Figure 3.2. Moreover, the higher instantaneous luminosity planned for Run 3 also leads to more combinatorial background. Therefore, the selection thresholds, *e.g.* for transverse momentum, used in a hardware trigger, must be higher, which might even reduce the signal yield for hadronic final states despite higher production rates. For the many statistically limited analyses at LHCb [36], this poses a serious problem because an increase in luminosity no longer translates to an increase in signal statistics if an inclusive trigger is used. Second, even if all interesting decays could be selected and output by the hardware trigger, the amount of storage space needed to persist all their events is beyond what is affordable. Hence, triggering at LHCb is figuratively coined as finding a specific needle in a haystack of needles.

The solution to this problem is to remove the inefficient hardware trigger, read out the detector at the LHC's bunch crossing frequency and perform the offline-quality event reconstruction on the incoming data directly, *i.e.* in real-time, and only select and store the reconstructed decay trees the data analysts want to study. The trigger does not anymore select whole collision events. Instead, only the interesting bit of the collision is carved out, solving the bandwidth and storage problem by drastically reducing the average size of data persisted per event [37]. Performing the offline-quality event reconstruction followed by a fine-grained selection is only possible using a software-based trigger system with a real-time aligned and calibrated detector. Passing the amounts of data mentioned above through such a system is challenging. It requires the reconstruction software to be fast, precise and efficient simultaneously, which is only possible if the detector is aligned and calibrated while taking data. To tackle this

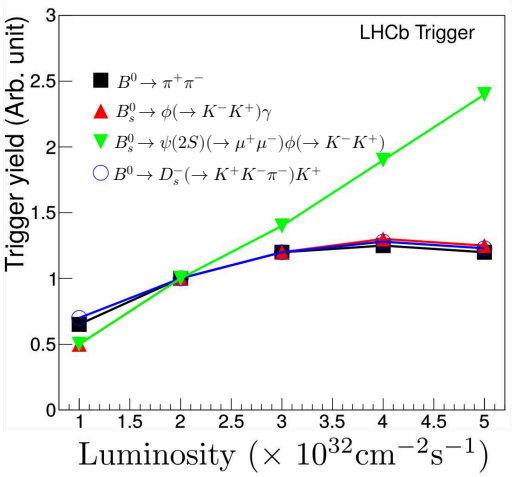


Figure 3.2: Signal yield for several interesting final states of LHCb's Run 1 and Run 2 trigger, relative to the yield at the design luminosity, in dependence on the instantaneous luminosity (adapted from [35]). For hadronic final states (black squares, red upward triangles and empty blue circles) the hardware trigger scheme limits the trigger yield. Only final states with muons (green downward triangles) can be efficiently triggered at higher luminosities.

challenge, the high-level trigger (HLT) is split into two parts separated by a storage buffer, as described in the next section, which allows for stretching the real-time interval up to days in which the events can be analysed automatically before writing their interesting parts to permanent storage.

3.2 Data in Two Steps

The purely software-based trigger developed and commissioned for LHCb in Run 3 is illustrated in Figure 3.3. The detector data coming from the sub-detectors' front-end electronics is transmitted to the data centre (see Section 2.6), where it is read out at the nominal LHC bunch-crossing frequency, *i.e.* every 25 ns, followed by the events being partially reconstructed and selected using the High-level Trigger One (HLT1) software running on GPUs. Because not every bunch crossing results in an inelastic pp collision and not all 25 ns bunches are filled, in practice, HLT1 needs to process events at a rate of about 30 MHz. The HLT1 selections reduce the number of events by roughly a factor of 30, and the raw detector data, together with trigger decision reports, are written to the buffer storage network. The reduction factor depends on the size of the buffer and the amount of data the second trigger stage can process, *i.e.* how fast the buffer can be emptied again. Parts of the HLT1 data are specifically selected for the detector alignment and calibration performed as soon as a sufficient amount of data is collected. In the subsequent data-taking runs, HLT1 can already profit from an aligned and calibrated detector, which is why the term real-time alignment is used.

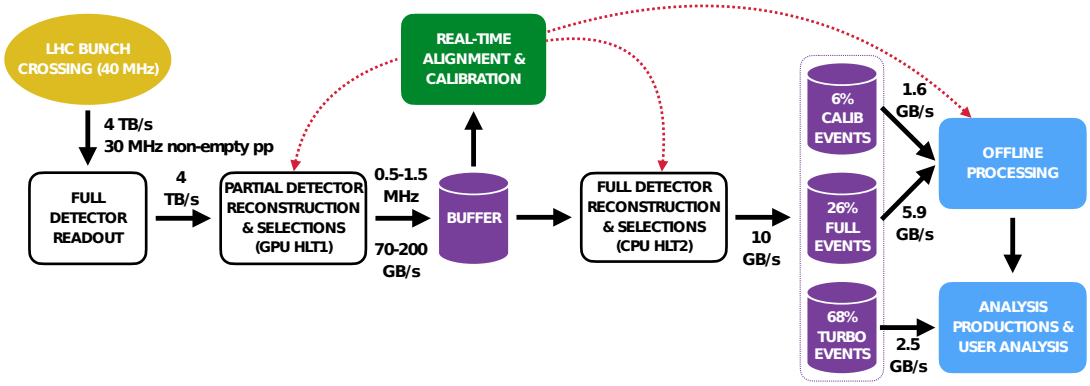


Figure 3.3: Data flow inside LHCb's purely software-based trigger performing a real-time analysis to efficiently select particle decays of interest [38]. The bandwidth numbers and percentages in the right part of the figure are only examples and do not necessarily reflect the actual stream distribution.

The High-level Trigger Two (HLT2) also uses the same alignment when processing the output of HLT1 from the buffer. HLT2 performs the full event reconstruction on CPUs and selects particle decay candidates according to many trigger lines implemented by the data analysts who want to study these decays. The amount of data must have been reduced to 10 GB/s at this point, a bandwidth that can be written to state-of-the-art permanent storage systems. This is done in three streams. Some data is collected for data-driven calibration, such as evaluating tracking and particle-identification efficiencies [39, 40] needed by data analysts. A substantial part of the bandwidth is used to store entire events, *e.g.* selected by an inclusive topological trigger. Yet, these and the offline-calibration data undergo a post-processing step to save disk space further. Most events are immediately available after HLT2 for further physics analyses (*turbo events*), which constitutes the actual real-timeliness of the data.

3.2.1 High-level Trigger One

The first trigger stage has a strict requirement on the time it takes to reconstruct the incoming raw detector data partially: it must process 30 million events per second, on average. Two solutions were developed before the start of Run 3. The first is based on the existing reconstruction code targeting the x86 CPU architecture; the second is based on the new, standalone trigger application Allen running on GPUs. A comparison between the two solutions can be found in Ref. [41]; both are viable for HLT1 but only the GPU solution was commissioned for Run 3.

Allen's partial event reconstruction steps are shown in Figure 3.4. The raw event data is transferred from the detector to LHCb's data centre, packed by event builder nodes and processed by GPUs hosted in the same nodes. Prior to any reconstruction, a

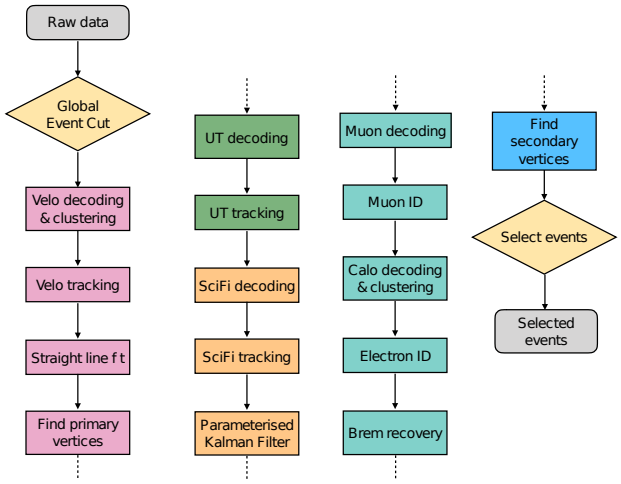


Figure 3.4: HLT1 reconstruction flow [42]. Note that the VELO decoding and clustering on GPU are optional. By default, they are performed upfront on the TELL40s when the event is built.

global event cut (GEC) is applied[1], rejecting around 10% of incoming minimum bias[2] events due to their high occupancy[3] in the tracking detectors. This is motivated by high occupancy events taking a disproportionate amount of resources to be processed, while the reconstruction quality is significantly lower than for other events [43]. For events passing the GEC, a Search by Triplet algorithm [44] finds hits that form straight lines in the VELO detector and fits the corresponding trajectory. The tracks in the VELO predominantly originate directly from the inelastic pp collision and are thus used to reconstruct the primary vertices [45]. Additionally, the straight tracks are extended to the UT detector where decoded UT hits are added to the track using the Compass algorithm [46]. Due to the magnetic field reaching into the UT, these tracks feature a first estimate of their charge and momentum, which is used to extrapolate them into the SciFi detector using parameterisations of the magnetic field. The extrapolation and SciFi-hit finding are performed by the HLT1 Forward tracking [42], which focuses on the reconstruction of tracks with momenta typical for particles created in heavy-flavour decays, *i.e.* $p_{\mathrm{T}} > 500\,\mathrm{MeV}/c$ and $p > 3\,\mathrm{GeV}/c$. For the first data-taking period in 2022, an HLT1 Forward tracking without dependency on the UT detector was also developed, which, however, is restricted to finding tracks with $p_{\mathrm{T}} > 1\,\mathrm{GeV}/c$ and $p > 5\,\mathrm{GeV}/c$ due to the computational performance loss of not having initial momentum information

[1] This is at least still the case as of writing this thesis. The HLT1 GEC will probably be removed soon as enough computing resources are available.

[2] Minimum bias refers to the exclusion of events without a visible pp collision. In LHCb, the term is usually used in the context of simulation in which invisible collisions are not recorded to save resources.

[3] This is estimated based on the size of the SciFi's and UT's raw data.

from the UT.[1] Tracks found by the Forward tracking have a momentum resolution of slightly better than 1%. The momentum estimate is used to improve the track parameter estimates of the track segment inside the VELO detector by using a Kalman filter with a parameterised noise matrix similar to what is described in Ref. [48]. The Kalman filter is not applied to the full trajectory to save processing time and because it was found that fitting the VELO segment is sufficient to allow efficient selections based on vertex information. VELO-(UT)-SciFi tracks are further extrapolated to the Muon stations, where they are matched to hits to identify muons [49]. Electrons are identified by matching their tracks to ECAL cell clusters and potentially correcting their momentum by recovering bremsstrahlung photons. Before a set of trigger lines selects events, pairs of fitted VELO track segments are used to find displaced secondary vertices, a feature of heavy-flavour decays.

The purpose of the HLT1 trigger selection is to reduce the event rate to a level that can be processed by HLT2. The great advantage compared to a hardware trigger is that reconstructed objects combine information from the entire tracking system and parts of the PID system, which allows selections specifically targeting the decay topologies of heavy-flavour decays with cuts the data analysts would also apply offline. Furthermore, trigger selections can be quickly adapted to changing needs, in principle even during data taking. Typical selections require a secondary vertex significantly displaced from any primary vertex and high-momentum tracks which do not point back to any primary vertex. Inclusive selections use one- and two-track neural network classifiers to improve the selection efficiency [26, 50].

The HLT1 reconstruction is performed on the EB servers in the data centre (see Section 2.6), where a single GPU processes roughly 120 thousand events per second. The raw detector data of the selected events and reports about which HLT1 trigger line fired are written to the buffer.

3.2.2 High-level Trigger Two

The second high-level trigger stage runs asynchronously on server CPUs in the data centre's event-filter farm (see Section 2.6). Asynchronously means that the event processing does not necessarily start immediately after HLT1 has written selected events to the buffer and the detector has been calibrated and aligned. The buffer size of around 30 PB was chosen such that HLT1 can take data for up to 80 hours at an output rate of 1 MHz before the storage space is filled to a critical level [15]. The processing delay is a crucial data quality safety measure as data not being selected by HLT2 is lost forever. Therefore, only a tiny fraction of the data on the buffer is

[1] This momentum restriction triggered the development of a standalone SciFi reconstruction on GPU following the recipe of the corresponding HLT2 algorithm. Then, analogously to what is done in HLT2, the SciFi tracks are matched to tracks found in the VELO to reconstruct the same tracks as the Forward tracking [47]. Both options are being commissioned for Run 3 and will likely run in the same arrangement as in the fast reconstruction discussed in Section 4.5.

reconstructed directly, and key quantities like spatial hit distributions in the tracking system, number of reconstructed tracks, and invariant mass spectra are monitored in the control room. If the data and reconstruction quality in the monitoring is satisfying, HLT2 starts to process the respective runs[1]. Nevertheless, the event throughput of HLT2 should roughly match the output rate of HLT1 not to risk filling up the buffer and loose data when the LHC delivers collisions during a long fill.

The offline-quality full-event reconstruction consists of three main components: charged particle track reconstruction, calorimeter reconstruction, and particle identification. The reconstruction of charged particles is described in Section 4.5.1. From the calorimeter system, only the ECAL is reconstructed for event selection purposes; the HCAL has no dedicated use case in the real-time software trigger and is mostly used for monitoring and improving PID performance. ECAL clusters are formed from a 3×3 block of readout cells around an energy peak using a graph-based clustering algorithm [51]. The clusters are classified using multivariate algorithms trained on the shower shapes and individual ECAL cell energies to distinguish single-photon clusters from those containing multiple photons. Electron clusters are identified by extrapolating reconstructed tracks to the ECAL, where matching energy depositions are searched. Some effort was put into optimising the ECAL algorithms for Run 3 regarding their physics and computational performance, documented in Ref. [15]. Identifying electrons, muons, pions, kaons, and protons in HLT2 is done by combining information from the two RICH detectors, the ECAL and the muon stations. The identification relies on reconstructed charged particle trajectory information for all particle species. Similarly to HLT1, muons are identified by extrapolating tracks into the muon stations and searching for hits matching the track. Because HLT2 is less constrained by the processing time, a multivariate classifier is employed additionally to optimise muon identification performance [49]. Identification of pions, kaons and protons mostly relies on matching Cherenkov-light rings to reconstructed tracks as described in Refs. [52, 53]. The information from all sub-detectors is combined into global multivariate PID classifiers, which are used in the trigger selections to reduce background from misidentification of particles.

The reconstructed event information is used by $\mathcal{O}(1000)$ trigger lines, written by the data analysts to select decays of interest exclusively. This is a crucial part of the real-time analysis strategy to cope with the high signal rates and large amounts of data recorded by LHCb. Most selections follow the `Turbo` analysis concept (see `Turbo` in Figure 3.3), which allows full flexibility on the amount of event information that is stored, from merely two tracks and a vertex for a two-body decay to all reconstructed objects in the event if necessary. This increases the rate of events that can be stored by decreasing the data volume persisted for each event significantly [37, 54, 55]. Examples of trigger lines using this reduced persistency model are given in Section 6.5.

[1] A *run* here means a short data-taking period of up to one hour with a unique run number. The shift leader in the control room either starts a run manually, *e.g.* when conditions change, or the experiment control software does it automatically.

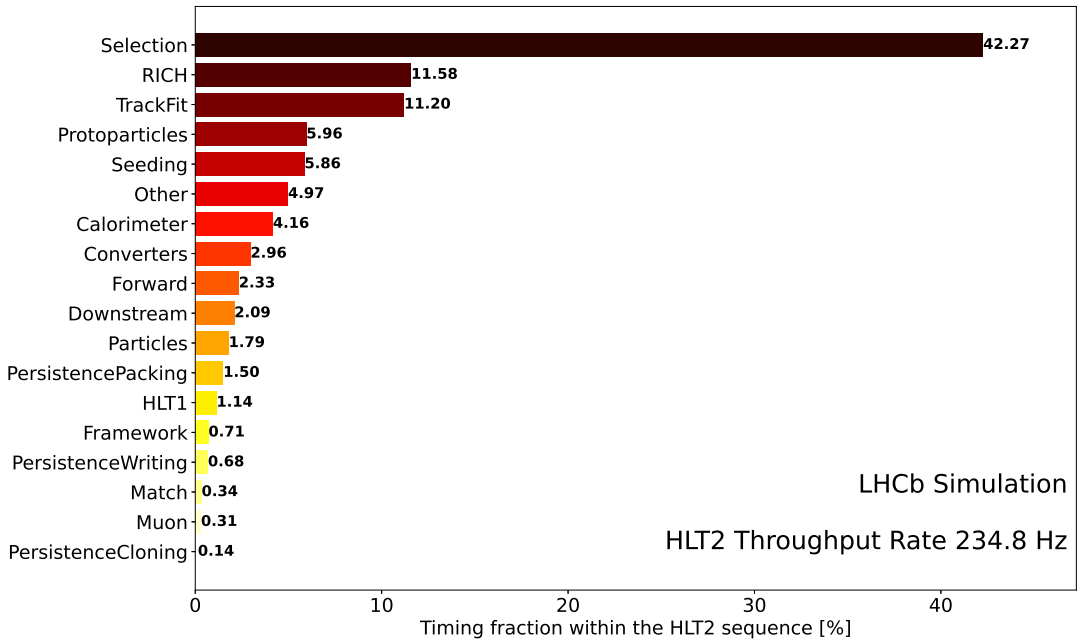


Figure 3.5: Breakdown of each component's HLT2 event throughput using the fast reconstruction sequence (*cf.* Section 4.5).

The selections are the single most time-consuming part of HLT2, as shown in Figure 3.5, measured on a reference node representative for the servers available in the event-filter farm (see Section 4.4.2 for specifics of the computational performance measurements). Developments to improve the throughput of the selections have been made [56, 57], using the performance optimisation techniques summarised in Section 3.3. However, they are still being integrated into the HLT2 software as of writing this thesis. They are needed because, with the current HLT2 throughput of around 235 events per second, and computing node, more than 4200 servers with a performance equivalent to the reference node would be needed to process the design output rate of HLT1. Currently, the event-filter farm consists of around 2400 reference-node equivalents. The lack of computational performance in the selections conversely highlights the tremendous achievements in improving the other components' throughput. In particular, the goal for the track reconstruction was to reach 500 evts/s to fit into the computing budget of HLT2 and thus was optimised extensively.

3.2.3 Online Alignment and Calibration

A crucial ingredient of the real-time analysis trigger paradigm is the online alignment and calibration of the detector to ensure that event selections in HLT1 and particle decay selections in HLT2 can be performed with high signal efficiency. The alignment proceeds in a sequence, with the VELO aligned first, followed by the UT and the SciFi

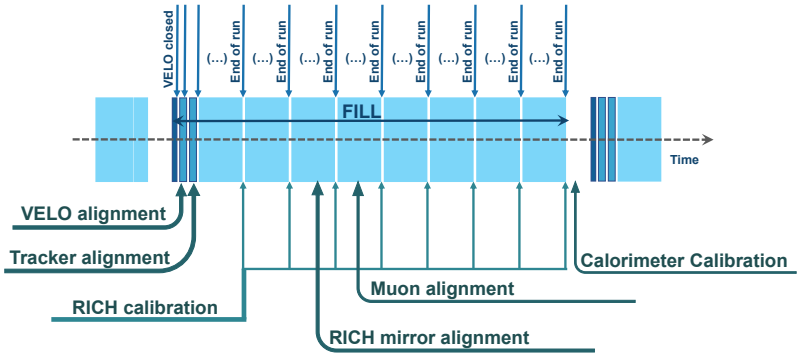


Figure 3.6: Schematic view of the real-time alignment and calibration procedure starting at the beginning of each LHC fill [15].

tracker, the RICH mirror alignment, and finally the muon detector alignment as shown in Figure 3.6 [15].

The alignment constants for the tracking detectors are determined by minimising the χ^2 of reconstructed tracks with respect to the degrees of freedom of each alignable detector element, *i.e.* translations and rotations in each spatial dimension [58]. The VELO alignment is performed at the start of each LHC fill after the VELO was moved from its safe open position to its closed data-taking position close to the beam. It requires a sample of tracks crossing all VELO modules at any azimuthal angle, which is collected in a few seconds after the start of the data taking. The alignment quality is checked by comparing primary vertex positions reconstructed with tracks from the left and right VELO halves, respectively. The UT and the SciFi tracker are aligned next, relying on reconstructed tracks traversing the entire tracking system. Because the SciFi stations are located downstream of the magnet, aligning them is essential to achieve high track-momentum resolution. The alignment can be improved by constraining the kinematics of tracks to the mass of their mother particle. HLT1 selects $D^0 \to K^-\pi^+$ and $J/\psi \to \mu^+\mu^-$ decays for this purpose, large samples of which can be collected within minutes of data taking following the VELO alignment. The best decay to improve the alignment is $Z \to \mu^+\mu^-$ as the muon tracks have high momentum and thus suffer less from multiple scattering. Recording a sufficiently large sample containing this decay, however, takes much longer.

The RICH mirror alignment requires a sample of well-reconstructed tracks to compare the detected photons' Cherenkov angles with the expected angles inferred from the tracks' traversal through the gas. The sample tracks need to be distributed equally among the different RICH1 and RICH2 mirrors, *i.e.* sufficiently many tracks in the peripheral areas of the RICH system must be collected. Consequently, selecting the RICH alignment samples and aligning the mirrors can take up to hours, which is unproblematic as HLT1 does not use the RICH system and the buffer stores the events for HLT2 in the meantime.

The muon station alignment uses muon tracks constrained to the decay $J/\psi \to \mu^+\mu^-$. Updates of this alignment are expected to be rare and necessary only at the beginning of a data-taking period and in case the muon stations were opened.

In addition to the alignment of various detector parts, the ECAL and the RICH system must be calibrated. The ECAL is fine-calibrated about once a month using the observed π^0 mass from the decay $\pi^0 \to \gamma\gamma$ in each calorimeter cell, and its photomultiplier tubes' high-voltages are adjusted after each fill to compensate ageing effects. The gas radiator refractive index of the RICH is sensitive to temperature and pressure variations. It is thus calibrated on a per-run basis while the data is stored in the buffer.

3.3 Tackling the Computational Performance Challenge

Operating a purely software-based trigger system in a large hadron-collider experiment puts much pressure on the computing resources. It is therefore vital to follow best practices of high-performance computing, described *e.g.* in Ref. [59]. Yet most recent developments in this field can be found in blog posts and conference documentation, *e.g.* Refs. [60, 61]. The most fundamental guideline is always that avoiding computations in the first place is the best performance optimisation. In addition, LHCb aims to follow the C++ Core Guidelines [62] throughout its software, which include performance considerations. Beyond that, the computational performance challenge can be tackled by applying techniques broadly falling into two categories:

Physics considerations can lead to performance optimisations if they allow for avoiding computational work. This might be, for example, a simplified description of the particle motion in the magnetic field by using parameterisations if their precision is sufficient. Also, the geometry or objective of the experiment might limit possible or necessary measurements. The software should not waste resources by trying to measure a physically impossible, improbable or undesired process. While this can significantly improve computational performance without drastically changing the codebase or hardware, it might impact physics performance, and tradeoffs must be studied.

Hardware exploitation concerns efficient use of available hardware resources. For the event-filter farm, this means fully utilising the capabilities of modern CPUs with their multicore architecture, on-chip accelerators, and other CPU features described below. In general, this may also include accelerators like GPUs and FPGAs. The physics performance is usually preserved when the computational performance is improved by more efficient hardware usage. Yet it does not come for free as deep changes in the codebase are often necessary to interface with the accelerators, which has staff costs in addition to the potentially expensive hardware.

A simplistic example of these two categories is the structure of HLT1. The partial event reconstruction performed in HLT1 mostly considers high-momentum physics in the sense that the algorithms are tuned to reconstruct high-momentum tracks and avoid the work involving reconstructing the low-momentum regime, hence performing an optimisation from the first category. Running the HLT1 reconstruction on GPUs belongs to an optimisation of the second category. Details on meeting the computing challenge in HLT1 can be found in Ref. [43].

More complex examples of the optimisations from the two categories can be found in HLT2. The following references examples of the first category and describes techniques belonging to the second category proven helpful for the performance optimisation of the HLT2 event reconstruction on CPUs.

Optimisations from the first category can be found throughout Section 5.2, where it is shown that segments of the particle trajectory or track parameters can be approximated by polynomials instead of numerically solving the equations of motion. Other physics-motivated optimisations include introducing an artificially increasing granularity of the SciFi tracker towards its edges where the particle flux is lower (see Section 5.2.5) and the division of the SciFi tracker into two distinct halves according to the properties of the magnetic field (see Section 5.2.3).

In the second category, only optimisations available for CPUs are relevant for the HLT2 track reconstruction. Yet, the concepts often also find applications on other architectures like GPUs.

A key feature of modern CPUs is their multicore architecture which allows the execution of several applications in parallel or splitting the workload of a single application into pieces processed in parallel. LHCb's Gaudi-based C++ software framework used to implement the trigger application defines functional algorithm templates, which model functions with no side effects and return results only depending on their explicitly declared inputs [63, 64]. This ensures thread safety so the trigger application can process events in parallel via multi-threading. Each thread manages an event-local storage space keeping the input and output data of the functional algorithms and possibly storing temporary algorithm-internal data if requested. This avoids costly memory (de-)allocations and locks that could otherwise block the execution of other threads.

Since the multi-threading is handled on the framework level, developers can focus on more low-level optimisation of the software components. For the track reconstruction, it was vital to understand and use modern CPU features beyond their multicore architecture. The logical CPU core running a thread uses multiple processing units, each dedicated to a single task, such as arithmetic or memory operations. When the thread executes a specific instruction on one unit, other units can be used to process other instructions in parallel. This is called pipelining[1], is done automatically by the processor and increases the number of instructions the CPU can process per clock cycle.

[1] A CPU with this capability is called *superscalar*.

It, however, only works efficiently if the pipeline can be filled with enough independent instructions. The CPU and the compiler reorder[1] instructions from the program such that independent tasks are clustered together, and the entire pipeline is used if possible. This includes the speculative execution of code branches, *e.g.* due to an if-statement, based on the decision of the CPUs branch predictor. The branch predictor follows static rules to detect simple patterns like the backward jump at the end of a loop but can also factor in dynamic patterns by saving the history of previous branch outcomes. If the branch predictor succeeds, the pipeline can be filled with the correct instructions. If the conditions are not predictable, the speculative execution is harmful because a pipeline containing the wrong instructions must be flushed and re-filled with the correct ones, introducing a latency of typically 15-20 CPU cycles that can significantly throttle the whole program. In that case, the bottleneck can often be resolved by reordering the code so conditions become transparent for the branch predictor. A well-known algorithm where this is not possible is binary search. However, binary search can be implemented such that branches are avoided altogether by predication, *i.e.* execution of both branches followed by selecting the correct result. In performance-critical parts of the event reconstruction, a branchless implementation of binary search [65] is used to optimise the CPU's instruction throughput and, consequently, the component's speed. To keep the pipeline filled in general, it should be avoided to implement long interdependent calculations and tight loops with dependencies on calculation results from the previous iteration.

A more direct way to optimise performance is to use processing units that execute a single instruction on multiple data (SIMD). Unlike pipelining, this optimisation must be explicitly used in the implementation[2] and is often referred to as *vectorisation*. Different instruction sets using the SIMD paradigm are available depending on the CPU architecture. LHCb's reconstruction software targets the Advanced Vector Extension 2 (AVX2) instruction set operating on 256-bit SIMD registers, *i.e.* processing eight single or four double precision floating point or integer numbers at once, illustrated in Figure 3.7. The most abundantly used vectorisation library in LHCb is the SIMDWrapper [66], explicitly developed for LHCb's needs.

Additionally, the instruction set is extended by fuse-multiply-add operations (FMA), which combine a multiplication followed by an addition into a single micro-operation. This yields performance improvements, especially for the evaluation of polynomials which are heavily used throughout the track reconstruction, *e.g.* for parameterisations mentioned above.

However, two other concepts need to be considered for vectorisation to yield significant performance improvements.

The first concept deals with the fact that a significant bottleneck for CPU calculations is the speed with which data in the memory can be accessed. All modern CPUs,

[1] This is called *out-of-order* execution.

[2] Under certain circumstances, the compiler can also auto-vectorise code. However, it is difficult to rely on that as it depends on the compiler versions and the exact implementation of the program.

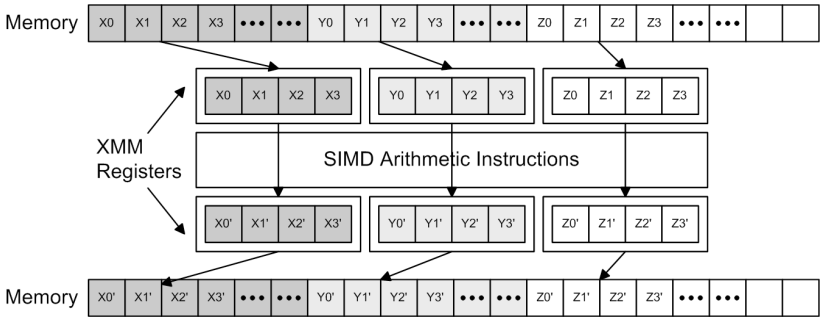


Figure 3.7: Illustration of data processing using SIMD instructions [67].

therefore, embed small but fast memory storage directly on the die to cache data from the main memory. The CPU cache is organised in multiple hierarchical levels, each increasingly large but with decreasing access speed as shown in Figure 3.8. For computations depending on data not already in a register, the CPU checks if the data is in the L1 cache. This will, for example, be the case if the same data or data close in memory was already used shortly before. The L1 data cache has a size of usually 32 KiB per core and a latency of around 4 cycles or 1 ns. It thus can hold one thousand single-precision values, which is too few for the track reconstruction that operates on several thousands of spatial position measurements, *e.g.* in the SciFi tracker. The goal is, therefore, to have most quantities used by the tracking in the L2 cache, which typically has a size of 256 KiB per core and a latency of around 10 cycles. If the data is not cached, a *cache miss* occurs, and the data must be fetched either from a higher cache level or from main memory, but in any case, with a higher latency. Already the L3 cache often has a latency of several hundreds of cycles as it is shared between cores. The latency also occurs for writing values to the cache. Therefore, any memory operation needs to be handled with care so that it does not become a severe bottleneck.

The CPU fetches and writes data in units of 64 B called a cache line. Hence, for performance optimisation, it is essential to keep data being processed together also

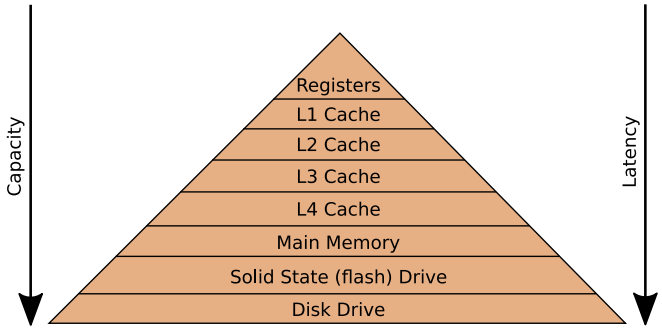


Figure 3.8: Sketch of the cache hierarchy, including CPU registers and larger storage devices ordered with increasing storage capacity and access latency from top to bottom [66].

stored together (temporal and spatial locality), preferably within the same cache line, but at least within the two fastest cache levels to avoid long latency. This directly leads to the second concept, which defines the data layout in memory according to the typical access pattern when using SIMD instructions.

The usual human-friendly data layout used in object-oriented languages like C++ packs data into a structure reflecting a higher-level object, *e.g.* a detector hit containing two positions x and y, and a weight w. Storing many hits in a container, this conceptual layout is also the layout in memory with each x-y-w collection next to each other, called an Array of Structures (AoS) shown in Figure 3.9. Suppose the program performs calculations on single hits using these three quantities. In that case, this layout is cache-efficient following spatial locality because the CPU will likely fetch all three using one cache line. If, however, many hits are processed, calculations can be optimised using the SIMD instructions. For this, the AoS layout is suboptimal because multiple x values cannot be directly loaded as a block to the SIMD registers to apply a single instruction. Therefore, a Structure of Arrays (SoA) layout is preferred, illustrated in Figure 3.9. This memory layout allows directly loading several values into the SIMD registers using a single cache line.

Related to these two concepts is the choice of the basic data type used for calculations. For most components in the event reconstruction, it is sufficient to use single precision data types for calculations.[1] This is beneficial for the performance because twice as many numbers fit in both the cache line and the SIMD register compared to double precision.

The techniques mentioned above are merely a subset of possible ways to tackle the computational performance challenge posed by LHCb's software trigger but are the ones proven successful during the development. To prove a technique successful, it is crucial to carefully measure the performance, *e.g.* in terms of execution time. For this, code profiling tools must be used to get to the root of an observed performance bottleneck. Within LHCb, Intel's VTune Profiler, Callgrind and Linux's `perf` are common and were also used for performance optimisation of the algorithms presented in this thesis.

[1] An exception is, for example, the current implementation of the Kalman filter. The calculations performed with the covariance matrix are slightly unstable using single precision. However, ways around this, *e.g.* storing the Cholesky decomposition of the matrix, are being discussed to optimise the performance.

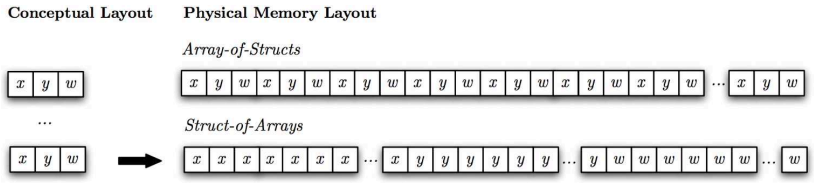


Figure 3.9: Illustration of the two different memory layouts for data [66].

4 Tracking Down Charged Particles

The reconstruction of charged particle trajectories is a crucial component in the event analysis of high-energy physics experiments. It aims to estimate the momentum vectors of single particles along their path through the experiment's detectors. To achieve this, tracking detectors record ionisation signals left by the particles. Specialised methods then group these signals to form track candidates. This is referred to as *track finding* and often uses well-known techniques from pattern recognition, a comprehensive summary of which can be found in Refs. [68, 69]. Although generalised approaches for common tracking problems across different experiments are actively developed [70], the peculiar constraints of LHCb's high-level trigger, described in Chapter 3, and the unique geometry makes it, for now, favourable to implement fully custom solutions. This chapter, therefore, gives an overview of the charged particle trajectory reconstruction at LHCb and discusses physical processes that govern the particle's traversal of the detector.

4.1 On the Importance of Track Reconstruction at LHCb

When LHCb was first proposed [71], its design was largely influenced by the then-contemporary HERA-B experiment [72], which would operate under similar conditions. The study of B mesons being the main objective of the experiments, it was clear that a high track finding efficiency, precise track reconstruction combined with excellent vertex resolution is necessary to trigger on and fully reconstruct the final states of interesting decays, which often contain more than four particles. To study the CP violation in the decay $B_s^0 \to J/\psi\phi$, for example, two muons from the decay $J/\psi \to \mu^+\mu^-$, two kaons from the decay $\phi \to K^-K^+$, at least one more particle to tag the flavour of the B_s^0 meson, and the primary vertex have to be accurately reconstructed. Assuming an average track reconstruction efficiency of 90% for the daughters of $B_s^0 \to J/\psi\phi$ and 75% for a soft kaon used for tagging, one out of two decays are not fully reconstructed on average, which thus significantly lowers the statistics of an already statistically limited study [73]. Reconstruction can be challenging because large fractions of the final state particles have total momenta smaller than 10 GeV/c and transverse momenta well below 2 GeV/c. In this lower momentum regime, multiple scattering in the detector material becomes an issue, making particularly robust track reconstruction algorithms necessary. At first glance, a tracking system with many stations and layers eases the track finding because the more measurements constrain the trajectory, the better it can be distinguished from random signal combinations. This can help find low-momentum tracks particularly as

they typically need more measurements to constrain their parameters. However, the lesson learnt from HERA-B was that large radiation and nuclear interaction lengths could severely obstruct the event reconstruction in a crowded hadronic environment, such as collisions at the LHC [74]. The more detector material there is, the more final state particles interact with the material, either altering their trajectory or stopping them in a hadronic interaction (see also Section 4.2.1), which produces secondary particles, complicates the event reconstruction and diminishes the available statistics. This is why the initial LHCb design was re-optimised by reducing the material budget of the tracking system [23] to a minimal amount necessary and still achieving high track reconstruction efficiency by focussing on the implementation of efficient track reconstruction algorithms. But also in LHCb's Run 3 design, still 13% of the kaons are lost due to hadronic interactions before reaching the end of the tracking system [75], additionally penalising the decay-reconstruction yield. Then, with LHCb's real-time analysis strategy, the track reconstruction does not only need to be robust and efficient in a light-weight detector, but also the computing time of the algorithms becomes more crucial than ever as track reconstruction always posed one of the most time-consuming components of the event reconstruction.

4.2 Passage Through the Tracking System

Before diving deeper into algorithmic track reconstruction of charged particles at LHCb, it is useful to understand how particles interact with the tracking system itself.

4.2.1 Material Effects

Although one of LHCb's design objectives is to keep its material budget small, any matter that is used to detect charged particles still needs to interact with them via *ionisation* or atomic excitation, *i.e.* the particle transfers a small amount of energy to the detector material's electrons. For track reconstruction, it is important to note that ionisation does not significantly change the momentum direction of the particle if it is much heavier than an electron. The amount of energy deposited in the detector material depends on the particle's charge and momentum and the material's properties. It is described by the Bethe equation, which has its minimum close to the lower end of the momentum range observed by LHCb at $1\,\mathrm{GeV}/c$ and slowly rises towards higher momenta [76]. Assuming a minimum-ionising particle[1] traversing ten sensors of the VELO detector or 2 mm of silicon, an energy of approximately 1 MeV is lost. For track finding, energy losses of $\mathcal{O}(1\,\mathrm{MeV})$ are negligible because the effect of multiple elastic *Coulomb scatterings* off the atomic nuclei, further referred to as *multiple scattering*, is dominating the momentum resolution.

[1] $-\langle\frac{\mathrm{d}E}{\mathrm{d}x}\rangle \simeq \rho \times 2\ \mathrm{MeV\,cm^2/g}$ with *e.g.* $\rho = 2.3\,\mathrm{g/cm^3}$ for silicon.

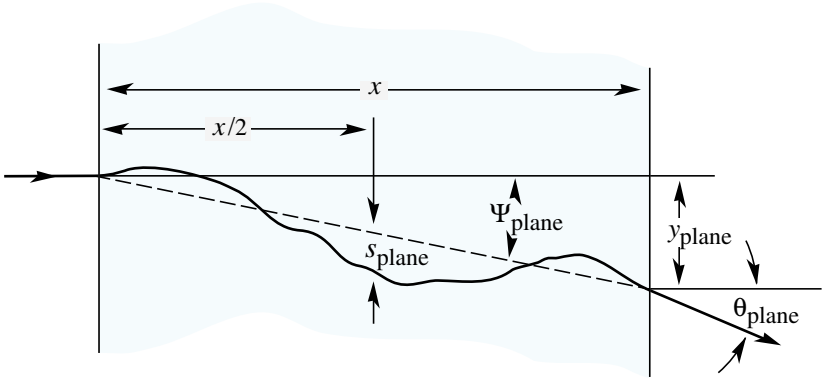


Figure 4.1: Illustration of multiple scattering of a charged particle on its way through some material [76].

Conversely to the ionisation described before, multiple scattering in thin[1] scatterers only changes the direction of the momentum vector. This, however, complicates the track finding and limits the precision of the track parameter estimation for particles up to a momentum of 80 GeV/c [19], after which the track bending by the magnetic field is small, and the tracking detector resolution becomes the dominating limitation. To estimate the effect of multiple scattering on track reconstruction, the distribution of small scattering angles can be approximated by a Gaussian distribution as done in the Highland formula [77] and its modified version from Refs. [76, 78], which describe the standard deviation by:

$$\sigma(\theta_{\text{plane}}) = \frac{13.6\,\text{MeV}}{\beta cp} z \sqrt{\frac{x}{X_0}} \left[1 + 0.038 \ln\left(\frac{xz^2}{X_0\beta^2}\right)\right] \tag{4.1}$$

where p, βc and z are the particle's momentum, velocity and charge, and x/X_0 the distance travelled in the scatterer in units of its radiation length (see Figure 4.1 for illustration).

While Equation 4.1 in principle also holds for electrons, their low mass leads to particularly strong acceleration in the field of the nucleus, and above an energy of $\mathcal{O}(10\,\text{MeV})$ they predominantly lose energy via *bremsstrahlung* [76]. On average, the energy loss is described by [68]:

$$\langle E(x) \rangle = E_0 \exp\left(-\frac{x}{X_0}\right) \tag{4.2}$$

where E_0 is the initial energy of the electron and x/X_0 the distance travelled in the scatterer in units of its radiation length[2]. Summing up the radiation lengths for the

[1] In a thin scatterer, the change in position of the scattered particle is negligible compared to the spatial resolution of the detector system, which is the case for LHCb's tracking system.

[2] Strictly speaking, this X_0 is not the same as used in Equation 4.1, however, for an argument here it

VELO, RICH1 and UT given in Sections 2.4 and 2.5, an electron with an initial energy of 3.5 GeV thus loses 1 GeV before the magnet on average. The distribution of the energy loss is not Gaussian, and the amount of energy lost is subject to large fluctuations [68], which makes electron track finding and parameter estimation more involved because it is difficult to predict and follow the trajectory through the magnet and the energy loss biases the momentum estimate.

In addition to the electronic material interactions described above, hadrons also participate in *hadronic interactions* with the atomic nuclei in the detector material. These strong scattering processes either occur elastically and contribute to multiple scattering or inelastically, in which case the incident particle is destroyed, and secondary particles are created. Destructive hadronic interactions have a much higher cross section at energies above 1 GeV than elastic ones [79] and thus can preclude the track reconstruction if the mean free path through the detector material is too short. Even with a light detector design as the one of LHCb, 13% of kaons and 21% of pions from B or D meson decays are lost for track reconstruction because they undergo a hadronic interaction before the RICH2 detector [75].

4.2.2 Motion in Magnetic Field

The field of LHCb's dipole magnet is strongest between the front of the UT detector and the end of the last SciFi station (see Figure 2.3). With known field values and directions $\mathbf{B}(\mathbf{x})$ at points in space $\mathbf{x}$, the trajectory of a charged particle traversing the field is described by the equations of motion given by the Lorentz force:

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = \frac{\mathrm{d}(m\gamma \mathrm{d}\mathbf{x}/\mathrm{d}t)}{\mathrm{d}t} = c^2 \kappa q [\mathbf{v}(t) \times \mathbf{B}(\mathbf{x}(t))] \tag{4.3}$$

with the momentum vector $\mathbf{p}$, particle rest mass m and signed charge q, the relativistic Lorentz factor γ, the speed of light c, velocity $\mathbf{v}$, time in the laboratory frame t and a factor κ for convenient units[1]. As the main purpose of the magnetic field is to allow an estimate of the particle's momentum from geometrical measurements along its path, it is useful to change the variable of the differential equation Equation 4.3 to the path length $s(t)$ (see Ref. [69] for derivation):

$$\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}s^2} = \kappa \frac{q}{p} \left[\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s} \times \mathbf{B}(\mathbf{x}(s)) \right] \tag{4.4}$$

where $p = |\mathbf{p}| = m\gamma\beta c$ is the absolute momentum in the laboratory frame. The forward geometry of LHCb following the beam along the z axis further makes it convenient to express the differentials in Equation 4.4 with respect to the z coordinate using $\mathrm{d}s/\mathrm{d}z = \sqrt{1 + (\mathrm{d}x/\mathrm{d}z)^2 + (\mathrm{d}y/\mathrm{d}z)^2}$. The result of the derivation, documented in

is convenient and fine to use the same quantity.

[1] $\kappa = 0.299\,792\,458\,\mathrm{T}^{-1}\mathrm{m}^{-1}\,\mathrm{GeV}/c$ for high-energy physics units.

Appendix A.1, is:

$$\frac{\mathrm{d}t_x}{\mathrm{d}z} = \kappa\frac{q}{p}\sqrt{1+t_x^2+t_y^2}\left[t_y(B_z + B_x t_x) - B_y(t_x^2+1)\right] \tag{4.5}$$

$$\frac{\mathrm{d}t_y}{\mathrm{d}z} = \kappa\frac{q}{p}\sqrt{1+t_x^2+t_y^2}\left[-t_x(B_z + B_y t_y) + B_x(t_y^2+1)\right] \tag{4.6}$$

with $t_x \equiv \mathrm{d}x/\mathrm{d}z$ and $t_y \equiv \mathrm{d}y/\mathrm{d}z$. While B_x is negligible in LHCb's magnet, Equation 4.6 shows that both B_y and B_z alter the y-z trajectory of the particle if it has significant slopes t_x and t_y. The strongest field component is B_y, *i.e.* the momentum measurement is most sensitive using the bending of the track in the x-z plane[1] and can thus be measured well by comparing the slopes t_x before and after the magnet. From Equation 4.5 follows:

$$\frac{q}{p} = \frac{t_x(z_2) - t_x(z_1)}{\displaystyle\int_{z_1}^{z_2} \sqrt{1+t_x^2+t_y^2}\left[t_y(B_z + B_x t_x) - B_y(t_x^2+1)\right]\mathrm{d}z} \tag{4.7}$$

where z_1 and z_2 are positions at which the track's slope is known, typically the end of the VELO and the SciFi detector. The integral in the denominator has to be solved numerically or approximated, as LHCb's magnetic field is inhomogeneous. Moreover, Equation 4.5 hints at the parameters that fully describe the motion of a particle in the LHCb coordinate system; there are six constants of integration, one of which is constrained by $|(\mathrm{d}\mathbf{x}/\mathrm{d}s)|^2$ being constant, leaving five free parameters at a reference plane z_{ref}. A natural choice for the definition of a track state vector in LHCb's setup is:

$$\mathbf{s} = (x \;\; y \;\; t_x \;\; t_y \;\; q/p)^\top_{z_{\mathrm{ref}}} \tag{4.8}$$

As discussed in Section 4.2.1, multiple scattering limits the precision with which these parameters can be estimated. The momentum resolution, in particular, is bound from below by the amount of material between the end of the VELO and the SciFi detectors. For a particle with $p = 2\,\mathrm{GeV}/c$, using the fractions of radiation lengths given in Sections 2.4 and 2.5 together with an estimate for the air in the cavern[2], Equation 4.1 yields $\sigma(\theta_{\mathrm{plane}}) \simeq 3$ mrad. Taking this as the error on the slope difference in Equation 4.7 and a mean slope difference of $\langle\Delta t_x\rangle_{p\simeq 2\,\mathrm{GeV}/c} \simeq 0.62$ rad, the achievable momentum resolution with the LHCb tracking system is $\Delta p/p \simeq 0.5\%$.

[1] Therefore also known as the *bending plane* of LHCb.
[2] $X_0^{\mathrm{air}} \simeq 36.1\,\mathrm{gcm}^{-2}/1.2\,\mathrm{kgm}^{-3} \simeq 301\,\mathrm{m}$ [80] $\implies x/X_0^{\mathrm{air}} \simeq 2\%$

4.3 Menu of Track Types at LHCb

Tracks reconstructed in the LHCb detector come from charged kaons and pions, protons, muons and electrons. Charged kaons and pions can be tracked because, at the energies of the LHC and the length of the path a particle takes through the LHCb detector, they usually decay outside of the detector volume. Only 5% of charged pions and 12% of charged kaons from B- or D-meson decays are lost for the track reconstruction because they decay inside the detector [75]. Independently of the particle species, the set of tracking detectors used to reconstruct a particle's trajectory defines its LHCb track type. As shown in Figure 4.2, the track types are:

VELO tracks are reconstructed from hits in the VELO detector only. They are used for primary vertex reconstruction, seeding of other track types and reconstruction efficiency studies. Without external constraints, no momentum measurement is possible for these tracks as the magnetic field is too weak in the VELO detector.

T tracks are reconstructed from hits in the SciFi detector only. They are used as seeds for the reconstruction of other track types and for studies of very long-lived hadrons such as the Λ hyperon. Because of the fringe magnetic field at the T stations, only a rough momentum estimate is possible, with a resolution of 25%-35% [81].

Upstream tracks are VELO tracks with an extension into the UT detector. They often stem from very low-momentum particles bent out of LHCb's acceptance by the magnetic field. Using the fringe field upstream of the magnet, a momentum resolution of 15%-25% is reached [82]. These tracks can also be referred to as UT-filtered VELO tracks depending on the configuration of the algorithm producing them (see Section 4.5.1).

Downstream tracks have hits in the SciFi and the UT detectors. They are important for the reconstruction of decays of long-lived hadrons such as $K^0_S \to \pi^+\pi^-$.

Long tracks have hits in the VELO and SciFi detector, and optionally also in the UT stations. They reach the best momentum resolution of 0.5% on average and are therefore best suited for precise measurements.

In addition to the physics track types described above, there are two more track-like objects important for the reconstruction:

Fake tracks consist of hits that cannot be associated with a single particle. They are therefore also called *ghost tracks* and can only be identified in simulation where the origin of hits on a track is known. In physics analysis, they are a potential background source. See also Section 4.4.1.

Clone tracks are copies or subsets of other tracks, *i.e.* they share a substantial part of detector hits. A clone track cannot be a fake track and thus must be associated to a single particle in simulation.

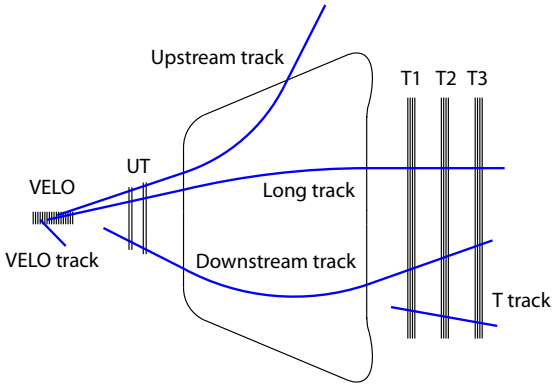


Figure 4.2: Visualisation of different track types reconstructed by the LHCb tracking system [75]. The x-z plane is shown, with the magnet yoke drawn in the centre.

4.4 Reconstruction Performance Metrics

To evaluate and compare the performance of different track reconstruction approaches, it is important to clearly define the metrics for which the reconstruction will be optimised and, furthermore, identify those that conflict with each other. The metrics relevant for track reconstruction fall into two categories, physics performance metrics and computational performance metrics, which are not independent of each other. For HLT2, with its aspiration to provide the highest-quality full-event reconstruction, there is no obvious working point in the multidimensional metric space as briefly outlined in Section 4.4.3. If not stated otherwise, results for the physics metrics are obtained from a mixture[1] of simulated[2] pp-collision events containing the decays $B_s^0 \to \phi\phi$, $B_s^0 \to J/\psi\phi$, $B^0 \to K^{*0}e^+e^-$ and $D^+ \to K_S^0\pi^+$. The samples were re-digitised to simulate radiation damage in the VELO and SciFi detectors equivalent to $5\,\text{fb}^{-1}$ integrated luminosity seen by the experiment to obtain realistic estimates. These decays are chosen because they reflect the core part of LHCb's physics programme and provide various track and particle species in the final states. The computational performance is evaluated on simulated HLT1-filtered minimum-bias samples[3] with a global event cut at a sum of 11000 UT and SciFi clusters.

[1] Also both magnet polarities are present in the samples, with 5000 events each.
[2] LHCb simulation version `Sim10-Up08-Digi15-Up04`.
[3] LHCb simulation version `Sim09c-Up02`.

4.4.1 Physics Performance

The two most important physics performance metrics for track reconstruction using pattern recognition at LHCb are the *track finding efficiency* and *fake track rate*. The track-finding efficiency is defined as

$$\varepsilon = \frac{N_{\text{reconstructible\&MC-matched}}}{N_{\text{reconstructible}}} \tag{4.9}$$

where N denotes the number of tracks in the category specified by the subscript. These categories are only well-defined using simulated tracks. A track is defined as *MC-matched* if 70%[1] of the hits it is made of belong to the same simulated particle; application of this criterion is further referred to as *truth matching*. The reconstructibility of a track is defined by the minimal amount of signals the simulated particle must have left in the tracking detectors. The amount depends on the track type, was optimised when the experiment was designed and is documented in Ref. [75]. Because electrons undergo bremsstrahlung and their track reconstruction is thus more challenging, they are excluded from the ratio yielding the efficiency for other particle species. For a Long track to be reconstructible, the simulated particle must traverse at least three VELO sensors leaving at least one hit in each and have a hit at least in one x and one stereo layer in each of the three SciFi stations. Related to the track finding efficiency, but subtly different, is the *track reconstruction efficiency*. As described in Section 4.5, track finding and precise determination of track parameters are done separately in LHCb, and a found track can still be rejected after its parameters and covariance matrix have been estimated by the track fit. The likelihood that a track is available for physics analyses is, therefore, better estimated by

$$\varepsilon_{\text{reco}} = \frac{N_{\text{reconstructible\&MC-matched\&fit}}}{N_{\text{reconstructible}}} \tag{4.10}$$

where the numerator is the number of found tracks that survive the track fit. It immediately follows that $\varepsilon_{\text{reco}} \leq \varepsilon$ and it is an implicit metric that the quality of the tracks found by the pattern recognition should be such that $\varepsilon_{\text{reco}} \simeq \varepsilon$.

If a track does not fulfil the truth-matching criterion, it is assumed to be a fake track. The fake track fraction, or ghost rate, is defined as

$$r_{\text{fake}} = \frac{N_{\text{fake}}}{N_{\text{tot}}} \tag{4.11}$$

where N_{fake} is the number of tracks that could not be associated with a simulated particle, *i.e.* it is not truth-matched, and N_{tot} is the total number of tracks produced by the algorithm. Similarly, the *clone fraction* is defined as the number of clone tracks

[1] This is an arbitrary choice made by LHCb and leads to uncertainties on reconstruction efficiencies used in data analyses if determined by simulation, also known as truth matching inefficiency.

relative to the number of truth-matched tracks.

To quantify the reconstruction quality on an ensemble of single tracks, the *hit efficiency* and *purity* for truth-matched tracks are defined as

$$\epsilon_{\text{hit}} = \frac{\sum_i \frac{n_{\text{i,found,true}}}{n_{\text{i,tot,true}}}}{N_{\text{reconstructible\&MC-matched}}} \tag{4.12}$$

and

$$f_{\text{pure}} = \frac{\sum_i \frac{n_{\text{i,found,true}}}{n_{\text{i,tot}}}}{N_{\text{reconstructible\&MC-matched}}} \tag{4.13}$$

where $n_{\text{i,tot}}$ denotes the total number of hits on a single reconstructed track i, $n_{\text{i,found,true}}$ is the number of found hits that truly belong to the particle matched to track i, and $n_{\text{i,tot,true}}$ is the true total number of hits the simulated particle left in tracking detectors. The sum runs over all tracks accounted for by the denominator.

Eventually, when a truth-matched track is found, it is useful to evaluate how precise its momentum is estimated. The momentum resolution of truth-matched tracks is determined by

$$\Delta p/p = \frac{p_{\text{true}} - p}{p} \tag{4.14}$$

where p_{true} is the momentum of the simulated particle. For the track finding, however, this is merely a cross-check as the previously mentioned metrics already evaluate the correctness of the reconstruction.

In summary, a well-performing track reconstruction algorithm has high track reconstruction efficiency, low fake track rate, high hit efficiency and purity, and a momentum resolution close to the limitations posed by the detector resolution and multiple scattering. Yet, these metrics do not give insight into the importance of individual tracks. The exact working point has to be defined according to the objectives of the whole experiment, *e.g.* prioritising tracks from heavy-flavour particle decays over light-flavour ones.

4.4.2 Computational Performance

The computational performance of the event reconstruction is measured in event throughput per computing node. This metric is useful because it naturally translates to the rate of *pp*-collision events provided by the LHC and can be thought of as the diameter of the event processing pipeline; the pipeline must be large enough to put all interesting events through it. To estimate and compare the expected performance of reconstruction algorithms in the HLT computing farm, a reference node with two fully-loaded[1] Intel Xeon E5-2630 v4 CPUs is used to measure the time t it took to process N events. If not mentioned otherwise, the GCC11.3 compiler with O3

[1] 2×20 threads, each on its own NUMA domain.

optimisation and AVX2 and FMA enabled is used. The event throughput rate on the reference node is then defined as

$$R_{\mathrm{TP}} = \frac{N}{t} \quad [\mathrm{Hz\ or\ evts/s}] \tag{4.15}$$

To measure the performance of a single component i, the average processing time[1] per event and thread

$$\langle t_i \rangle_{\mathrm{ref}} = \frac{t_i}{N} \tag{4.16}$$

is used.

4.4.3 Performance Trade-offs

A typical trade-off encountered when optimising track reconstruction algorithms is the possibility of sacrificing low-momentum track reconstruction efficiency for improving track processing speed, *e.g.* as is done in HLT1. For HLT2, this is not a useful trade-off because tracks at the lower end of the momentum spectrum are potentially interesting for physics analysis. Reconstructing low-momentum tracks, however, leads to a drop in event throughput as more tracks are processed and increases the fake track rate, as explained in Section 5.3.1. A higher fake track rate consequently increases the track fit's workload and can negatively impact the signal-to-background ratio in physics analysis.

Ultimately, the successful but work-intensive way to deal with these trade-offs is to optimise the throughput using techniques from the second category described in Section 3.3, under which the physics performance is invariant. If the reconstruction surpasses its computational performance goals, event throughput can be sacrificed to increase the track-finding efficiency and lower the fake track rate.

4.5 Track Reconstruction Sequences in HLT2

This section overviews the track reconstruction components and sequences developed for LHCb in Run 3 and shows key performance results.

4.5.1 Components and Baseline Track Reconstruction

The basic HLT2 track reconstruction sequence, illustrated in Figure 4.3, mostly follows what was used by LHCb during Run 2 of the LHC [83] and has been used for the Run 3 commissioning phase[2] in 2022. The individual components have been redesigned and optimised for Run 3. A new scheduler [63] determines the exact order of the

[1] The time is measured centrally by the HLT control flow manager using the C++ standard library's `chrono` header.

[2] The UT detector did not make it in time for the start of Run 3. The sequence was therefore run without the VeloUT tracking.

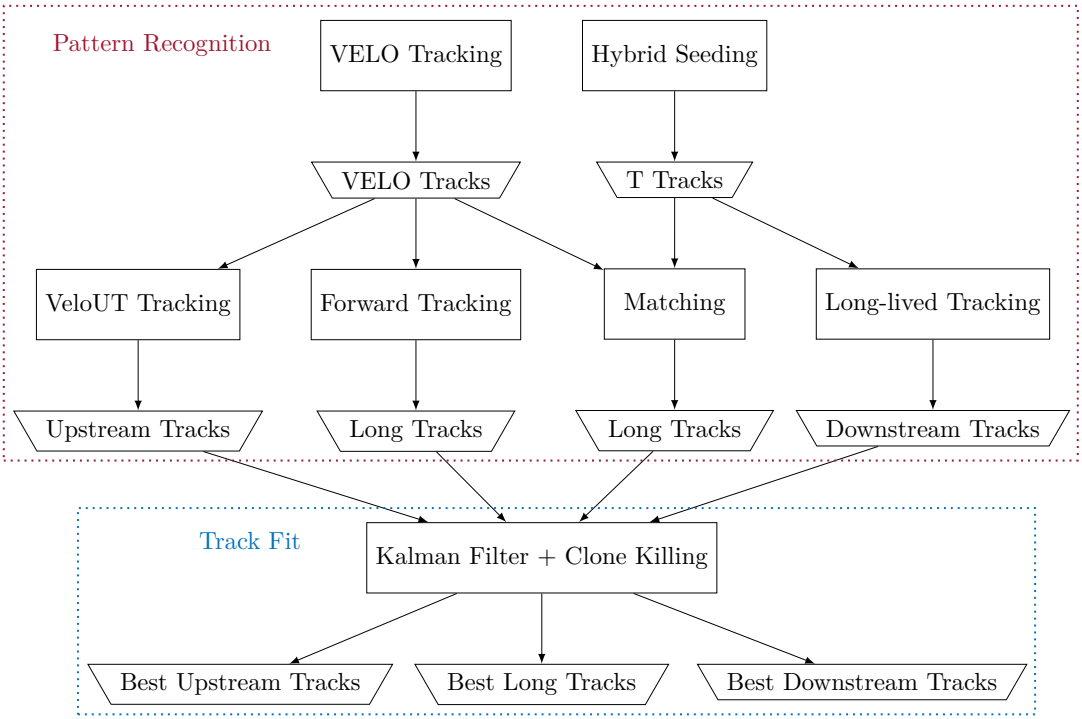


Figure 4.3: Data flow in the baseline reconstruction sequence. Algorithm components are shown in rectangular nodes; track containers are depicted in trapezoidal shapes. Detector hits, which are also inputs to the algorithm, are not shown.

components' execution, which guarantees that the input to a component is available before its execution using the thread-safe Gaudi functional-algorithm templates (see Section 3.3). Track reconstruction algorithms usually take other track types as input but also the detector hits, which can be prepared by separate components not shown in Figure 4.3. The hit preparation typically converts the signal clusters recorded by the tracking detector from readout-channel information to hit positions and properties in the global coordinate system and thus takes not much processing time.

The **VELO track reconstruction** algorithm is usually the first tracking component executed. It reconstructs straight lines from the 3-dimensional measurements provided by the VELO's pixel sensors and applies a simplified Kalman filter to improve the parameters of the track state vector (Equation 4.8) close to the beam. The Kalman filter considers multiple scattering using a parameterised noise matrix [48] for the pixel sensors and the RF foil. Since no momentum measurement using the VELO detector is possible, the Kalman filter uses noise parameters calculated for an average of the momentum spectrum. The VELO track reconstruction is optimised for computational performance using SIMD instructions. Details can be found in Ref. [84].

The **T track reconstruction**, better known as *Hybrid Seeding*, reconstructs tracks

using only information from the SciFi detector stations. It first builds track seeds from hits in two x layers and subsequently tries to form a triplet by adding a hit from an x layer in between. Then, x hits from so far unused layers are added using a third-order polynomial as the T-track model, followed by a search for matching stereo-layer hits. The magnet's fringe field reaching into the SciFi stations allows the momentum of T tracks to be estimated with a resolution of about 25%-35%, enabling the reconstruction of particles that decayed after the UT detector [81]. Details on the Hybrid Seeding algorithm can be found in Ref. [85].

After the successful execution of the two components mentioned above, the core part of the pattern recognition is run on their outputs.

The **Downstream track reconstruction**, also called *Long-lived tracking*, extrapolates T tracks through the magnet to the UT detector using a parametric model of the expected trajectory for a track with the momentum estimated by the Hybrid Seeding. UT hits close to the extrapolated positions are added to the T track to form the Downstream track. These tracks are indispensable for the physics programme of LHCb because they provide the bulk of statistics for studying decays involving long-lived hadrons such as K^0_S and Λ. The Run 3 algorithm is heavily based on its Run 2 predecessor, detailed in Ref. [86].

The components that find Long tracks are even more important for the physics output of LHCb. The **Long track reconstruction** consists of two algorithms, the *Matching* and the *Forward Tracking*, both aiming to find as many Long tracks as possible. This leads to a large overlap between the two sets of found Long tracks, which is beneficial for the Long track reconstruction efficiency [83] but is costly from a computational performance point of view. The redundancy can be resolved to obtain a fast reconstruction sequence as explained in Section 4.5.2. The Forward Tracking algorithm is described briefly in Ref. [87] and in great detail in Chapter 5. A comparison to the Matching algorithm is given in Section 5.4.1.

The **Upstream track reconstruction**, also called the *VeloUT tracking*, extends VELO tracks using hits in the UT detector. The VELO track is linearly extrapolated to a reference surface in the UT detector, and hits within a tolerance around the extrapolated position are collected from each layer. The UT hits are then clustered by forming doublets and trying to complete these into triplets and quadruplets by linearly extrapolating to the next layer, checking for matching hits. If at least three VELO-track-compatible UT hits are found, the track parameters are estimated using a simple least-square fit. If the fit quality is good enough, at most one Upstream track per VELO track is created. The achieved momentum resolution in the fringe field within the UT detector is around 15% in the mid and lower momentum range. More details on the algorithm can be found in Ref. [82]. Upstream tracks are scarcely[1] used for physics analysis. Their primary purpose in Run 3 was to improve the computational

[1]There is particular interest in using them for analyses including electrons, as 11% of them are wiped out of LHCb's acceptance by the magnet because they have low momentum or lost a significant part of their energy by bremsstrahlung [75].

performance of the Forward tracking in a CPU-based HLT1. They would have replaced the VELO tracks as input to the Forward tracking, allowing it to use the Upstream track's momentum estimate to narrow the search range for Long tracks and significantly speed up their reconstruction at the cost of a lower track finding efficiency because of the acceptance of the UT detector and inefficiencies of the VeloUT algorithm. More on this approach in the context of a CPU-based HLT1 can be found in Ref. [13]. The computational performance of the VeloUT algorithm itself was optimised by using SIMD instructions to perform extrapolations and fits in parallel for multiple VELO tracks. Of course, also in HLT2, the potential improvement of computational performance by using Upstream tracks as the input to the Forward tracking is of interest. The author of this thesis, therefore, implemented a *filter mode* for the VeloUT algorithm, which only requires that at least three UT hits are found within the tolerance around the extrapolated position. The filter mode accepts zero UT hits if the VELO track crosses the hole in the UT detector close to the beam pipe (see Figure 2.5). The result is a standard Upstream track if found UT hits can be clustered and successfully fitted. Otherwise, the track is accepted without a momentum estimate. The Forward tracking is designed to handle this kind of input as explained in Section 5.2. Note that in filter mode, the resulting Upstream tracks do not contain the found UT hits because these are not a requirement to form a Long track. Also, the Long track reconstruction algorithms use a tool to add UT hits to a Long track (see Section 5.2.9), which achieves a higher UT hit efficiency because of the more precise momentum estimate of Long tracks. Upstream tracks created by the VeloUT algorithm in filter mode are sometimes referred to as UT-filtered VELO tracks.

The pattern recognition algorithms only perform least-square fits to estimate track parameters for the track segments they find. The final track-state vectors and their covariance matrices are obtained by applying a **Kalman filter** [88, 89]. The Kalman filter predicts the track parameters at a z position of interest based on known track parameters at a reference position. The propagation through the magnetic field is done by a linear, parabolic or fifth-order Runge-Kutta method, depending on the step size and the strength of the magnetic field at the z positions [90]. For Run 3, LHCb's Kalman Filter algorithm has been re-implemented with contributions from the author of this thesis. The main objective was to improve its computational performance, as it was by far the most time-consuming component of the event reconstruction. This was achieved by tailoring the algorithm to the data structures returned by the pattern recognition algorithms, optimising the code used for track propagation in the magnetic field and employing parameterisations of the noise matrix [48] as well as the energy loss correction. For simplicity, the Kalman filtering and clone track removal are depicted as a single component in Figure 4.3. In reality, the basic track reconstruction sequence first fits the Long tracks found by the Forward tracking, then removes the overlap (*clones*) between these and the Long tracks found by the Matching, and then fits the remaining Long tracks, yielding the selection of the best Long tracks. By definition of

the track types, a Downstream track might be a segment of a Long track. Therefore, the overlap between them and the best Long tracks is removed before fitting the Downstream tracks. The Upstream tracks are processed analogeously.

The reconstruction efficiencies of Long tracks in the basic sequence are shown in Table 4.1. Tracks from a B-meson decay or with a momentum above 5 GeV/c are reconstructed with a likelihood of more than 90%. Electrons only reach 80% reconstruction efficiency as they emit bremsstrahlung when traversing the detector material (see Section 4.2.1) and are therefore harder to find. The fraction of clones is negligible at below 0.3% in all categories and is thus not further discussed here. The computational performance is shown in Figure 4.5a. Including the particle identification algorithms, a throughput of 388 events per second and computing node is reached. The track reconstruction takes roughly 60% of the computing time in this scenario.

4.5.2 Fast Track Reconstruction

The basic reconstruction sequence shown in Figure 4.3 is not optimal from a computational performance perspective. It does redundant work by finding almost the same set of Long tracks twice, using the Matching and the Forward tracking. However, the redundancy is easy to resolve by the sequence shown in Figure 4.4, which is the one currently foreseen for nominal data taking during Run 3. In this fast track reconstruction sequence, the Matching algorithm is run first, pairing VELO tracks with T tracks to form the first set of Long tracks. Subsequently, only the unmatched, residual VELO tracks are passed on to the Forward tracking. In addition, also the hits in the SciFi detector that are already used by the first set of Long tracks are removed, such that the Forward tracking runs over a reduced set of hits as well.

Similarly, the Long-lived tracking only runs on T tracks that are not a segment of a Long track in the first set and UT hits which are not part of the Long tracks. As shown in Table 4.1, this leads to a drop in reconstruction efficiency of about 1% for Long tracks. The decrease in efficiency is acceptable given that the event throughput is increased by 28% with the fraction of time spent on the track reconstruction below 50% and reaching the goal of processing roughly 500 events per second and node as shown in Figure 4.5b. More details on the track reconstruction performance can be found in Refs. [15, 91].

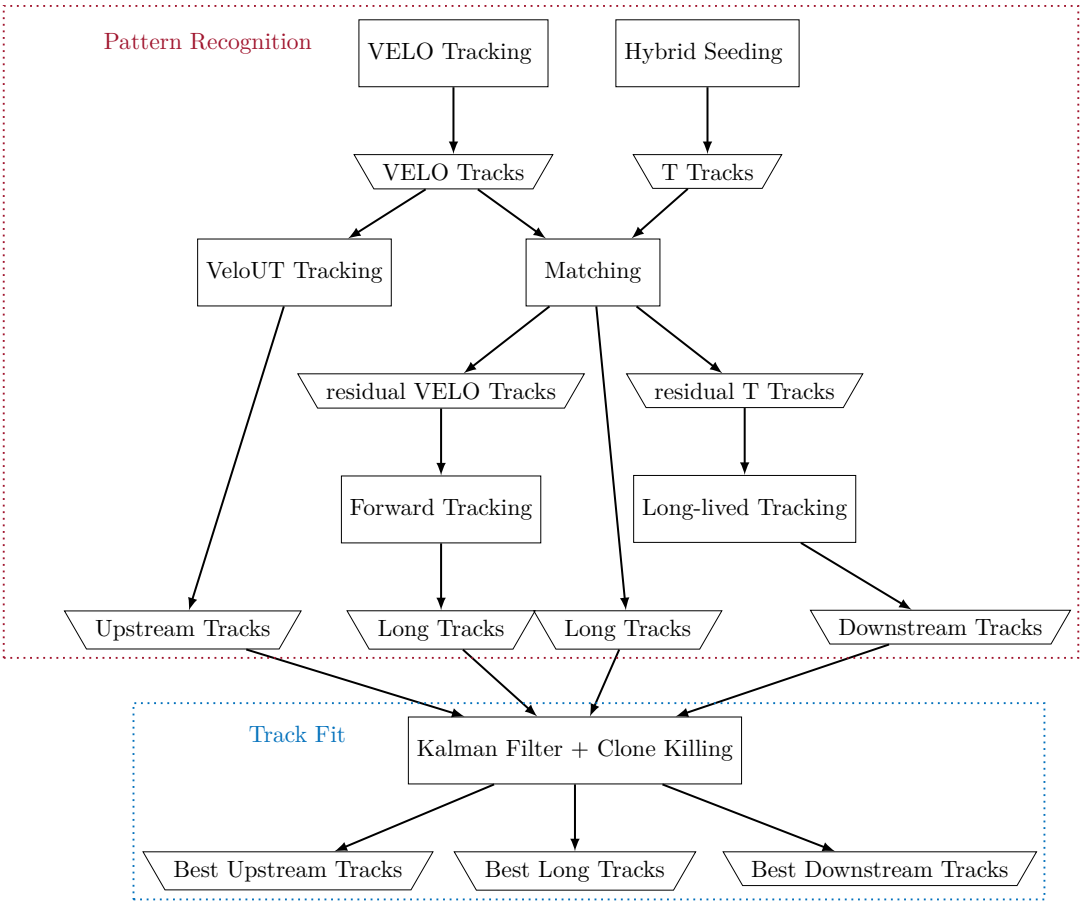
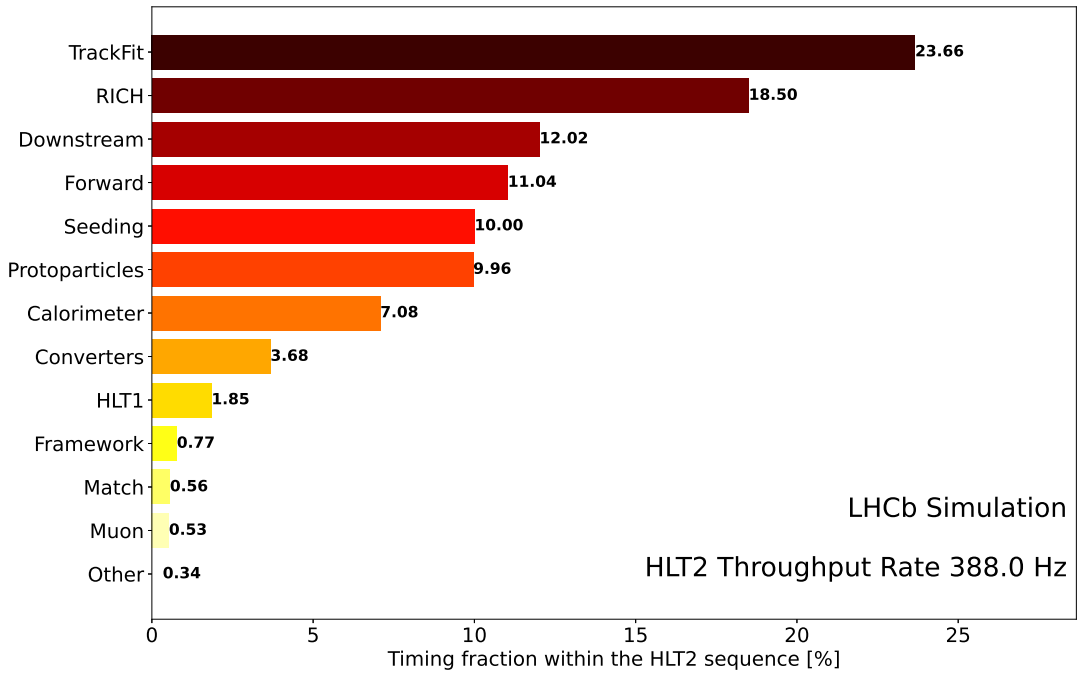


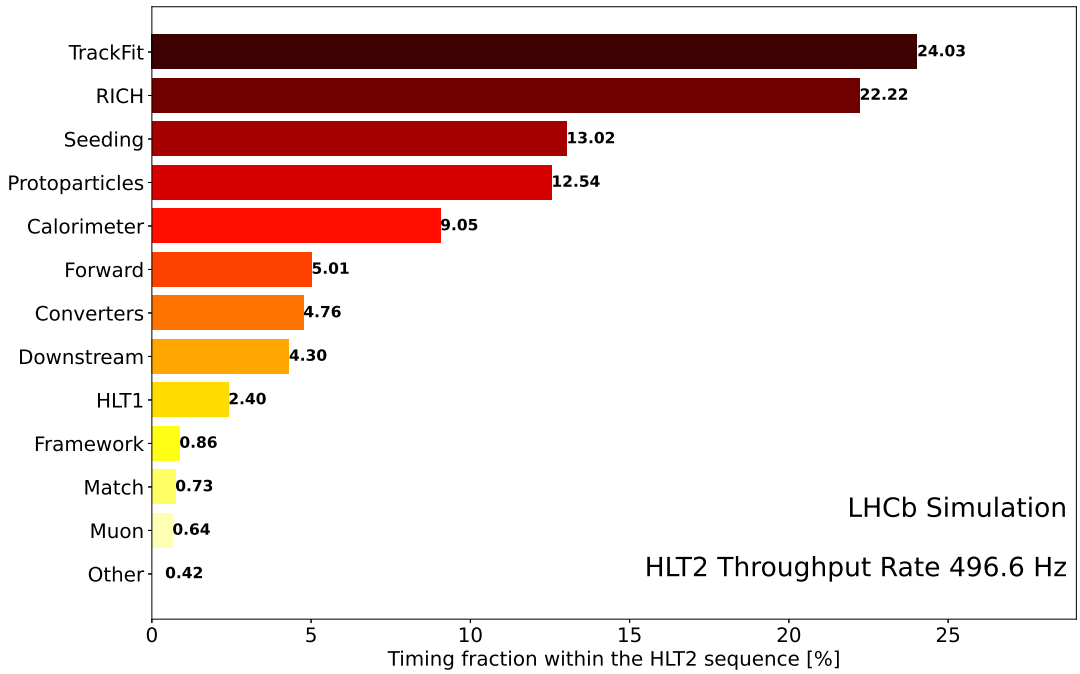
Figure 4.4: Data flow in the fast reconstruction sequence. Algorithm components are shown in rectangular nodes; track containers are depicted in trapezoidal shapes. Detector hits, which are also inputs to the algorithm, are not shown.

Table 4.1: Long track reconstruction efficiencies of the baseline and fast sequence for different efficiency categories evaluated on simulation.

	Baseline Sequence [%]	Fast Sequence [%]
Ghost rate	11.64 ± 0.04	10.27 ± 0.04
$\varepsilon_{\text{reco}}$		
Long	89.02 ± 0.04	87.60 ± 0.05
Long $p > 5\,\text{GeV}/c$	92.61 ± 0.05	91.60 ± 0.05
Long from B	92.0 ± 0.2	91.0 ± 0.2
Long $p > 5\,\text{GeV}/c$ from B	94.7 ± 0.2	93.9 ± 0.2
Long $e^{\pm}$ from B	80.5 ± 0.5	77.4 ± 0.5



a: Baseline reconstruction sequence.



b: Fast reconstruction sequence.

Figure 4.5: Breakdown of the HLT2 reconstruction event throughput. The numbers right of the bars are given in percent. The Kalman filter is called track fit here.

5 Forward Tracking in HLT2

This chapter describes the HLT2 Forward tracking algorithm, related components and data structures developed and implemented by the author of this thesis. The intention is to document the algorithm used in LHCb's software trigger during Run 3 of the LHC so that future development has an extensive reference. Documentation for the Forward tracking algorithms in Runs 1 and 2, and the first developments made for Run 3, can be found in Refs. [92–94]. A description of a CPU-HLT1 Forward tracking is given in Ref. [13]. The GPU-HLT1 approach is summarised in Ref. [42].

5.1 Objective and Requirements

The goal of the Forward tracking is to find a forward extension of a given VELO track in the SciFi tracker and to estimate an initial momentum of this Long track [87]. If tracks from the VeloUT algorithm are available, the Forward tracking should be able to use the additional, coarse momentum estimate and find a set of SciFi hits belonging to these.

The specification of the SciFi tracker aims at 99% single-hit efficiency. Including the 1.7% dead regions per layer (see Section 2.4.3), the probability of detecting a single hit is around 97.3%; hence the number of SciFi hits on a Long track is expected to be twelve only for 72% of the tracks. The Forward tracking, therefore, aims to reconstruct the 99.6% tracks with ten or more SciFi hits. Furthermore, the objective is to achieve a purity of $f_{\text{pure}} > 99\%$ and a hit efficiency of $\epsilon_{\text{hit}} > 98\%$ (see Section 4.4.1), where having a pure track is more important than finding all possible hits. This is due to the subsequent Kalman filter component, which must expensively remove outlier hits. Regarding the track finding efficiency and fake track fraction, the obvious goal is to have the former as high as possible while keeping the latter as low as possible. More quantitatively, the objective is to meet at least the estimates made when the LHCb upgrade was planned, shown in Table 5.1. There is no strict requirement on the computational performance of individual trigger components. The whole HLT2 application, however, has to be able to process the 1 MHz output rate of HLT1 and thus needs a certain amount of computing nodes, as explained in Section 3.2.2. In June 2019, the reconstruction sequence comparable to the baseline sequence (Figure 4.3) had a throughput of 80 evts/s on the reference node[2]. The Forward tracking, which

[2]The best-performing platform in June 2019 was `x86_64+avx2+fma-centos7-gcc8-opt`, *i.e.* Advanced-Vector-Extension 2. Fuse-multiply-add optimisation enabled with GCC8.1 on the O3 optimisation level.

Table 5.1: Forward tracking's ghost rate and track finding efficiency as estimated in Ref. [19] for the Run 3 upgrade and Run 1 for comparison. On average, Run 1 had one primary vertex per event, while Run 3 has five.

	Upgrade TDR [%]	LHCb Run 1 [%]
Ghost rate	38.2	25.4
ε		
Long	85.2	91.9
Long $p > 5\,\mathrm{GeV}/c$	92.3	96.1
Long from B	91.1	94.8
Long $p > 5\,\mathrm{GeV}/c$ from B	94.7	96.8

took about 8% of this sequence, had a throughput of 1000 evts/s. At that time, the usual assumption was that 1000 reference-node equivalents would be available for HLT2 processing in Run 3. Hence, the Forward tracking alone would have consumed all available resources to process the output of HLT1 and thus had to gain several factors in throughput without compromising the physics performance.

5.2 Algorithm Design and Description

The central problem is finding the single correct combination of a VELO track with SciFi hits among the many possible extensions or finding that no such combination is present. The complexity scales with the number of input tracks and the total number of hits the SciFi tracker recorded. Typical Run 3 events with inelastic pp collisions at LHCb lead to $\mathcal{O}(10^2)$ VELO tracks and $\mathcal{O}(10^3)$ SciFi hits. In simulated HLT1-filtered minimum-bias samples, only about 50% of the VELO tracks are part of a trajectory through the whole tracking system, and most SciFi hits are left by secondary particles which do not have a VELO-track segment. Therefore, the immediate design goal of the Forward tracking is to provide a procedure that quickly decides whether a VELO track can have a reasonable extension in the SciFi tracker. If it is not possible to reject a VELO track early in the algorithm, at least the number of SciFi-hit sets considered as extensions to the VELO tack should be reduced fast to control the complexity. Hence, the initial combinatorics reduction is critical to performance and, therefore, heavily uses the full capabilities of modern CPUs described in Section 3.3.

5.2.1 Overview

Figure 5.1 illustrates the most important parts of the Forward tracking, which are explained in more detail in the following sections. The flow chart shows method

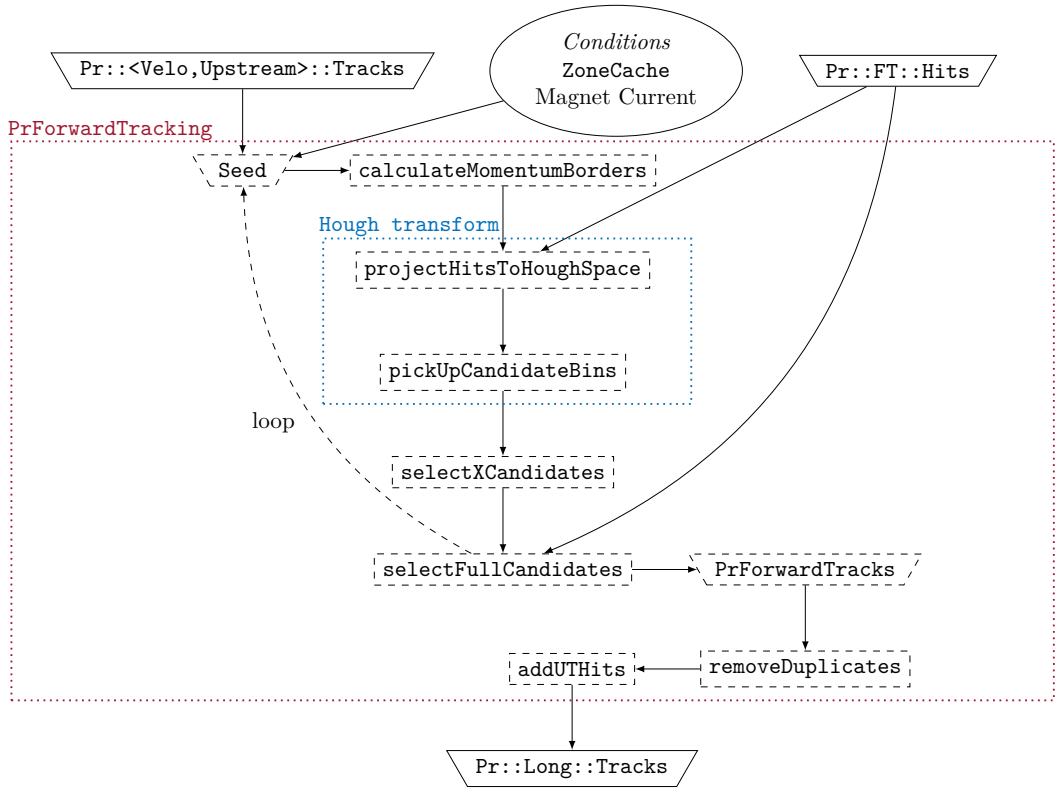


Figure 5.1: HLT2 Forward tracking algorithm overview. The top layer shows the inputs to the Forward tracking where the oval field contains inputs from the conditions database. The red dashed box contains the methods executed within the algorithm. The blue dotted box indicates methods performing the Hough-like transform. The bottom layer shows the output of the algorithm. The solid black arrows hint at which methods use the respective inputs. The dashed black arrow indicates a loop over the input tracks.

names also used in the algorithm's implementation such that the detail-interested reader can navigate the code more easily. The sections contain documentation of the parameterisations used to model certain trajectory parameters. It is not necessary to understand the exact polynomial expressions as they are not unique (see also Appendix A.3); they are only given for completeness.

Algorithm Concept The Forward tracking takes a track found by the VELO or VeloUT tracking algorithm as input seed. The seed contains information about the trajectory upstream of the magnet, which is used to set limits on the hit-search region within the SciFi tracker downstream of the magnet (`calculateMomentumBorders`). Hits within the search window are projected to a reference plane according to the trajectory expected from the input track (`projectHitsToHoughSpace`). This is done for SciFi hits from both x and stereo layers. The reference plane is divided into bins that count the number of SciFi layers from which the projected hits originate (see

Figure 5.8 for illustration). Only the projected position of x-layer hits is kept for later use. The bins are scanned for high numbers of accumulated SciFi layers, indicating a promising candidate set of SciFi hits (`pickUpCandidateBins`). Subsequently, the stored x-layer projection from candidate bins and their neighbours are examined to form one or several initial track candidates by clustering the projected x-layer hits. The xz trajectory of the track candidates within the SciFi tracker is determined, and low-quality tracks are rejected (`selectXCandidates`). The xz trajectory is then used to select the correct stereo-layer hits which complete the Long-track candidate (`selectFullCandidates`). The procedure is repeated for each input track. After all Long-track candidates have been found, duplicates are removed (`removeDuplicates`) and matching UT hits are added to each Long track (`addUTHits`).

5.2.2 Inputs

The algorithm has three inputs: detector conditions such as the magnet current, VELO or Upstream tracks and SciFi hits.

The SciFi hit container is prepared by the `PrStoreScifiHits` component that uses already clustered detector channels (see Section 2.4.3) to store their SciFi layer index (*plane code*), and the x position at $y = 0$ within LHCb's coordinate system. These two quantities are vital for the first, most performance-critical part of the Forward tracking and therefore stored in an SoA format as shown in Figure 5.2. The twelve SciFi planes are divided into upper and lower halves, defining 24 zones shown in Figure 5.3. The hits are stored in ascending order of their x positions in each zone. For easy access to the hits in each zone, their start and end indices are also stored in the hit container. Each zone end is protected by a sentinel hit with an unphysically large x position, simplifying navigation in the container, as will become clear later. Additional

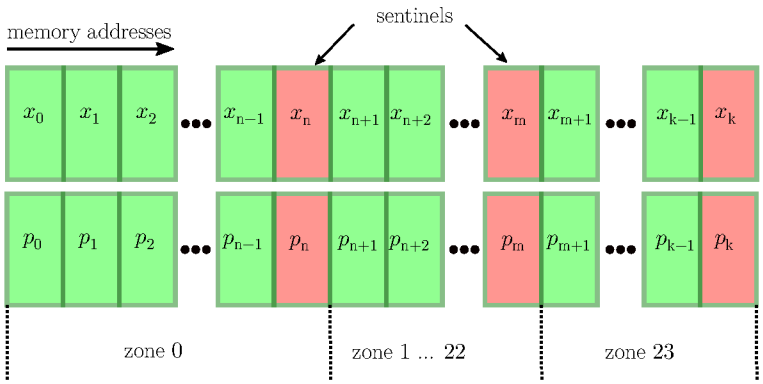


Figure 5.2: Storage layout of SciFi hits x positions and layer number p structured into zones, each of which ends with a sentinel entry.

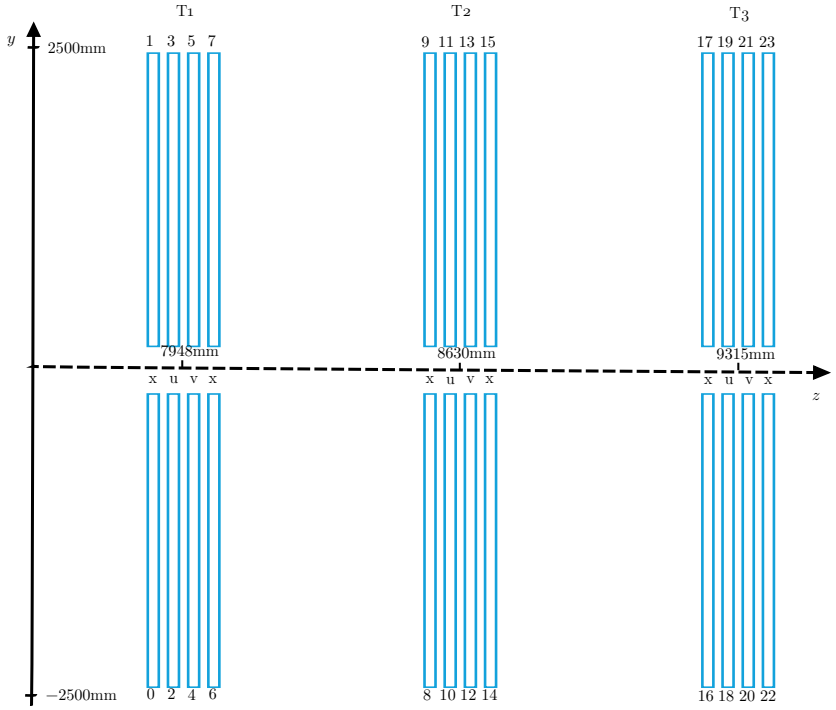


Figure 5.3: Overview of the zone numbering for each SciFi tracker layer and their positions within the global LHCb coordinate system.

information about the hits is split into a hot and a cold part[1] and stored in an AoS layout to maximise spatial cache-locality for this particular data (see Section 3.3). The hot part contains the reciprocal of the variance of the measurement determined by the size of the cluster (see Section 2.4.3), *i.e.* the hit weight, the stereo angle $\mathrm{d}x/\mathrm{d}y$, tilting $\mathrm{d}z/\mathrm{d}y$ and z position of the SciFi mat. The cold part stores information about the y endpoints of the SciFi detector at the position of the hits and a unique identifier for the central detector channel used to build the cluster. To avoid dynamic memory (de)allocations, each field in the hit container uses LHCb's event-local memory pool. The SciFi hits' positions are derived from the detector conditions and thus include the best available alignment of the tracking stations.

Furthermore, the Forward tracking algorithm reads the sign of the magnet current from the conditions database. For the computational performance, it is helpful to keep geometric zone information derived from alignable objects in a cache (`ZoneCache`) that gets updated in case the detector conditions change (see Section 3.2.3).

The input tracks for the Forward tracking are produced either by the VELO tracking or the VeloUT algorithm. They are already available in an SoA data structure, which, however, in the current implementation of the Forward tracking, does not lead to computational performance benefits as the loop over the input tracks is scalar and

[1] *Hot* and *cold* here refer to the frequency with which the CPU accesses the corresponding memory.

encloses most of the algorithm as shown in Figure 5.1. Thus, the first step in the input-track loop is to transform the input track into a more useful data structure called the *seed*, which contains track-state-vector parameters at the end of the VELO detector ($z_\mathrm{v} = 770\,\mathrm{mm}$) and pre-computes useful quantities like the Jacobian (Equation A.2) already encountered in Equation 4.5, polynomial terms used in extrapolations and reference positions using linear extrapolation of the VELO track state.

5.2.3 Estimating Trajectory Boundaries

After the initialisation of the seed, the next step is to choose the SciFi hits that are considered for an extension. A precise choice is beneficial for the computational performance of the algorithm because it reduces the number of hits that must be processed in the subsequent steps. The first property of the trajectories to notice is their approximate straightness in the y-z plane due to the negligible magnitude of the magnetic field's B_x component and $t_x, t_y \ll 1$. Only 0.02%[1] of reconstructible Long tracks traverse both the upper and lower halves of the SciFi tracker. Thus, the search for extending hits is restricted to only one half of the detector, corresponding to even or odd zone numbers. While in the y direction, there is nothing more to gain because the scintillating fibres run mainly along the y axis, boundaries of the trajectory in the x direction can be found if a minimum momentum is assumed. The track state vector of the seed is then given by $(x_\mathrm{v}, y_\mathrm{v}, t_x, t_y, \pm q/p_\mathrm{min})$ at the end of the VELO z_v, with a sign ambiguity because of the unknown charge of the seed particle.[2] Using this state vector and the magnetic field map, the two xz trajectories enveloping the SciFi area interesting for reconstruction can be determined by numerically solving the equations of motion from Section 4.2.2, which, however, is computationally not cheap. Instead, a parameterisation is used to estimate the difference between the straight-line extrapolation and the position of the trajectory $x_\mathrm{ref}^\mathrm{extp}$ at a reference plane:

$$\Delta x_\mathrm{border} := |x_\mathrm{ref}^\mathrm{straight} - x_\mathrm{ref}^\mathrm{extp}|$$

with

$$x_\mathrm{ref}^\mathrm{straight} := x_\mathrm{v} + t_x(z_\mathrm{ref} - z_\mathrm{v})$$

at $z_\mathrm{ref} = 8520\,\mathrm{mm}$ and parameterised by

$$\Delta x_\mathrm{border}^\mathrm{min} = \frac{1}{p_\mathrm{min}}\left[L + \frac{1}{p_\mathrm{min}}\left(Q + \frac{1}{p_\mathrm{min}}\left(C + c_8\frac{1}{p_\mathrm{min}}\right)\right)\right] \tag{5.1}$$

with

$$L = c_0 + c_1 t_y^2 + c_2 t_x^2 + q_\mathrm{mag} c_5 t_x^3 + c_9 t_x^2 t_y^2 + c_{12} t_x^4$$

[1] The magnetic field is only responsible for half of these; the rest crosses from one detector half to the other via scattering.

[2] Note that for simplicity, t_x and t_y are the slopes at z_v if not explicitly stated otherwise.

$$
\begin{aligned}
Q &= q_{\text{mag}}(c_3 t_x + c_{10} t_x t_y^2) + c_6 t_x^2 \\
C &= c_4 + q_{\text{mag}} c_7 t_x + c_{11} t_y^2 \\
q_{\text{mag}} &= \begin{cases} q \cdot r_I & \text{if } \frac{q}{p} \text{ known} \\ |r_I| & \text{else} \end{cases}
\end{aligned}
$$

where r_I is the signed relative current through the magnet; under nominal detector conditions, the relative current is $r_I = \pm 1$, with negative and positive signs defining the *Down* and *Up* magnet polarities, respectively. The $|r_I| \neq 1$ case has not been studied and is thus not guaranteed to be covered by the parameterisations. The sign of q_{mag} determines the sign of the track curvature; a negative q_{mag} deflects the particle to the left, *i.e.* in the positive x direction. The coefficients c_i are determined using simulated tracks and given in Table A.1. A general method to obtain these parameterisations is described in Appendix A.3. The default configuration of the Forward tracking uses $p_{\text{min}} = 1500\,\text{MeV}/c$ and $p_{\text{T,min}} = 50\,\text{MeV}/c$; below these values, most tracks are not reconstructible anymore because they are deflected out of acceptance by the magnet, or the particle decays before the SciFi tracker [75].

The minimum momentum requirement can be configured for different physics use cases. An example is the ion-ion collision programme in which it makes sense to restrict the reconstruction to higher momenta to cope with the high track multiplicity.

The minimum p_{T} value is projected to a minimum p value, using the known track slopes t_x and t_y. The smaller $\Delta x^{\text{min}}_{\text{border}}$ calculated from both momenta is eventually used to define the border of the trajectory, also called the *search window*. At the reference plane, it is given by

$$
\begin{aligned}
x_{\text{min}} &= x^{\text{straight}}_{\text{ref}} - \Delta x^{\text{min}}_{\text{border}} \\
x_{\text{max}} &= x^{\text{straight}}_{\text{ref}} + \Delta x^{\text{min}}_{\text{border}}
\end{aligned}
$$

There is no profound reason to define the search window on the reference plane initially; it simply is convenient as this reference plane is used in the context of other parameterisations too. To select the hits, the search window must be adjusted to the z position of each SciFi layer (see Section 5.2.5). The search window efficiency is shown in Figure 5.4. The turn-on-curve shape of the efficiency is due to multiple scattering, which smears the momentum border; the efficiency difference between the first and the last SciFi layer is an artefact of the simplistic adjustment to account for the z position of the layer. Because HLT2 aims at reconstructing the whole event, including low-momentum particles, the search windows can be significant to the extent that they cover the whole SciFi tracker and thus collect all hits of one half. Figure 5.5a gives two examples of how these search windows look.

The only way to avoid extensive search windows is to either increase the minimum-momentum requirement, *i.e.* limiting the physics reach of the experiment or to use an initial estimate of charge and momentum. The latter is also supported by Equation 5.1,

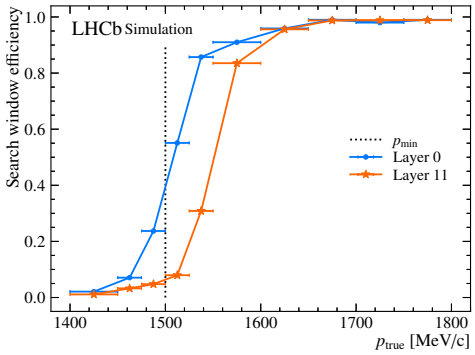


Figure 5.4: The search window efficiency is defined as the sum of all SciFi hits within the search window truth-matched to reconstructible Long tracks, divided by all true hits from the reconstructible Long tracks. The blue circles and orange stars show the search window efficiency for the first and last SciFi layer, respectively, in bins of true momentum with $p_{\text{min}} = 1500\,\text{MeV}/c$.

e.g. using Upstream tracks, in which case it returns an estimate for the actual deviation from the straight-line trajectory at the reference plane. The search window size is then driven by the momentum resolution of the input tracks and thus becomes asymmetric because $\Delta x_{\text{border}} \propto 1/p$. This has been studied in Ref. [13], the result of which is used to define the momentum-dependent search window as:

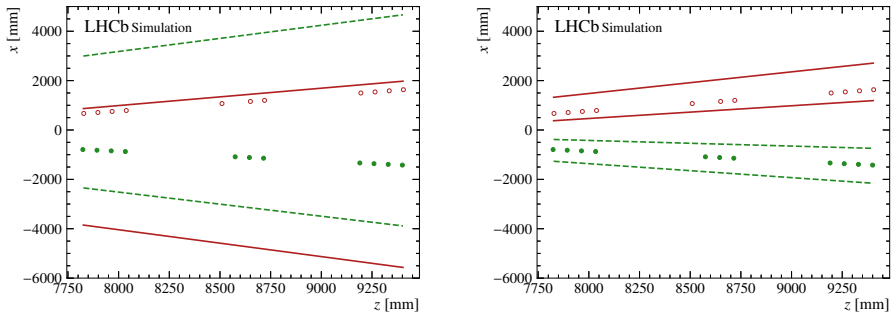
$$x_{\text{min}} = \begin{cases} x_{\text{ref}}^{\text{straight}} - \Delta x_{\text{border}} - \Delta_{\text{lower}} & \text{if } q_{\text{mag}} > 0 \\ x_{\text{ref}}^{\text{straight}} + \max(\Delta x_{\text{border}} - \Delta_{\text{higher}}, 0) & \text{if } q_{\text{mag}} < 0 \end{cases}$$

$$x_{\text{max}} = \begin{cases} x_{\text{ref}}^{\text{straight}} + \min(-\Delta x_{\text{border}} + \Delta_{\text{higher}}, 0) & \text{if } q_{\text{mag}} > 0 \\ x_{\text{ref}}^{\text{straight}} + \Delta x_{\text{border}} + \Delta_{\text{lower}} & \text{if } q_{\text{mag}} < 0 \end{cases}$$

with

$$\Delta_{\text{lower}} = \max(\delta_{\text{min}}, \min(\delta_{\text{lower}} + b_\delta/p, \delta_{\text{max}}))$$
$$\Delta_{\text{higher}} = \min(\delta_{\text{higher}} + a_\delta/p, \delta_{\text{max}})$$

The δ values are momentum-resolution dependent. For Upstream tracks from the VeloUT tracking with a resolution of roughly 15%, the parameters are $\delta_{\text{min}} = 150\,\text{mm}$, $\delta_{\text{lower}} = 100\,\text{mm}$, $b_\delta = 2800\,\text{mm}\,\text{GeV/c}$, $\delta_{\text{higher}} = 50\,\text{mm}$, $a_\delta = 1400\,\text{mm}\,\text{GeV}/c$ and $\delta_{\text{max}} = 600\,\text{mm}$. The momentum-dependent search window is visualised in Figure 5.5b. More on this approach can be found in Section 5.5.



a: Hit search window using minimum-momentum boundary.

b: Hit search window using momentum-dependent boundary.

Figure 5.5: Visualisation of hit-search windows for minimum-momentum boundary at $p_{\text{min}} = 1500\,\text{MeV}/c$ (a) and momentum-dependent boundary using the true momentum of the simulated tracks (b). The red solid line and empty circles are the momentum borders and the SciFi hits, respectively, left by a π^+ with $p_{\text{true}} = 1508\,\text{MeV}/c$. The green dashed line and filled circles show the equivalent for a π^- with $p_{\text{true}} = 2999\,\text{MeV}/c$.

5.2.4 Simplified Track Model

Before discussing the next step in the algorithm, it is instructive to define a simplified track model following the Optical Model method [92] in which the magnetic field is treated as a thin lens refracting tracks like light rays. The goal is to identify a small set of parameters that effectively describe the state transfer from before the magnet to downstream of the magnet. This is useful because no measurements are made within the core part of the magnetic field, and the track finding is not interested in the exact path the particle took through the magnet. Furthermore, the model allows to perform linear projections within the SciFi tracker needed for the optimised Hough-like transform described in Section 5.2.5. The Optical Model is sketched in Figure 5.6, with the trajectory described by the input track state, the magnet kick position z_{mag} (optical centre), and the difference between the incoming and outgoing slopes Δt_x. Once a single hit $(x_{\text{hit}}, z_{\text{zone}})$[1] downstream of the magnet is taken into account, predicting the x coordinate at a given z position is a simple linear extrapolation within the model:

$$x_{\text{proj}}(z) = x_{\text{mag}} + \frac{x_{\text{hit}} - x_{\text{mag}}}{z_{\text{zone}} - z_{\text{mag}}}(z - z_{\text{mag}}) \tag{5.2}$$

with $x_{\text{mag}} = x_{\text{v}} + t_x(z_{\text{mag}} - z_{\text{v}})$. LHCb's magnetic field, however, fringes into the SciFi tracker (see Figure 2.3). Hence, the hits do not fall on a straight line, and for track finding, the Optical Model method has to be extended by a model for the trajectory within the SciFi tracker. The x projection of the trajectory within the SciFi stations is

[1]The z position of the hit is taken as equivalent to the global z position of the detector layer. This is an approximation reducing the necessary memory accesses as z_{zone} can be taken from the `ZoneCache`.

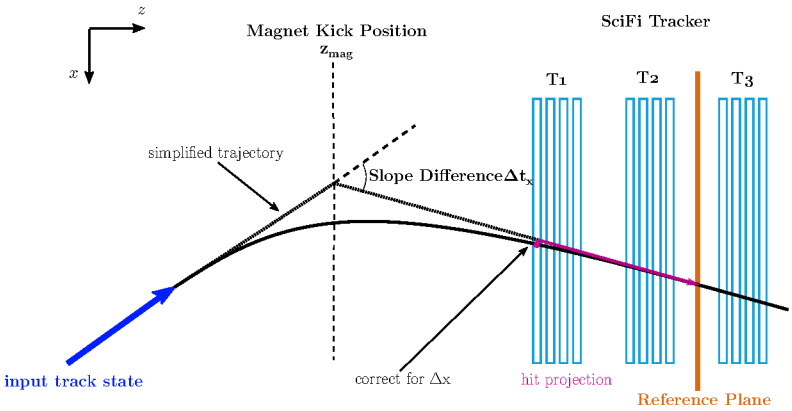


Figure 5.6: Illustration of the Optical Model method to effectively describe a trajectory through the magnet. Using the model, measured hit positions can be projected to the reference as indicated by the magnet arrow. Note that the reference plane is drawn between T2 and T3 only for illustration purposes.

described by a third-order polynomial

$$x(z) = a_x + b_x(z - z_{\text{ref}}) + c_x(z - z_{\text{ref}})^2 + d_x(z - z_{\text{ref}})^3 \tag{5.3}$$

where the higher-order monomials can be seen as the fringe-field correction to the Optical Model

$$\Delta x(z) := c_x(z - z_{\text{ref}})^2 + d_x(z - z_{\text{ref}})^3 \tag{5.4}$$

The z position of the reference plane is set to $z_{\text{ref}} = 8520\,\text{mm}$ and defines the origin of the local coordinate system used to describe the tracks within the SciFi tracker. The choice is in principle arbitrary, yet a position within the SciFi detector gives the notion of a virtual detector layer such that the coefficients are easy to understand, *e.g.* a_x can be seen as a virtual hit on the reference plane.

The slope difference used in the Optical Model method is defined as

$$\Delta t_x := b_x - t_x \tag{5.5}$$

The optical centre of the magnet is defined as the intersection between the trajectory tangents before and after the magnet:

$$z_{\text{mag}} = \frac{x_{\text{v}} - t_x z_{\text{v}} - a_x + b_x z_{\text{ref}}}{\Delta t_x}$$

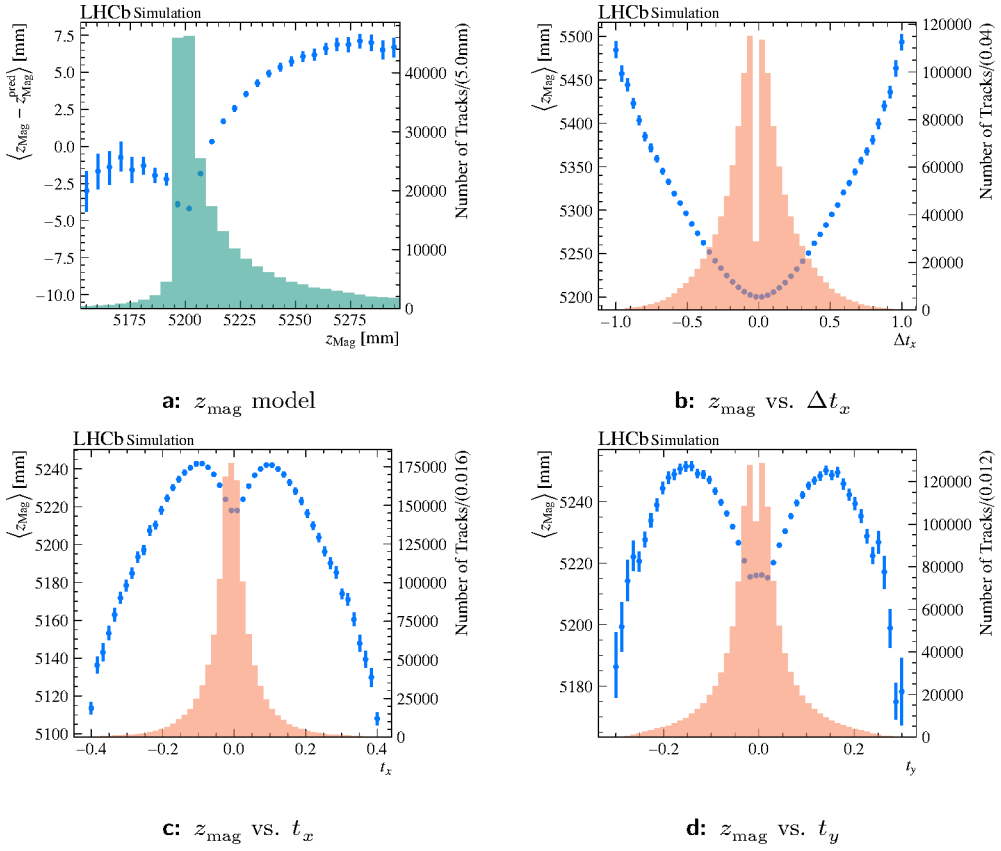
Both Δt_x and z_{mag} are a priori unknown for a given input track because they directly depend on the final track state that has yet to be found. The slope difference is not accessible as it is a proxy parameter for the momentum, $\Delta t_x \propto 1/p$, which is not directly inferrable from the initial track state. The magnet centre, however, is at least partly a property of the apparatus, constrained by the position of the magnetic field as

shown in the histogram in Figure 5.7a. Furthermore, Figures 5.7c and 5.7d show that t_x and t_y are correlated enough to z_{mag} that an initial estimate is possible using only this information from the input track. Thus the initial estimate called z'_{mag} is given by Equation 5.7 neglecting the last term, which is not accessible yet because of the cyclic dependency with Δt_x. Then, combining z'_{mag} with a single measurement $(x_{\text{hit}}, z_{\text{zone}})$ in the SciFi tracker gives a first estimate of Δt_x by

$$\begin{aligned}\Delta t'_x &= \frac{x_{\text{hit}} - x_{\text{mag}}}{z_{\text{zone}} - z'_{\text{mag}}} - t_x \\ &= \frac{x_{\text{hit}} - (x_{\text{v}} + t_x(z_{\text{zone}} - z_{\text{v}}))}{z_{\text{zone}} - z'_{\text{mag}}}\end{aligned} \tag{5.6}$$

The correlation between z_{mag} and Δt_x, shown in Figure 5.7b, is then used to improve the z_{mag} estimate by

$$z_{\text{mag}} = c_0 + c_1 t_x^2 + c_3 t_y^2 + \Delta t'_x(c_2 t_x + c_4 \Delta t'_x) \tag{5.7}$$



a: z_{mag} model **b:** z_{mag} vs. Δt_x

c: z_{mag} vs. t_x **d:** z_{mag} vs. t_y

Figure 5.7: Regression plots for the z_{mag} model and the variables it depends on.

with the coefficients listed in Table A.2. In principle, this procedure could be repeated to refine the respective estimates further. In practice, calculating z_{mag} once using $\Delta t'_x$ is sufficient for the precision needed for the pattern recognition in the next section. The residuals achieved by Equation 5.7 are shown in Figure 5.7a. They indicate that the calculated z_{mag} is slightly biased depending on the true magnet centre position. Yet, because z_{mag} is still determined with a precision up to a few millimetres, the bias picked up by x_{proj} from Equation 5.2 is much less than a millimetre and thus negligible given that the single-hit resolution is of the same order.

In principle, the same model can be defined for the y projection of the trajectory. However, the minor bending of the track in the $y - z$ plane and the accordingly small Δt_y, combined with the lack of precise y-coordinate measurements in the SciFi tracker, make the Optical Model method impractical in this case.

5.2.5 Optimised Hough-like Transform

This part of the algorithm performs the actual pattern recognition; it takes care of the bulk of the combinatorics reduction and is the most time-consuming within the algorithm. It, therefore, needs to be implemented with computational performance in mind. The pattern recognition is inspired by the Hough transform [95] and follows a map-reduce strategy to sieve out sets of SciFi hits that do not form a matching extension to the input track. The x positions of SciFi hits selected by the search window are mapped to the reference plane using the simplified track model. They are then filled into a histogram (`projectHitsToHoughSpace` in Figure 5.1). The histogram counts the number of unique SciFi detector layers present among the hits in one bin as the number of layers is the best discriminant against random combinations. This way, hits that match the input track accumulate layers in a few bins as depicted in Figure 5.8. Subsequently, the histogram is scanned for small groups of neighbouring bins exceeding a layer-count threshold, thus reducing the large set of hits from within the search window to none, one or several small sets of hits, which become candidates for the extension of the input track (`pickUpCandidateBins` in Figure 5.1).

Hit selection First, for each zone, the SciFi hits within the search window must be found. Equation 5.1 only gives the momentum borders at the reference plane and thus has to be scaled to the individual SciFi layers. A simple linear scaling

$$\begin{aligned} x_{\text{min}}(z_{\text{zone}}) &= x_{\text{zone}}^{\text{straight}} - s \cdot \Delta x_{\text{border}}^{\text{min}} \\ x_{\text{max}}(z_{\text{zone}}) &= x_{\text{zone}}^{\text{straight}} + s \cdot \Delta x_{\text{border}}^{\text{min}} \end{aligned}$$

with

$$s := \frac{z_{\text{zone}} - z_{\text{mag}}}{z_{\text{ref}} - z_{\text{mag}}}$$

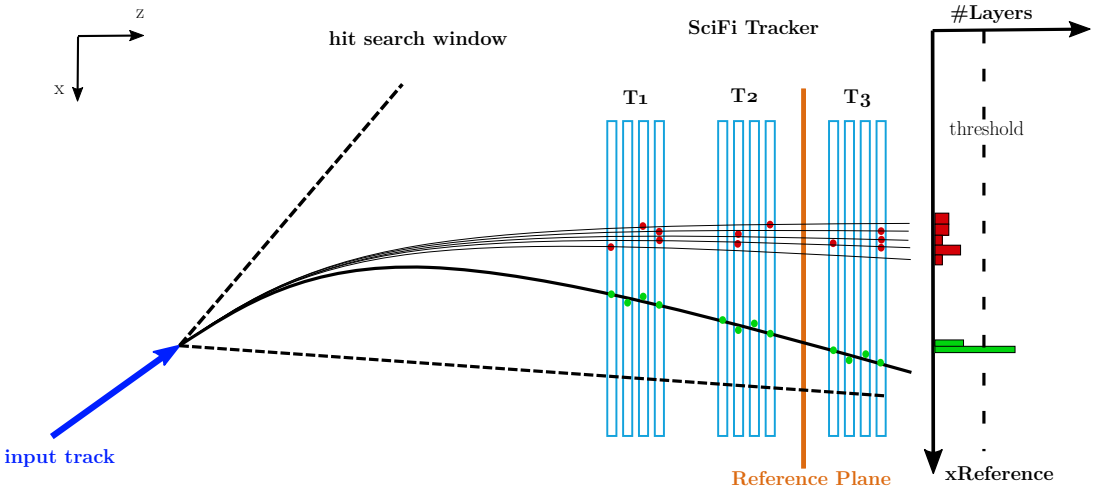


Figure 5.8: Sketch of the Hough-like transform. Different hit projections for one input track are shown. The hits are collected in a histogram, where the true hits belonging to the input track (green) reach bin counts over the threshold, while wrong hits (red) form a more uniform distribution.

yields a sufficient[1] hit efficiency at the momentum border (see Figure 5.4). Next, the hits with the smallest and largest x positions within the search window must be searched for. The hit container data structure, depicted in Figure 5.2, is well-suited for this because $\mathcal{O}(100)$ hits are stored in ascending order for each zone. This makes binary search [96] the optimal algorithm in terms of computational complexity to find the hits at the edges of the search window. Yet it was found that the vanilla binary search suffers from branch mispredictions[2] by the CPU (see Section 3.3) and poses a bottleneck in the performance of the optimised Hough-like transform. Therefore, a branchless implementation of binary search is used [59, 65], which avoids the bottleneck. Because the search typically needs seven iterations[3] to find the lower bound, the first iterations are implemented as an unrolled loop to avoid the loop overhead. Yet without additional checks, if the search window is larger than the largest hit position, the unrolled binary search loop makes an out-of-bounds access. This is avoided by putting the sentinel at the end of the zone (see Figure 5.2).

Hit projection Before reducing the selected SciFi hits to track candidates, they must be mapped to a space that reveals their interconnection. The mapping is done by projecting SciFi hits to the reference plane using the simplified track model. If

[1]The hit efficiency could be increased by enlarging the search window for the last SciFi station. However, the recovery of some low-momentum tracks does not seem worth the extra computational effort.

[2]The CPU cache is not the most pressing bottleneck here because the data is stored sequentially in memory and typically only spans $\mathcal{O}(10)$ cache lines.

[3]Complexity of binary search is $\mathcal{O}(\log_2 n)$, which gives $6 + 1$ iterations for 100 hits.

hits come from the same particle, they form local clusters in this Hough space. For a projection using Equation 5.2, knowledge about the input track's magnet kick z position is required. As described in Section 5.2.4, an iterative method is applied to infer z_{mag} from an initial estimate of Δt_x via Equation 5.7. However, the projection still has to be corrected for the fringe field within the SciFi tracker. This is done using a parameterisation of the expected curvature within the SciFi tracker (see Equation 5.4 and Figure 5.6). An important parameterisation feature is that it vanishes for high-momentum tracks ($p \to \infty \implies \Delta t_x \to 0$) because they hardly deviate from a straight line in the SciFi tracker. The fringe-field correction is obtained by

$$\Delta x = \Delta t'_x \left[T_c + T_d \left(z_{\text{zone}} - z_{\text{ref}}\right)\right] \left(z_{\text{zone}} - z_{\text{ref}}\right)^2 \tag{5.8}$$

with

$$T_{c/d} = c_0 + c_1 t_x + c_2 t_y + c_3 t_x^2 + c_4 t_x t_y + c_5 t_y^2$$

and the coefficients for T_c and T_d given in Tables A.3a and A.3b, respectively. They are found by extrapolating simulated input track states into each SciFi layer to get a set of perfect[1] track measurements. These are then fitted using Equation 5.3 such that the distributions of c_x and d_x in dependence on other track parameters can be studied using the method described in Appendix A.3. The projection including the correction reads:

$$x_{\text{proj}} = x_{\text{mag}} + \frac{x_{\text{hit}} - \Delta x - x_{\text{mag}}}{z_{\text{zone}} - z_{\text{mag}}} (z_{\text{ref}} - z_{\text{mag}}) \tag{5.9}$$

In the low-momentum regime, hits in the first SciFi station deviate up to several millimetres from the straight-line trajectory, making this correction an essential ingredient for precisely projecting the hits to the Hough space. The relevant metric here is the width of the local hit cluster in Hough space shown in Figure 5.9. The narrower the clusters are on average, the easier they are to distinguish from random hit-input-track combinations. Low-momentum tracks, in particular, exhibit cluster widths that, without the correction, would be larger or of the same magnitude as accidental combinations[2] of the input track with a set of SciFi hits.

Additionally, for the projection of stereo-layer hits, the difference between the measured x value at $y = 0$ (x'_{hit} in Figure 5.10) and the actual x position at y_{track} (x_{hit} in Figure 5.10) has to be taken into account. Otherwise, the x_{proj} positions of stereo-layer hits would systematically deviate from the ones of x-layer hits, impeding spotting a local hit accumulation in Hough space. The difference can be accounted for by

$$x'_{\text{hit}} = x_{\text{hit}} - \tan(\alpha_{\text{stereo}}) y_{\text{track}} \tag{5.10}$$

[1] *Perfect* here means that only the trajectory through the magnetic field is taken into account, and no material effects are considered.

[2] Such combinations often occur when a particle decays between the UT and the SciFi tracker: a real VELO track is then often paired with the SciFi hits from one of the decay products.

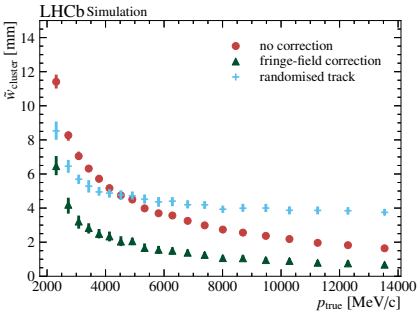


Figure 5.9: Median Hough cluster width comparison between Optical Model method without correcting for the fringe field, with correction for the x position (Equation 5.8), and with randomised track parameters. The randomised track has its VELO track parameters smeared by a Gaussian to mimic a close-by input track for which the SciFi hit set would also be a possible extension.

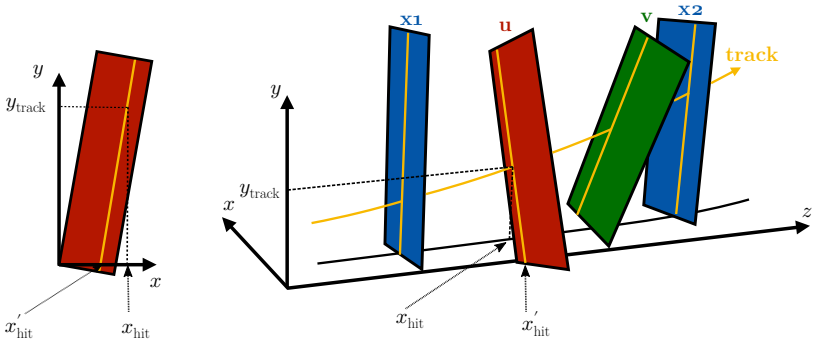


Figure 5.10: Sketch of the measured x position x'_{hit} and the real x coordinate of the track's SciFi mat traversal x_{hit}, which depends on the track's yz trajectory.

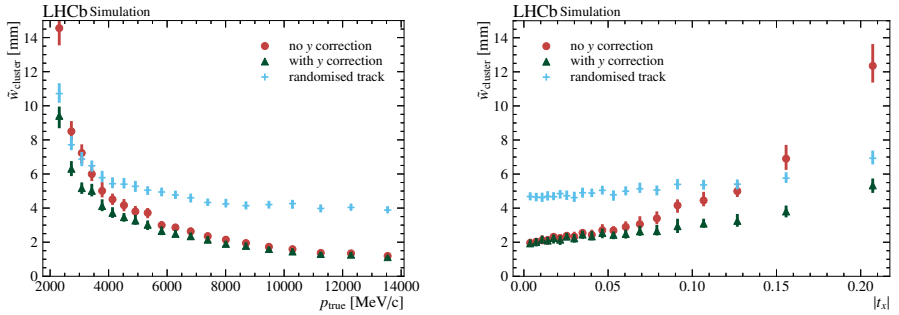
where $\alpha_{\text{stereo}} = \pm 5^\circ$ is the nominal SciFi stereo-mat rotation. As shown in Figure 5.11 (red circles), using a simple straight-line yz trajectory with parameters taken from the VELO track as a first approximation for y_{track} already works well. This is due to the small track curvature in the y-z plane (see also Section 5.2.3). Nevertheless, applying a correction to the straight-line extrapolation

$$y^{\text{L}}_{\text{track}} = y_{\text{v}} + t_y(z^{\text{L}}_{\text{zone}} - z_{\text{v}}) + y^{\text{L}}_{\text{corr}} \tag{5.11}$$

where $y^{\text{L}}_{\text{corr}}$ is calculated using a parameterisation for each stereo layer

$$y^{\text{L}}_{\text{corr}} = \Delta t_x(c^{\text{L}}_0 + c^{\text{L}}_2 t_x t_y + c^{\text{L}}_5 t_x t_y^3 + c^{\text{L}}_6 t_x^3 t_y) \tag{5.12}$$

$$+ |\Delta t_x|(c^{\text{L}}_1 t_y + c^{\text{L}}_3 t_y^3 + c^{\text{L}}_4 t_x^2 t_y + c^{\text{L}}_7 t_x^2 t_y^3) \tag{5.13}$$



a: Width in dependence on true track momentum.

b: Width in dependence on reconstructed VELO x slope.

Figure 5.11: Median Hough cluster width comparison between estimating y_{track} using a straight line (no y correction), with correction of the y coordinate (Equation 5.12), and with randomised track parameters.

with coefficients listed in Table A.4, can significantly decrease the width, as shown in Figure 5.11 with green triangles. This is particularly the case for low-momentum tracks (Figure 5.11a) and tracks with a steep slope in x (Figure 5.11b). The latter is due to the larger momentum p_x orthogonal to the magnetic field component B_z deflecting the trajectory in the y-z plane. The need to estimate y_{track} makes the projection of stereo hits less precise than the x-hit projection (*cf.* Figures 5.9 and 5.11).

Looking at the projected hit positions on the reference plane, *i.e.* in Hough space, a human could now already spot good hit-cluster candidates that match the input track pattern as shown in Figure 5.12. Fortunately, the spotting of candidate clusters and the projection must happen so fast that humans are unfit for this job. However, the CPU must also be carefully instructed on how to project, spot and pass on candidates

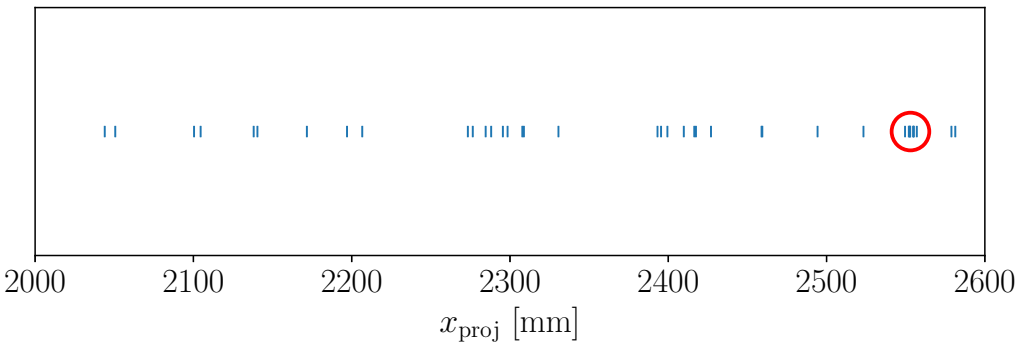


Figure 5.12: Hit projections for a simulated π^+ with $p = 2.5\,\text{GeV}/c$. A red circle marks the hit cluster belonging to the pion.

fast. This is achieved by fully implementing the loop over the measured x positions per zone with SIMD instructions. The platform foreseen for data taking during Run 3 enables vectorisation with AVX2 and FMA (see Section 3.3). The projection thus happens for eight single-precision x-positions in parallel with the evaluation speed of the polynomials used in the simplified track model additionally profiting[1] from fewer CPU operations due to FMA. The design of the hit container data structure with all x positions sequentially in memory was chosen precisely to perform this vectorisation efficiently.

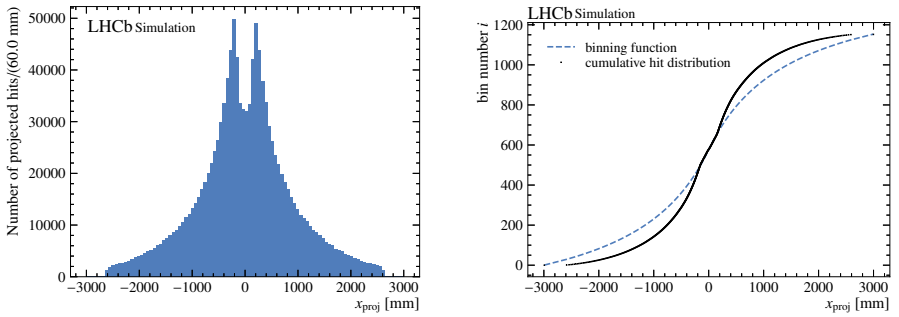
Hough histogram A key optimisation ingredient for the Hough transform is a fast way to find the Hough clusters. Three observations can give guidance on how to do this:

1. the number of SciFi layers within a cluster is the strongest discriminant against random clusters
2. projected hits from x layers are more reliable
3. the centre of the SciFi tracker has the highest hit density and thus the largest potential for random clusters

The data structure used to collect the projected hit position should, therefore, be able to exploit these three properties efficiently. The first is simply a matter of counting, for which the histogram is a natural choice. The second is taken into account by only persisting the projection of x-layer hits, which saves memory space and improves cache locality but, more importantly, saves the time it takes to write the less useful stereo-layer information to memory. The third can be used to define a non-uniform binning for the histogram. In an average event, most projected hit positions are distributed around the beam-pipe position at $x = 0$, with the distribution quickly falling off towards the edge of the detector as shown in Figure 5.13a. It would be a waste of resources to have the same histogram granularity at the edges of the detector as in the centre. For the same reason, detectors are often designed with coarse granularity in regions of low particle flux, which is not the case for the SciFi tracker. The shape of the desired non-uniform binning is obtained by transforming the bin edges such that the hit distribution becomes uniform. To do this, the i-th bin edge is identified with the i-th quantile x^i_{Q}, defined via the cumulative distribution:

$$F(x^i_Q) = \int_{-\infty}^{x^i_{\mathrm{Q}}} f(x)\mathrm{d}x = \frac{i}{N_{\mathrm{bins}}} \tag{5.14}$$

[1] Depending on how the compiler implements multiplication followed by addition, FMA might even yield a more precise floating-point result, because only one rounding step is necessary. This potential precision difference does not matter for the physics performance but can cause numerical differences between results obtained on different platforms.



a: Distribution of projected hits positions from several tracks.

b: Cumulative distribution of projected hits.

Figure 5.13: Distributions used to determine the binning scheme of the Hough histogram.

The distribution of the projected hits $f(x)$ and its cumulative distribution are solved numerically[1] for the i-th bin edge x_{Q}^{i}. The number of bins N_{bins} is a vital parameter for the Hough histogram because it specifies the scale of the bin size; a small number can be processed fast but makes it difficult to spot real hit clusters in the noise of random hits also being present in a bin, while a large number impacts the computational performance because clusters may be split across many bins, each of which must be processed. Conveniently, the necessary scale of the bin size is already known from the precision of the hit projection shown in Figure 5.9. The smallest clusters have a width of about 1 mm, which is the smallest bin size in the cumulative distribution plotted in Figure 5.13b with 1152 bins. Now for each projected hit position, a mapping is needed to determine which bin it belongs to. There are two possibilities: calculate a look-up table that stores the bin numbers corresponding to all possible x positions on the reference plane or find a function that effectively describes the shape of the distribution. The former possibility was abandoned because the computational performance of the loop over the hits already suffers from cache misses, which worsens if more information stored in memory needs to be accessed (see Section 3.3). A function, however, is only better if it can be evaluated fast and for multiple hits in parallel, *i.e.* with SIMD instructions. This essentially leaves one choice for the description of the sigmoidal form:

$$i(x_{\text{proj}}) = \lfloor p_0 + \frac{p_1 x_{\text{proj}}}{1 + p_2 |x_{\text{proj}}|} \rfloor \tag{5.15}$$

Although it is adequate to use this function to fit the cumulative distribution, the smallest bin size was chosen a bit too tight. In the end, the average cluster width depends on momentum and not on the position on the reference plane. Studying the performance for different numbers of bins and coefficients in Equation 5.15, it was found that $N_{\text{bins}} = 1152$ performs well if the binning function is tweaked such that the

[1]The Python package `pandas` provides a neat function for this called `qcut`.

minimum bin width is 1.7 mm. The binning function used in the algorithm is shown in Figure 5.13b; the coefficients are listed in Table A.9.

Because the loop over the hits within the search window is performed per zone, the result of the vectorised Equation 5.15 is a vector of eight single-precision integers representing the Hough histogram bins for projected hits from the corresponding SciFi layer. The histogram counts the number of unique layers per bin, so a fast and vectorised way to keep track of this is needed. In addition, the projected positions of the x-layer hits and their indices in the SciFi hit container need to be stored. An SoA data structure is used for this, with two fields storing the x_{proj} values and hit container indices per bin, respectively, and one field to encode[1] the total number of layers per bin, together with the information on which specific layers are present. To update the layer information for all bins inside the vector, the encoded layer information for this bin must be gathered from memory, updated, and scattered back[2]. This poses one of two bottlenecks in the loop over the hits because these memory operations have high latency. The other bottleneck is scattering the projected x-hit position and its hit-container index to the memory representing the Hough histogram. An essential subtlety about the data structure has to be noted: the number of hits stored per bin in the first two fields is restricted to 16. In practice, this is enough to hold all x-layer hits that fall into the bin on average. After all, the binning follows the hit density in the detector, the total number of SciFi hits is restricted by the bandwidth of the read-out hardware (see Section 2.4.3), and a global event cut[3] is applied for nominal running.

Hough search After projecting every hit from within the search window to the reference plane and storing the necessary information in the Hough histogram, the candidate hit clusters must be extracted (`pickUpCandidateBins` in Figure 5.1). First, a fast threshold scan over the field encoding the layer information is performed. This can be done for eight bins in parallel using SIMD instructions because the field has the SoA layout. If a single bin contains hits from at least four SciFi layers, it is promoted to a cluster candidate. In a subsequent step, both neighbouring bins of a candidate are checked for complementary layers, requiring a total of at least ten layers. This is done for eight candidates in parallel. The distribution of the number of bins a

[1] For details of the encoding it is best to look into the code. The basic idea is to count the number of unique SciFi layers in the first 8 bits of a 32-bit integer while the following 12 bits serve as flags, keeping track of which layer is present in a bin. This slightly cumbersome approach was chosen to use the SIMD instructions available with AVX2 efficiently.

[2] *Gathering* and *scattering* is SIMD instruction terminology, meaning that values are loaded and stored from memory addresses reachable via shared base address, *i.e.* not necessarily directly sequential in memory. According to Ref. [97] gathering needs 14 CPU micro-operations on the reference node's Broadwell architecture and is thus a heavy instruction. Scattering is not implemented in hardware for AVX2 and is emulated by the SIMDWrapper library.

[3] For *pp* collisions, the global event cut currently removes events with more than 9750 SciFi clusters. This might change in the future depending on the general performance of the experiment. For heavy-ion collisions, the global event cut is best removed, in which case the restriction to 16 hits per bin might need to be revised.

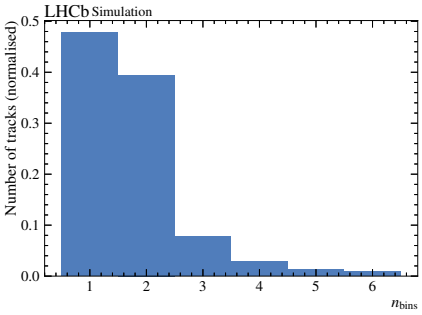


Figure 5.14: Number of bins in the Hough histogram the projected hits of a reconstructible track are split across.

track is split across is shown in Figure 5.14; the median number is two, which is the expected number if the binning has the right granularity. Building candidates using three bins covers 96% of particles, where mainly tracks that scattered with large angles and low-momentum particles are lost. Some low-momentum tracks can be recovered later, however, only if a rogue hit is present, which lets the candidate pass the ten-layer requirement.

5.2.6 Track Candidate Selection

The optimised Hough-like transform results in a list of bins containing a sufficient amount of hits matching the pattern expected for the input track. The next step is cleaning up the hit sets using the stored x_{proj} values from the x layers to eventually form initial track candidates (`selectXCandidates` in Figure 5.1). This is necessary because random hits can be present in candidate bins, and sometimes multiple track candidates can be formed from the hit sets. Broadly, the approach is the following:

1. sort the x_{proj} values in all selected bins and write them to a list
2. form reduced sets of hits from the list by checking for gaps and narrow sub-clusters
3. fit the x hits with the xz-trajectory model (Equation 5.3) and reject bad hit sets

If a candidate was found using the x-layer hits, it has to be confirmed by checking for matching stereo-layer hits (`selectFullCandidates` in Figure 5.1). This uses the track candidates found in the previous step to predict the position where the track traverses the stereo layers:

4. select stereo-layer hits close to the xz trajectory by employing another Hough-like transform
5. fit stereo-layer hits with a yz-trajectory model and reject bad hit sets
6. fit all hits with the xz-trajectory model

All the above steps can be executed a second time if no appropriate track candidate was found at first. This is called *second loop* and was found to balance the algorithm's overall performance by addressing the fact that the difficulty of finding a track candidate depends on the quality of the hit projection. The first loop applies more strict selections and thus mostly finds the high-quality[1] tracks. The second loop is, on average, entered for more than half of the input tracks that pass the Hough search. Omitting the second loop costs 3.5% track finding efficiency but also detracts 1% from the fake track fraction and reduces the time spent in the Forward tracking by 20%. It is a general observation that a significant time in the algorithm is spent on recovering crude tracks. In the following paragraphs, selection parameters that are different for the second loop are given in brackets behind the values for the first loop.

Sorting The sorting of x_{proj} values posed a serious computational bottleneck in the early stages of the algorithm development. Various approaches have been tested, from sorting algorithms specific to the order in which the projected positions appear in the list [98, 99], over generally optimised sorting [100], to avoiding the sorting altogether by using different hit clustering approaches. In the end, the best-performing approach computationally and physics performance-wise was developed in conjunction with the optimised Hough transform. Using the properties of the Hough histogram, the sorting can be done in small chunks[2] of work instead of one heavy sorting procedure. The sorting profits, in particular from the ordering property of the Hough histogram: only hits within a bin must be sorted, all x_{proj} in the higher, adjacent bin are guaranteed to be larger by the strict monotonicity of the binning function. Furthermore, the size of each bin is restricted to 16 hits and is thus small from a computational point of view, fitting inside a single cache line. The most efficient sorting algorithm on modern CPUs for this scenario is Insertion Sort [59, 96], which is thus opted for here.

Clusterisation On the resulting list of sorted x_{proj} values, the candidate clusterisation starts by forming an initial cluster out of the first five(four) hits. The width of this cluster is checked against

$$w = w_{\text{max}} + b_w(|x_{\text{proj}}^{\text{min}}| + |x_{\text{proj}}^{\text{min}} - x_{\text{ref}}^{\text{straight}}|) \tag{5.16}$$

where $x_{\text{proj}}^{\text{min}}$ is the smallest x_{proj} in the cluster and $x_{\text{ref}}^{\text{straight}}$ the straight line extrapolation of the input track to the reference plane. The other two are configurable parameters set to $w_{\text{max}} = 1.2\,\text{mm}(1.5\,\text{mm})$ and $b_w = 0.002\,\text{mm}^{-1}$. The width described by Equation 5.16 is larger for low-momentum candidates[3], which accounts for the behaviour

[1] This is often the same as high-momentum tracks.

[2] This is an excellent example of the *divide-and-conquer* technique [96] from computer science.

[3] Momentum is not the only relevant track parameter here. Also steep tracks, *i.e.* with a large t_x, tend to have larger cluster widths because they are less well modelled by the parameterisations used for the hit projection.

already seen in Figure 5.9. The cluster width function defined in Equation 5.16 mostly overestimates the width. This is on purpose to not lose any candidates at this stage and makes the cluster finding more robust against misalignment of the detector. Yet, a better model for the width might pose a further optimisation opportunity.

If the width of the initial cluster is larger than w, the first hit is omitted, and the next hit from the list is added. This is repeated until a cluster with a width smaller than w is found or the end of the list is reached, in which case no SciFi extension of the input track is found. Otherwise, the next hits from the list are added to the cluster if the width of the resulting hit cluster is smaller than w and the hits' x layers are not used in the cluster yet. If x_{proj} of the next hit is less than 1.2 mm(0.5 mm) away from the previous one, it is added regardless of the width w. The number of layers present in a cluster is still the best discriminant against random hit combinations. Therefore, a cluster candidate is rejected if it consists of less than five(four) x layers. The accepted clusters, then, on average, consist of seven(five) hits, ranging up to cluster sizes of 40(10) or more hits in rare cases. Of these, only up to six can be the true x hits belonging to the input track. Consequently, the clusters need to be stripped of rogue hits. The most likely case is that the cluster has at least two SciFi layers contributing a single hit while other layers contribute multiple hits. If so, a straight line is fitted to the single-layer x_{proj} values, and only hits with the smallest χ^2 with respect to that line are used such that each SciFi layer only contributes a single hit (see also the next paragraph on how fits are implemented within the algorithm). If the cluster has only multiple hits, it is directly passed on to the next step.

Simple x-projection fits Although finding zero, one or two clusters per input track are the most abundant cases with a median of 3(0), the average number is 15.7(11.8) and thus, random hit combinations need to be rejected. Since the clusters are already made of at least five(four) x hits, the track model within the SciFi tracker from Equation 5.3 can be used to clean the sets of hits further and evaluate if they fit the expected trajectory. It has to be noted that fitting the clusters is not a computationally cheap operation. The reason is the minimisation of the model's distance to the data points. Conveniently, the track model applied here consists only of polynomials, hence fits using the linear least-square method have an analytic solution to the minimisation problem. Nevertheless, the analytic solution requires a potentially expensive matrix inversion. The model from Equation 5.3 has four parameters, *i.e.* the inversion of a 4×4 matrix would be needed. This could be done efficiently using Cholesky decomposition[1] [101]; however, the inversion of matrices with lower rank is reasonably straightforward, a summary of which is given in Appendix A.2. This, combined with the observation that the parameter d_x in Equation 5.3 can be kept at a fixed value to describe the trajectory well, makes fitting the x hits with only the parabolic part of the model a

[1]The decomposition involves evaluating square roots, which have some latency and can be a computational bottleneck for small matrices.

good solution.

First, a track candidate is initialised with preliminary values for a_x, b_x, c_x and a constant for d_x. The initial offset a_x is given by the mean of the x_{proj} values. The slope b_x is estimated using Equation 5.6 and the curvatures c_x and d_x are obtained evaluating the same parameterisation already used during the hit projection (Equation 5.8). To reject a first set of random clusters, it is enough to fit only the linear part of the track model while the quadratic and cubic coefficients are kept constant. For each hit in the cluster, the χ^2 with respect to the fitted model is calculated. The hit with the highest χ^2 is removed if the cluster has multiple hits on the corresponding SciFi layer or the value is larger than 100. If a hit is dropped, the fit and hit removal is repeated until no hit is rejected or the number of hits falls below 5(4), where the whole cluster is rejected. After the linear fit, the cluster consists only of a single hit per layer. The updated parameter estimates for a_x and b_x are then used to fill possible holes in the trajectory, which improves the hit efficiency and, eventually, the track finding efficiency. Missing hits can occur either because they were not picked up by the clusterisation or ended up in a different bin than the ones considered during the Hough search (see Figure 5.14). To recover those hits, the fitted model predicts the track's x position on a missing layer and the χ^2 of hits within a search window, defined by Equation 5.16, is calculated. The hits with the lowest χ^2 from each missing layer are added to the cluster, and the linear fit is repeated. The effect on the track finding efficiency is substantial; without the hit recovery, about 3% of reconstructible tracks are lost in this step. After the linear fit, the number of cluster candidates is reduced to a median of 1(0) at the cost of roughly 10% of the algorithm's execution time.

For further fits, to ensure an accurate fit result, the tilting of the SciFi layers with respect to the y coordinate, $\mathrm{d}z/\mathrm{d}y$ (see Section 2.4.3), is taken into account from here on, *i.e.* some knowledge about the yz trajectory is required to precisely know the z position of the x hits. Without any correction, the measured z positions of the hits would deviate up to several millimetres from their true value as shown in Figure 5.15. A second-order polynomial models the yz trajectory

$$y(z) = a_y + b_y(z - z_{\text{ref}}) + c_y(z - z_{\text{ref}})^2 \tag{5.17}$$

Analogously to the estimates of the x-trajectory model parameters, the coefficients of the y-trajectory model are parameterised using polynomials. An estimate for the y-offset at each layer was already used during the Hough transform to precisely project the stereo hits, so the parameterisation function given by Equation 5.12 can be re-used but with parameters listed in Table A.5. The slope b_y is modelled by

$$\hat{b}_y = t_y + \Delta t_x(c_0 t_y t_x + c_4 t_y t_x^3 + c_1 t_y \Delta t_x) + |\Delta t_x|(c_2 t_y^3 + c_3 t_y t_x^2 + c_5 t_y^3 t_x^2) \tag{5.18}$$

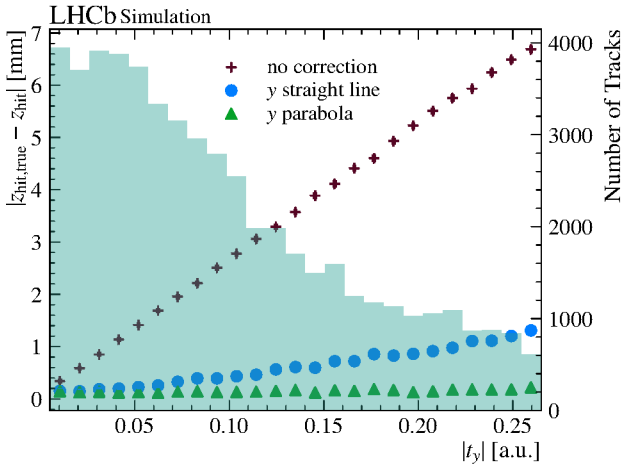


Figure 5.15: Average residual of the hits' measured z positions for tracks with momenta $p \in [1.5, 2]\,\mathrm{GeV}/c$ in dependence of the absolute y slope. The dark-red crosses show the deviation if no correction is applied. The blue circles demonstrate the residuals if the yz trajectory is assumed to be a straight line with slope t_y, while the green triangles have the full correction applied, using the parabolic yz trajectory with parameterised coefficients. The histogram shows the distribution of $|t_y|$ for $p \in [1.5, 2]\,\mathrm{GeV}/c$.

with the coefficients listed in Table A.6. Similarly, c_y is estimated using

$$\hat{c}_y = \Delta t_x(c_1 t_y t_x + c_2 t_y \Delta t_x) + |\Delta t_x|(c_0 t_y + c_3 t_y^3 + c_4 t_y t_x^2) \tag{5.19}$$

with the coefficients listed in Table A.7. The slope difference Δt_x is determined using the b_x value from the previous linear x-trajectory fit as defined in Equation 5.5. A better estimate using Equation 5.17 for the z position of a hit is then calculated by

$$z'_{\mathrm{hit}} = z_{\mathrm{hit}} + \tan(\alpha_{\mathrm{d}z/\mathrm{d}y}) y(z) \tag{5.20}$$

where the nominal tilting angle is $\alpha_{\mathrm{d}z/\mathrm{d}y} = 0.21°$. The correction given by Equation 5.20 removes the deviation almost completely, as shown in Figure 5.15. The candidates that survived the linear fit are now fitted with c_x as an additional floating parameter, *i.e.* a parabola is fitted with the cubic coefficient d_x kept constant. The procedure is the same as for the linear fit, yet with slightly altered requirements for the hit removal: the hit with the largest χ^2 is dropped if it is larger than 15 or the reduced χ^2 of the fit is larger than 7. If the candidate also survives this fit, the updated model parameters are used to search again for matching hits on missing layers. If successful, this is followed by repeating the fit. Afterwards, a cluster is promoted to an x-track candidate, and its hits are removed from the sorted list such that they cannot be used by the next cluster or in the second loop.

At this stage, the number of clusters has been reduced to 1.7(1.8). For the clustering, only hits from the x layers have been used so far and the clusters look like true track candidates. However, only one extension per input track is expected, which shows that further selection is needed. This is also reflected in the fake track fraction, which is unacceptably high at $r_{\text{fake}} \simeq 83\%(99\%)$. Nevertheless, the heaviest part of the combinatorics reduction is done at this point, with the selection of the x-track candidates taking roughly 30% of the algorithm's execution time. Due to its high computational cost, it has been studied whether this part would profit from utilising SIMD instructions, *e.g.* by fitting several clusters in parallel. It was found that vectorised versions are not faster than the scalar implementation described above. The main reason is the already small number of clusters coming from the Hough transform, which makes the necessary changes for vectorised code inefficient for large chunks of the data. An additional inefficiency of a SIMD version lies in the varying sizes of the clusters. A vectorised loop over the clusters' hits must always continue until the largest cluster is finished, essentially introducing the worst-case execution time for all clusters in a SIMD vector. The best performance, for now, was reached by storing the x_{proj} values and their indices in the SciFi-hit container in an SoA memory layout as these are accessed sequentially during the clusterisation and kept in the CPU cache. The information used in the fits is taken directly from the SciFi-hit container, where it is stored in an AoS layout to improve cache locality (this is referred to as *hot part* in Section 5.2.2). This performs better than an SoA layout since the memory access pattern is not sequential; the fitted hits always come from different zones. Ultimately, the computational performance of the clusterisation and the candidate selection strongly depends on the quality of its input, *i.e.* any precision improvement of the Hough transform also improves the computational performance of the subsequent step. This is, of course, only successful if the precision improvement comes at a lower computational cost than what is gained later.

Stereo-hit selection The compatible stereo hits must be found for each previously found x-track candidate. The Hough transform already used stereo-hit information to ensure that Hough clusters exhibit enough stereo layers to form an entire track, *i.e.* at least ten hits from a different SciFi layer each. Because of the more considerable uncertainty on the projection of the stereo hits, the information was not persisted, though. Hence, the stereo hits must be collected from the hit container. With the xz trajectory already constrained by the x-layer hits and some input tracks having already been rejected, this is more precise and computationally less expensive than storing the stereo-hit information during the Hough transform.

The stereo hits are collected by extrapolating the xz trajectory into each stereo layer. The predicted position $x'^{\text{pred}}_{\text{stereo}}$ of the stereo hit is obtained by correcting for the stereo angle using Equation 5.10. The $y_{\text{track}} = y(z_{\text{zone}})$ coordinate on the stereo layer is calculated using the y-trajectory model from Equation 5.17 with the coefficients as

set in the previous step. Similarly to the x-hit clusterisation (*cf.* Equation 5.16), stereo hits from each stereo zone are accepted if they fall within a search window defined by

$$\begin{aligned} \Delta x_{\text{stereo}} &= \Delta x_{\max} + b_{\Delta x}\left(|y(z_{\text{zone}})| + \left|x'^{\text{pred}}_{\text{stereo}} - x^{\text{straight}}_{\text{zone}}\right|\right) \\ x_{\text{stereo,min}} &= x'^{\text{pred}}_{\text{stereo}} - \Delta x_{\text{stereo}} \\ x_{\text{stereo,max}} &= x'^{\text{pred}}_{\text{stereo}} + \Delta x_{\text{stereo}} \end{aligned} \tag{5.21}$$

with $\Delta x_{\max} = 5\,\text{mm}$ and $b_{\Delta x} = 0.002$, which covers more than 99% of the sought-after, true hits. The limiting factor here is the knowledge about the yz trajectory, which, at this point, only comes from the parameterisations used to estimate the yz-trajectory model's coefficients (Equations 5.18 and 5.19). Since the parameterisations yield merely an initial guess for the yz trajectory, the calculated value of y_{track} in each zone is usually systematically over- or underestimated. This can be exploited to find the correct set of stereo hits as the predicted stereo-hit position relative to the measured hit position exhibits the same systematic shift. Only the sign of the deviation is different depending on the sign of the layer's stereo angle. The signed deviation of the measured hit position from the prediction is thus defined as

$$\delta x_{\text{signed}} = \text{sgn}(\tan(\alpha_{\text{stereo}}))\left(x'_{\text{hit}} - x'^{\text{pred}}_{\text{stereo}}\right)$$

The stereo hits belonging to the track have similar values of δx_{signed} and are, therefore, selected using another Hough-like transform. The δx_{signed} values of hits within the search window span the Hough space and are filled into a Hough histogram with 15 bins and a bin width of 2 mm. The histogram's range[1] depends on the configuration of Equation 5.21 and is calculated using reasonably large position values $y(z_{\text{zone}}) = 2\,\text{m}$ and $x'^{\text{pred}}_{\text{stereo}} - x^{\text{straight}}_{\text{zone}} = 3\,\text{m}$.

When collecting the stereo hits to confirm an x-track candidate, a subtlety regarding the split of the zones into the upper and lower half of the detector must be considered. The distinction hardware-wise comes from the fact that there are separate SciFi modules for both halves of the tracker. However, the modules in the stereo layers reach into the other half due to their rotation as illustrated in Figure 5.16. If the value of $y(z_{\text{zone}})$ is smaller than 22.7 mm, stereo hits from the other half are therefore also collected in the Hough histogram. Without this *triangle search*, the track finding efficiency is roughly 0.5% lower.

After filling the histogram, a threshold scan requires hits from at least three stereo layers in one bin. If a bin is over the threshold, the two neighbouring bins are checked

[1] If this range needs to be changed for whatever reason, the proponent should ensure that the number of bins, the bin width, and the range of the histogram work well together. They cannot be chosen independently. The number of bins is static, *i.e.* hard-coded in the C++ code, and the current configurable parameters are chosen such that the histogram covers $\delta x_{\text{signed}} \in [-15, 15]\text{mm}$, which works well for a bin width of 2 mm. In practice, this only needs to be changed if there is reason to believe that the magnitude of δx_{signed} values changed, *e.g.* because of massively improved predictions or a massively misaligned detector.

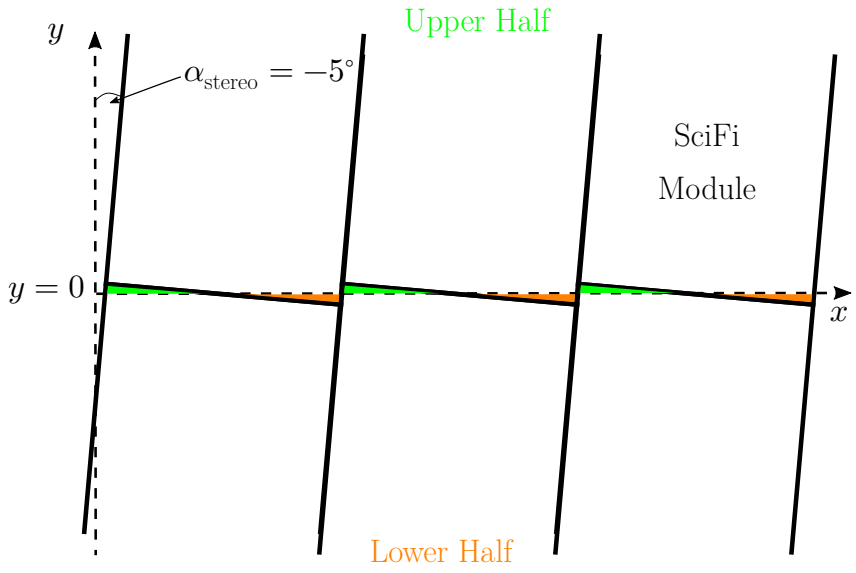


Figure 5.16: Illustration of the SciFi stereo modules reaching into the opposite side of the detector.

for hits from other layers. If the three bins together exhibit enough stereo layers such that a track candidate with at least ten hits can be formed, the set of stereo hits is promoted to a stereo-track candidate. The median number of stereo-track candidates created per x-track candidate is zero. This is precisely expected, given that the fake x-track fraction is 83%. For about a quarter of the x-track candidates, a single stereo-track candidate is found, 11% have two, and roughly 4% have three or more stereo-track candidates. For this part of the algorithm, it was also studied whether vectorisation can be used to improve the computational performance. However, the effort was quickly abandoned because neither the number of stereo hits collected nor the number of x-track candidates is sufficiently high to fill SIMD vectors.

Simple y-projection fits Just as was the case for the clusters of x hits, the sets of stereo hits can include rogue hits, which must be removed. This is again done by fitting the track model to the stereo hits with a strategy analogous to the x-projection fits described before. For each stereo-track candidate, first, only the linear part of Equation 5.17 is fitted, and hits with a $\chi^2 > 60$ are removed. The candidate is rejected if the sum of the remaining stereo hits and the x hits on the x-track candidate is less than ten. Subsequently, the full parabolic y-trajectory model is fitted, dropping hits with $\chi^2 > 4.5$ and possibly rejecting the stereo-candidate according to the same minimum hit requirement as previously. If the stereo candidate has missing SciFi layers, the fitted y-trajectory model and the fitted x-projection are used to collect hits within the window defined by Equation 5.21. The hit with the smallest $\chi^2 < 4.5$ from

each missing stereo layer is added before the y projection is fitted a last time. Because the stereo layers of the SciFi tracker still mostly measure the x position, the hit has to be projected onto the y axis to be fitted by the y model. The distance between the yz trajectory and the projected y hit is given by

$$\begin{aligned} d_y &= \frac{x(z'_{\text{hit}}) - x_{\text{hit}}}{\tan(\alpha_{\text{stereo}})} \\ &= \frac{x(z'_{\text{hit}}) - [x'_{\text{hit}} + \tan(\alpha_{\text{stereo}})y(z'_{\text{hit}})]}{\tan(\alpha_{\text{stereo}})} \end{aligned}$$

The covariance matrix holding the hit errors for the fit is populated with the stereo hits' uncertainty due to the SiPM channel clusterisation (see Section 2.4.3). This neglects the additional uncertainty picked up by the stereo hit projection onto the y axis, *i.e.* the errors used in the fit are underestimated, and the y-trajectory is actually poorly constrained. The errors could be improved by studying the actual uncertainty of d_y in simulation; however, the y-projection fits are merely used to confirm the x-track candidate and neither their estimator covariance nor the χ^2 distribution are used further. Instead, a virtual measurement at z_{mag} is included in the fit to improve the quality of the parabolic fits. This builds on the circumstance that the bending of the yz trajectory is small. The distance $d_y^{\text{mag}} = (a_y + b_y(z_{\text{mag}} - z_{\text{ref}})) - (y_{\text{v}} + t_y(z_{\text{mag}} - z_{\text{v}}))$ is used with a weight given by $w = (10\,\text{mm} + 0.015\Delta t_x)^{-2}$.

Suppose multiple stereo candidates per x candidate survive the fits. In that case, the candidate with the most stereo hits is selected[1]. In case of an equal number of hits, the candidate with the smallest mean squared distance to the model is chosen.

Final x-projection fit With an x and stereo candidate found, all candidate hits within the SciFi tracker are fitted. The hits from x layers contain no information about the yz trajectory, and also the stereo-layer hits carry mostly x-position information. Furthermore, the track's momentum is most sensitive to the bending of the track in the xz-plane. Therefore only the x-trajectory model is fitted to the full hit set. Again, the same strategy as before removes outliers and adds missed hits. The hit with the worst χ^2 is removed if it is larger than 15 or the reduced χ^2 of the fit is larger than four.

Technically speaking, the pattern recognition and the track finding are complete at this point. The reconstructed tracks could be passed on to the Kalman filter, which would reject low-quality tracks and provide the best estimate of the track parameters. The Kalman filter, however, is the most expensive component of the reconstruction sequence, as already seen in Figure 4.5b. It is, therefore, worthwhile to spend some more time in the Forward tracking to improve the set of tracks handed to the Kalman filter. For this, the essential track parameter estimated by the fit is the slope b_x, used

[1] This is the same as selecting the candidate with the most stereo layers because only one hit per layer is used.

in the next section to estimate the track's momentum.

5.2.7 Momentum Estimation

The momentum of a track is estimated using Equation 4.7. The numerator is given by $\Delta t_x = t_x - b_x$ where b_x is the track's slope at the reference plane within the SciFi tracker as obtained by the fit in the previous section. Before attending to the integral in the denominator, it is helpful to define the expectation on the momentum estimate. At the end of the algorithm, a Long track must have a track state vector in the SciFi detector, including the momentum, as input to the Kalman filter. As long as the hit set on the track has high purity, the Kalman filter works well, even with only a rough initial estimate of the track parameters. The momentum is further helpful for the fake track rejection, particularly when the input tracks already have a momentum estimate to which it can be compared, *e.g.* Upstream tracks. But also in this case, an approximate momentum is sufficient. Eventually, the momentum is used to find UT hits that belong to the Long track. The hits are searched for by using a simple parameterised extrapolation into the UT for which the momentum resolution is not the limiting factor. Hence, a highly accurate momentum estimate is not necessary here, and the integral in the denominator of Equation 4.7 can be approximated using a polynomial[1]. This avoids numerically solving the integral, which would be computationally expensive because of the large distance between the VELO detector and the SciFi stations. The integral is parametrised by

$$\begin{aligned} B_{\text{int}} =& c_0 + t_y^2(c_1 + c_5 t_y^2 + c_6 t_x^2) + t_x b_x (c_{10} t_x + c_3 + c_7 t_y^2) \\ &+ c_{11} b_x^4 + c_2 t_x^2 + b_x^2 (c_4 + c_8 t_y^2) + c_9 t_x^4 \end{aligned} \tag{5.22}$$

with the coefficients listed in Table A.10. The momentum is then calculated as

$$\frac{q}{p} = \frac{\Delta t_x}{r_I B_{\text{int}}} \tag{5.23}$$

with r_I as the relative signed current through the magnet. The momentum resolution achieved with this is $\Delta p/p \simeq 1\%$ and thus not too far from the multiple-scattering limit of $\Delta p/p \gtrsim 0.5\%$ estimated in Section 4.2.2. More on this can be found in Section 5.3.1.

5.2.8 Fake Track Rejection and Duplicate Removal

After the final fit of the x projection using hits from all layers and estimating the momentum, the final piece of the algorithm is cleaning the set of found tracks to keep away unnecessary work from the expensive Kalman filter. The fake track fraction at

[1] Even simpler would be integrating Equation 4.5 assuming a homogeneous magnetic field along the y axis and $t_x, t_y << 1$, $\sqrt{1 + t_x^2 + t_y^2} \simeq 1$ yields $\Delta t_x \simeq -\kappa B L \frac{q}{p}$, where L is the length of the magnetic field.

this point with VELO tracks as input is still significant at about 46%, out of which roughly 12% qualify as duplicates, *i.e.* a track that shares many hits and a phase space region with another track. The cleaning is done in two stages, the first of which is the last part of the `selectFullCandidates` section in Figure 5.1, which employs a neural network to estimate the goodness of a track and rejects low-scoring tracks. The second stage operates on all Long tracks found by the Forward tracking in one collision event and identifies track duplication (`removeDuplicates` in Figure 5.1).

Fake Track Classification Binary classification of tracks into real tracks and fake tracks is a problem well-suited for supervised machine learning. The necessary labelled data is obtained by running the detector signals left by a simulated particle through the track finding and truth-match the found track candidates to the simulated particle. The *embarras de richesses*[1] of machine learning algorithms is avoided by opting for one of the simplest neural networks: a multi-layer perception (MLP). This is a good choice because MLPs model non-linear decision boundaries, which were found to be necessary for the problem at hand, they are straightforward to implement, possibly utilising SIMD instructions, and are fast to evaluate if designed with computational performance in mind. The training is performed using the Toolkit for Multivariate Data Analysis (TMVA) [102] on a mixture of simulated $B_s^0 \to \phi\phi$, $B_s^0 \to J/\psi\phi$, $B^0 \to K^{*0}e^+e^-$, $Z \to \mu^+\mu^-$,$D^{*+} \to D^0\pi^+$ and $D^+ \to K_S^0\pi^+$ decay samples[2]. The networks are fitted using TMVA's backpropagation to adjust the network weights with Cross-Entropy as the loss function. To ensure a high computational performance of the classification task, a rectifier linear unit[3] is used as the neuron activation function. A logistic function calculates the network response, mapping the output to the interval $[0, 1]$. After the training, TMVA provides an implementation of the network in principle directly usable by the Forward tracking. However, because the Forward tracking is not the only algorithm in the reconstruction sequence making use of such a neural network, an optimised generic implementation was added to the LHCb software stack, splitting the network implementation from the definition of the network weights and allowing to change the floating-point data type used by the network, *i.e.* single precision and SIMD data types.

Two different MLPs were trained for the Forward tracking, one for input tracks with momentum information, *i.e.* tracks from the VeloUT algorithm, and one for tracks without knowledge about the momentum, *i.e.* VELO tracks. The latter classification problem is harder because less information about the input is available. This is also reflected in the network architecture, which is given by two hidden layers with $N + 4$ and $N + 2$ neurons, respectively, where N is the number of input variables. The network that has access to the momentum of the input track only has a single hidden layer with $N + 2$ neurons. The input variables are given in Table 5.2. They are essentially

[1] *Qual der Wahl* in German, freely translated to English: Agony of choice.
[2] LHCb simulation version `Sim10-Up02-OldP8Tuning`.
[3] $\mathrm{ReLU}(x) = \max(0, x)$

Table 5.2: Input variables of the two MLPs used in the Forward tracking. Quantities with a hat denote the initial estimate via a parameterisation.

Input-track type	Variable	Preselection
VELO or Upstream	$\lvert a_y - \hat{a}_y \rvert$	$< 140\,\text{mm}$
	$\lvert b_y - \hat{b}_y \rvert$	< 0.055
	χ^2/ndf	< 8
	D_x^{match}	$< 140\,\text{mm}$
	D_y^{match}	$< 500\,\text{mm}$
	$\lvert t_x \rvert$	
	$\lvert t_y \rvert$	
	$\lvert q/p \rvert$	
	$\lvert b_x - \frac{a_x - x_{\text{mag}}}{z_{\text{ref}} - z_{\text{mag}}} \rvert$	
Upstream	$\log(\lvert q/p - (q/p)_{\text{input}} \rvert)$	

a collection of all the information encountered during the track finding and compare the found track with on average expected properties. The only new quantity is the deviation from a straight-line y extrapolation of the input track at the end of the SciFi tracker at $z_{\text{EndT}} = 9410\,\text{mm}$, defined by

$$\begin{aligned} y_{\text{corr}}^{\text{EndT}} =& \Delta t_y (c_0 + c_6 t_x^2 + c_5 \Delta t_y t_y) + \Delta t_x (c_4 \Delta t_x t_y + c_3 t_y t_x + c_8 t_y^3 t_x) \\ &+ |\Delta t_x| (c_1 t_y + c_7 t_y t_x^2) + c_2 |\Delta t_y| t_y \end{aligned} \tag{5.24}$$

where $\Delta t_y = b_y - t_y$ and with coefficients listed in Table A.8. It is used to define the D_y^{match} input variable as

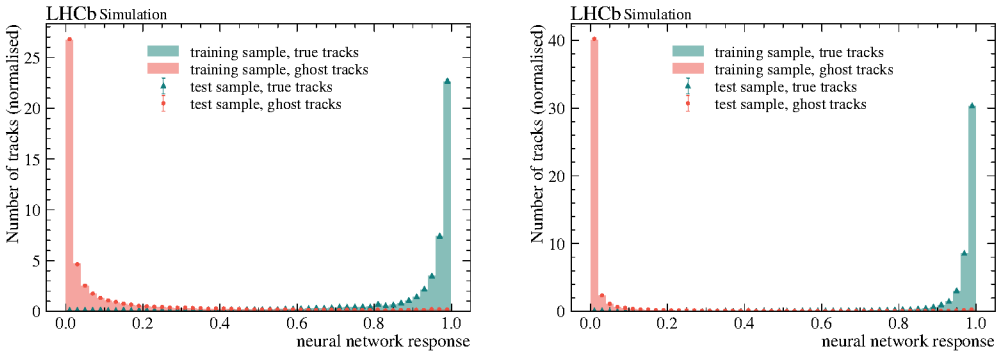
$$D_y^{\text{match}} := |y_{\text{v}} + t_y (z_{\text{EndT}} - z_{\text{v}}) + y_{\text{corr}}^{\text{EndT}} - y(z_{\text{EndT}})| \tag{5.25}$$

The idea behind this quantity comes from the other Long track reconstruction algorithm, the Matching: the y position of a real Long track at the end of the SciFi tracker, once obtained by extrapolating the input track through the magnet using the parameterisation, and once calculated from the determined yz trajectory, match well. This is possible because the VELO detector with its pixel sensors provides two-dimensional measurements and hence has a superior y-coordinate resolution compared to the SciFi tracker's stereo layers. Extrapolating the yz trajectory of the input track to the end of the SciFi tracker is, therefore, sufficiently reliable and provides a powerful variable for

fake track discrimination. Similarly, the D_x^{match} variable

$$D_x^{\text{match}} := |x_{\text{mag}} - x(z_{\text{EndT}}) - \frac{\mathrm{d}x}{\mathrm{d}z}\Big|_{z_{\text{EndT}}} (z_{\text{mag}} - z_{\text{EndT}})| \tag{5.26}$$

quantifies the match of the x coordinate at the magnet kink position, which is a natural quality-control variable within the Optical Model method used for pattern recognition. The slopes t_x and t_y, and the track's momentum have little direct discrimination power. However, they provide useful information via correlations with the other input variables in the same way they were useful for all previously used parameterisations. Preselection cuts are applied to variables to remove obvious fake track contributions before the training such that the neural networks only model the decision boundary in regions without efficient one-dimensional separation. The classifier responses are shown in Figure 5.17. The networks separate real from fake tracks well and correctly model the decision boundary, as seen from the comparison between the training and the test sample. A Kolmogorov-Smirnov (KS) test yields distribution consistency probabilities of 99.7% for the true and 62.7% for the fake tracks with VELO tracks as input. The corresponding probabilities with Upstream tracks as input to the algorithm are 81.5% and 26.4%. A cut on the MLP's response now controls the fake track fraction and the track reconstruction efficiency. If VELO tracks are given as input to the Forward tracking, Long tracks with a response smaller than 0.14 are rejected. The cut has a true track efficiency of 97% and removes 77% of fake tracks. It is chosen relatively loose because true tracks rejected here are lost for physics data analysis. In contrast, fake tracks can still be rejected later by the Kalman filter or, lastly, by the data analyst. If the input track momentum is available, the cut value is 0.1 with a true track efficiency of 99% and a fake track rejection of 90%. The evaluation of the network for a given



a: VELO tracks as input.

b: Upstream tracks as input.

Figure 5.17: Output of the MLPs for the training (bar) and test (data point with error) sample. The distribution of true tracks is green with triangular markers, and the distribution of the fake tracks is red with circular markers.

track candidate is fast; the calculation of the response takes, on average, 0.5% of the Forward tracking's execution time, which is why it was decided not to make use of vectorisation here.

Even though zero or one Long track candidate per input track are the most frequent cases, the algorithm allows for more than one SciFi extension per input track. The only reason for this is a peculiarity mostly[1] observed with electron pairs created in photon conversions. The angle between the two electrons can be so small that the VELO detector and accordingly the VELO track reconstruction cannot resolve them, *i.e.* they share a VELO track. The magnet then splits the pair, and two valid SciFi extensions can be found for a single VELO track[2]. Therefore, multiple track candidates per VELO track are accepted if their response value is close to the highest scoring SciFi extension. This concludes the full candidate selection. The result is a container of Long tracks (`PrForwardTracks` in Figure 5.1), with at least ten SciFi hits each and reasonable confidence that they are actual tracks.

Duplicate Removal Finally, the container of Long tracks is scanned for track duplication, which happens if two input tracks have similar track state vectors. First, the tracks are sorted in increasing order according to their x position at z_{EndT}. Duplicates are then efficiently found by checking for each track if any following track in the container has an x position within 50 mm of the track under consideration. If that is the case, whether the y position is within 100 mm and whether the two tracks share more than half of their SciFi hits is checked. Only the one with the highest score from the fake track rejection is kept from the tracks fulfilling these criteria.

5.2.9 Finding UT Hits

The final change to the Long tracks is the addition of UT hits. This is not required but improves the momentum resolution of the Long tracks achieved later by the Kalman filter, and also helps to filter out fake tracks after a successful track fit. Since access to the UT hits and the geometry of the UT is necessary, but executing the hit addition is optional in the Long-track reconstruction, it is implemented as a Gaudi tool handling its own data dependencies. This way, the code can be used by other algorithms too, *e.g.* by the Matching, and the UT hits addition can be switched off centrally in case there is a problem with the detector it depends on, without jeopardising the Long track reconstruction.

The distance from the end of the VELO detector to the UT is around 1.5 m with a relatively weak magnetic field integral in between. The Long track is therefore extrapolated into each UT layer by a straight line from the VELO track segment with

[1] For other processes, this happens extremely rarely. But it was also seen for *e.g.* $\phi \to K^+K^-$.

[2] The current implementation of the VeloUT algorithm does not allow for more than one UT-hit set per VELO track and therefore cannot reconstruct these pairs. It is planned to change this behaviour, at least for the VeloUT tracking in filter mode.

a small correction for the x position depending on the Long track momentum. UT hits around the predicted track-layer crossing are collected, and their deviation from the predicted track position is projected to a reference plane in the centre of the UT detector, resembling the projection from Equation 5.2. Starting from the hit with the smallest projected deviation from the track, a group of at least three hits from three or more different UT layers is formed by adding the hit with the next smallest deviation. A group of hits is fitted with a straight line, and the hit with the worst χ^2 is removed if the group consists of too many hits, similar to the fit procedure described in Section 5.2.6. If multiple groups of UT hits match the Long track, the one with the lowest total χ^2 and most hits is selected. The tool adds four UT hits to a Long track on average. The UT hit efficiency on Long tracks from a B meson is above 98% with a purity of 99.4%. For general Long tracks, the hit efficiency and purity are a bit lower at 96.5% and 97.8%, respectively.

5.2.10 Output

The Forward tracking outputs the found tracks in an SoA container (`Pr::Long::Tracks` in Figure 5.1) that is further processed by the Kalman filter. The index of the Long track's ancestor, *i.e.* a VELO or Upstream track, is stored along with the number of VELO hits, UT hits, and SciFi hits the Long track is made of. For the hits themselves, a unique identifier is stored together with their indices in the hit container, which makes it easy for the Kalman filter algorithm to access hit information directly. Finally, the track state vector at the end of the SciFi tracker is stored, closing the Long track finding.

5.3 Reconstruction Performance

The performance of the track finding is evaluated using the metrics defined in Section 4.4.

5.3.1 Physics Performance

If not stated otherwise, all efficiencies given in this section refer to the Long-track finding efficiency of the Forward tracking defined by Equation 4.9. This implicitly includes the input-track finding efficiency in the numerator because the track segment upstream of the magnet is a prerequisite to finding the corresponding Long track. In practice, this has hardly any effect as finding the VELO-track segment of a Long track is more than 99% efficient (see Appendix A.4.1).

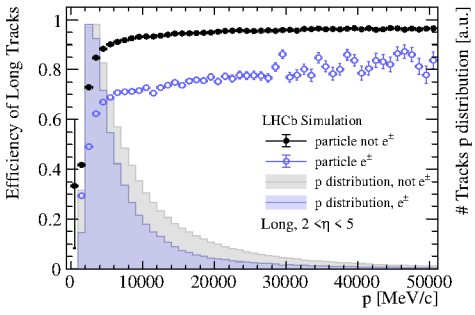
The Long-track reconstruction efficiency of the full sequences can be found in Table 4.1 and Ref. [91].

Efficiencies and Fake Track Fractions First, the efficiency in dependence on various kinematic variables of the Forward tracking using VELO tracks as input is presented

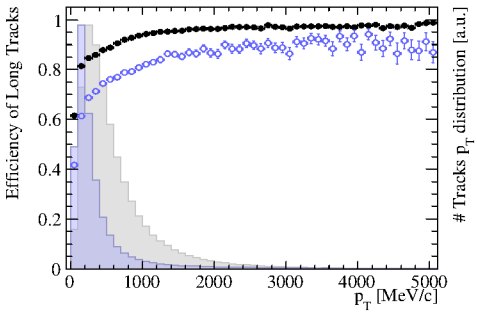
in Figures 5.18 and 5.19. The former figure shows the general Long track category without kinematic requirements except the track being in LHCb's acceptance. The efficiency drops rapidly at low momenta, which is due to three effects: the hit search window exhibits a similar turn-on curve at low momenta as shown in Figure 5.4, the hit projection during the Hough transform is less precise in the low momentum regime, and in general low-momentum tracks undergo more multiple scattering and are thus harder to reconstruct. The efficiency as a function of pseudorapidity and azimuthal angle of the tracks gives some insight into the distribution of material within the detector. Around $\eta = 4.3$, a slight drop is visible, which comes from the conic shape of the beam pipe within the RICH1 detector; after RICH1, the beam pipe has an edge and is smaller again but still with a conic shape, *i.e.* increasing radius throughout the rest of the LHCb detector (see Ref. [23]). Tracks around $\eta = 4.3$ and $\eta > 4.5$ cross beam-pipe material, and its support structures (see the picture in Figure A.13), and thus undergo more scattering or interact hadronically, visible as a slight drop in the η distribution of the reconstructible particles and the corresponding efficiencies as reconstructible particles that scattered in the material are harder to find. The same effects can be seen in the azimuthal angle where the efficiency drop due to the beam-pipe material is visible at $\phi = 0, \pm\pi$, which corresponds to the x-z plane. The low-η region also exhibits a lower track finding efficiency. One reason is that the parameterisations used during the pattern recognition perform worse for steep low-momentum tracks, which is why the effect is not as pronounced when looking at Long tracks from a B meson, *e.g.* in Figure 5.19c, which have a harder momentum spectrum. The other reason is that tracks with a large t_x leave low-resolution hits in the SciFi tracker, which leads to lower quality tracks and lower track finding efficiency. The efficiency in dependence on the number of primary vertices, given in Figure 5.18e, is reasonably flat, showing that the track finding works well also for busy events.

The efficiencies for electrons are shown separately. They are significantly lower than the efficiencies of other particles as they emit bremsstrahlung in addition to undergoing multiple scattering when traversing material. The emission of bremsstrahlung photons upstream of the magnet makes it more difficult to follow and find the trajectory in the SciFi tracker. All parameterisations created for the pattern recognition explicitly exclude electrons for this reason, and currently, no measures are taken to recover electron tracks.

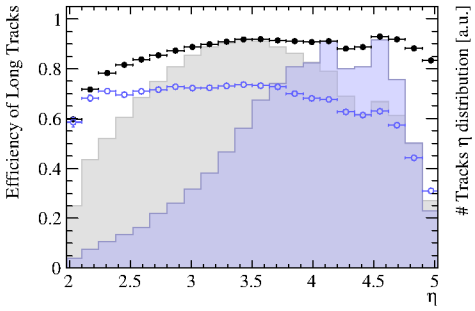
The fake track fraction is shown in Figure 5.20. The most prominent feature in the Figures 5.20a and 5.20b is the high fraction of ghosts at low momenta; low p_{T} in particular. They are typically built from VELO tracks with large pseudorapidity and small t_y, which extrapolate well into the SciFi tracker region with the highest hit occupancy. The abundance of hits in this region then makes it likely to find an extension by chance. Moreover, the same reason that reduces the track finding efficiency at high η increases the fake track fraction in that region, as visible in Figure 5.20c; for a particle scattering off the beam pipe before the UT detector, it is likely to find



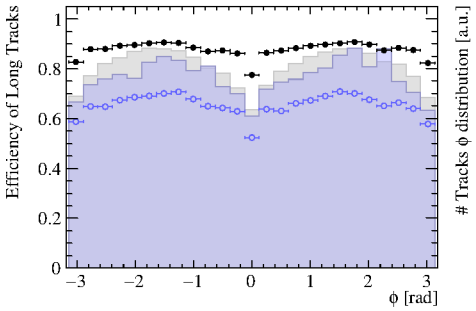
a: Efficiency *vs.* momentum.



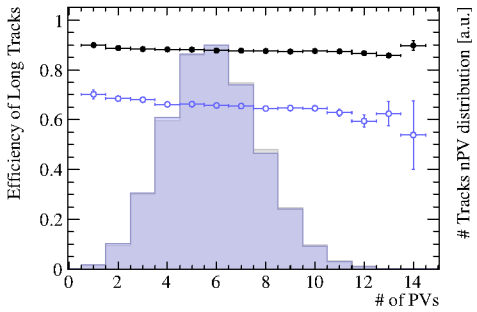
b: Efficiency *vs.* transverse momentum.



c: Efficiency *vs.* pseudorapidity.

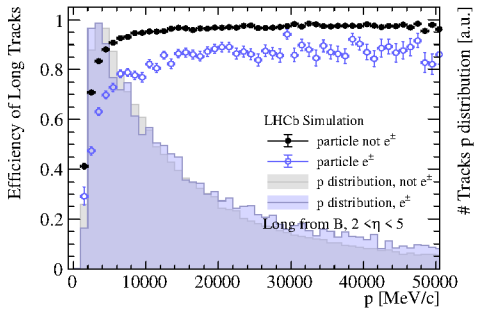


d: Efficiency *vs.* azimuthal angle.

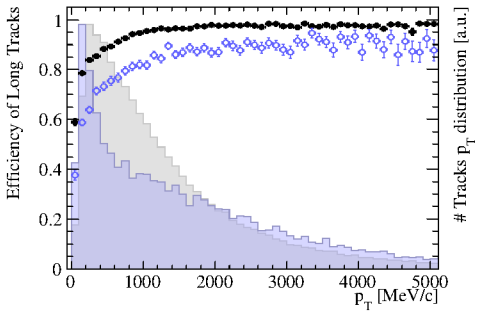


e: Efficiency *vs.* number of primary vertices.

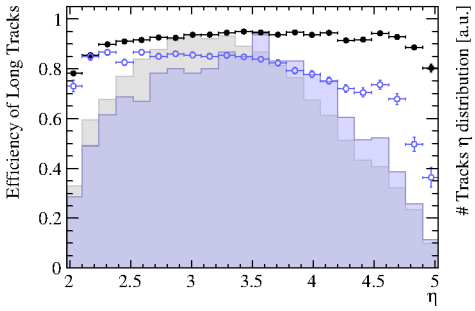
Figure 5.18: Track finding efficiency of the Forward tracking algorithm with VELO input tracks for simulated Long tracks with $2 < \eta < 5$ in dependence on various kinematic variables and the number of primary vertices. The variables' distributions and the efficiencies are given separately for reconstructible electrons and other reconstructible particles; electrons in blue with empty-circle markers and other particles in black with full-circle markers. The true, simulated variable values are used for the distributions.



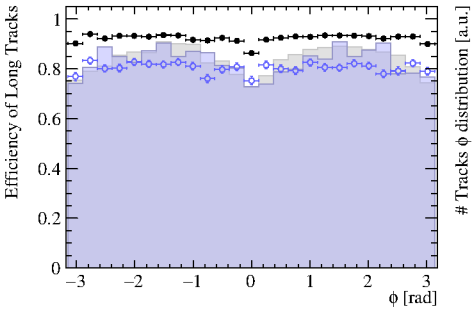
a: Efficiency *vs.* momentum.



b: Efficiency *vs.* transverse momentum.



c: Efficiency *vs.* pseudorapidity.

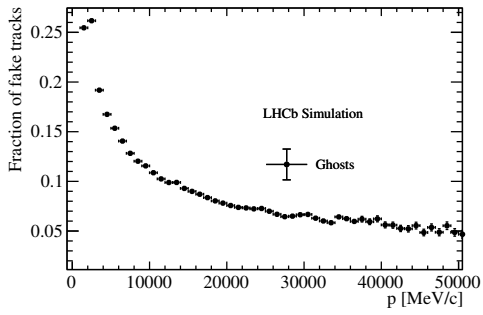


d: Efficiency *vs.* azimuthal angle.

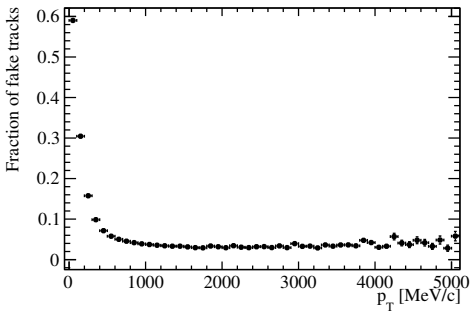
Figure 5.19: Track finding efficiency of the Forward tracking algorithm with VELO input tracks for simulated Long tracks with $2 < \eta < 5$ originating from a B meson in dependence on various kinematic variables and the number of primary vertices. The distributions of the variables and the efficiencies are given separately for reconstructible electrons and other reconstructible particles; electrons in blue with empty-circle markers and other particles in black with full-circle markers. The true, simulated variable values are used for the distributions.

a wrong set of SciFi hits, mimicking a low-momentum track because for these the requirements during the pattern recognition are the weakest. Similarly, VELO tracks left by unstable particles decaying between the VELO tracker and the SciFi stations are often paired with the SciFi hits produced by their decay products, *i.e.* secondaries and therefore real T tracks. Lastly and unsurprisingly, the fake track fraction rises with the number of primary vertices. The reason is that events with many primary vertices typically also exhibit many SciFi hits, which increases the probability to find random hit combinations that look like a Long track.

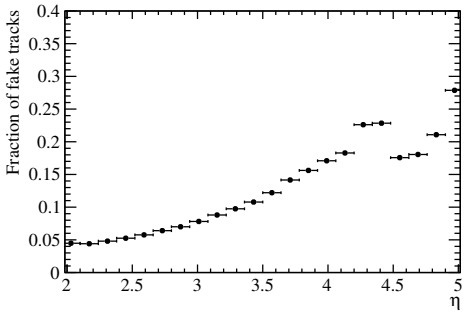
Integrated efficiency numbers and other physics performance metrics are given in Table 5.3. Comparing these to the objectives defined in Section 5.1 and Table 5.1 shows that the Forward tracking surpasses the performance estimated during the planning of the LHCb Upgrade. In particular, the fake track fraction is almost a factor of three



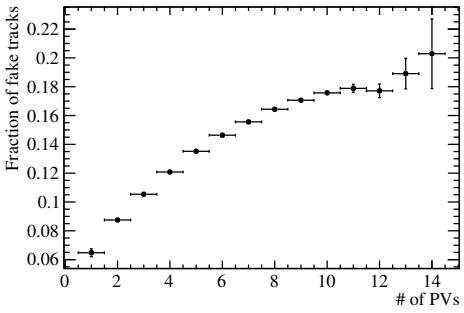
a: Fake track fraction *vs.* momentum.



b: Fake track fraction *vs.* transverse momentum.



c: Fake track fraction *vs.* pseudorapidity.



d: Fake track fraction *vs.* number of primary vertices.

Figure 5.20: Fake track fraction of the Forward tracking algorithm with VELO input tracks in dependence on kinematic variables and the number of primary vertices.

smaller[1] due to the precise track parameterisations and the neural network used for ghost rejection. The Run 1 performance can be approached if a comparable fake track fraction is accepted[2]. It is not reached, though, which is expected given that the algorithm is much more constrained by execution time and that the instantaneous luminosity is five times higher. The purity and hit efficiency show that the algorithm constructs clean tracks on average. Again electrons exhibit a worse track quality than other particles.

If the Forward tracking is fed with UT-filtered VELO tracks (see Upstream track type in Section 4.3), similar efficiencies are reached at half the fake track fraction as shown in Table 5.4 (see Appendix A.4.2 for efficiency plots). This is possible because VELO tracks for which no UT hits are found likely also do not have an extension in the SciFi tracker, and rejecting them upfront using the VeloUT algorithm avoids

[1] Removing the global event cut adds roughly 5% to the fake track fraction at comparable efficiencies.
[2] For example, by cutting at 0.01 on the neural network response and keeping candidates that have a response within 0.4 compared to the best candidate.

finding fake extensions and improves the computational performance. Suppose the UT-filtered VELO track has a momentum estimate. In that case, the hit search window

Table 5.3: Integrated Forward tracking efficiencies with VELO input tracks obtained from 5000 simulated $B_s^0 \to \phi\phi$ events. Efficiencies for Long electron tracks from B are obtained from 5000 simulated $B^0 \to K^{*0}e^+e^-$ events. Efficiencies for tracks from D mesons are obtained on 5000 simulated $D^+ \to K_S^0\pi^+$ events. A category not marked with $e^\pm$ implies that electrons are excluded. The uncertainties are below 10^{-3}% and thus are not shown here.

Category	Efficiency ε[%]	Purity f_{pure}[%]	Hit Efficiency ϵ_{Hit}[%]
Long	87.97	99.26	98.45
Long $e^\pm$	64.12	97.55	98.03
Long $p > 5\,\text{GeV}/c$	93.41	99.38	98.83
Long from D	88.76	99.39	98.66
Long $p > 5\,\text{GeV}/c$ from D	94.23	99.54	99.04
Long from B	92.59	99.45	98.86
Long $p > 5\,\text{GeV}/c$ from B	95.77	99.52	99.06
Long $e^\pm$ from B	83.52	98.76	98.83
Long $e^\pm$ $p > 5\,\text{GeV}/c$ from B	86.45	98.81	98.93
Fake track fraction r_{fake}	14.00%		

Table 5.4: Integrated Forward tracking efficiencies with UT-filtered VELO tracks as input. Here, the same samples as for the training of the fake track rejection classifier (Section 5.2.8) are used to avoid a specific bug related to UT hit decoding present in the samples otherwise used for efficiency determination. This means no radiation damage to the SciFi tracker is simulated here. The uncertainties are below 10^{-3}% and thus are not shown.

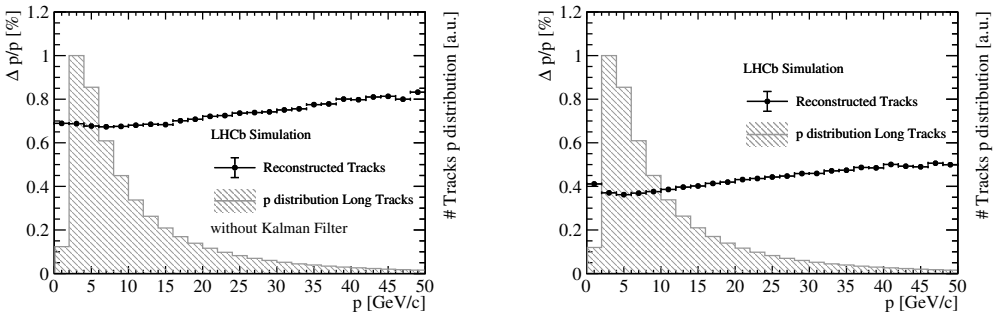
Category	ε[%]	p-dependent search window ε[%]
Long	87.49	85.83
Long $e^\pm$	61.39	58.80
Long $p > 5\,\text{GeV}/c$	92.78	91.60
Long from B	90.57	89.10
Long $p > 5\,\text{GeV}/c$ from B	94.40	93.25
Long $e^\pm$ from B	75.06	71.36
Long $e^\pm$ $p > 5\,\text{GeV}/c$ from B	81.29	77.38
Fake track fraction r_{fake}	7.56%	6.60%

Table 5.5: Integrated Forward tracking efficiencies with Upstream tracks from the default VeloUT algorithm as input. Here, the same samples as for training the fake track rejection classifier (Section 5.2.8) are used to avoid a specific bug related to UT hit decoding present in the samples otherwise used for efficiency determination. This means no radiation damage to the SciFi tracker simulated here. The uncertainties are below 10^{-3}% and thus are not shown.

Category	ε[%]	p-dependent search window ε[%]
Long	54.86	54.21
Long $e^{\pm}$	16.99	16.06
Long $p > 5\,\mathrm{GeV}/c$	68.48	68.06
Long from B	71.85	71.10
Long $p > 5\,\mathrm{GeV}/c$ from B	83.01	82.42
Long $e^{\pm}$ from B	55.48	53.15
Long $e^{\pm}$ $p > 5\,\mathrm{GeV}/c$ from B	67.08	64.44
Fake track fraction r_{fake}	2.08%	1.67%

in the Forward tracking can be configured to take advantage of that knowledge as explained in Section 5.2.3. In the current configuration, the momentum-dependent search window decreases the track-finding efficiency by roughly 1.5% and reduces the fake track fraction by 1%. Electron efficiencies especially suffer in this scenario because their momentum is reduced by bremsstrahlung, such that they move outside of the search window. Also, electrons created in photon conversions that share a VELO track are penalised here because the VeloUT algorithm only allows one Upstream track to be created per VELO track. Similarly, nominal Upstream tracks further reduce the fake track fraction but dramatically lower the efficiencies, as shown in Table 5.5. This is because the nominal VeloUT algorithm was tuned to reconstruct higher momentum tracks from B decays with $p > 2.5\,\mathrm{GeV}/c$ and $p_{\mathrm{T}} > 500\,\mathrm{MeV}/c$ in a CPU-HLT1 scenario. The VeloUT filter mode can be viewed as the HLT2-tuning of the VeloUT algorithm.

Momentum Resolution The momentum resolution of the tracks found by the Forward tracking is shown in Figure 5.21. The fast momentum estimate calculated using Equation 5.23 performs reasonably well, reaching a resolution of $\Delta p/p \simeq 0.8\%$ over the whole momentum spectrum. This momentum, however, is merely used as an input to the full track fit performed using a Kalman filter, which provides the final estimate of the track parameters. The Kalman filter's momentum result is shown in Figure 5.21b, which is slightly better than the rough estimate of the momentum-resolution limit due to multiple scattering made at the end of Section 4.2.2. The effect of multiple scattering on the resolution is evident at momenta below $5\,\mathrm{GeV}/c$. For high momenta, the resolution gradually worsens because the track's bending becomes less significant



a: The momentum is estimated by Equation 5.23. No Kalman filter is applied.

b: With Kalman filter applied.

Figure 5.21: Track resolution as a function of momentum for tracks reconstructed by the Forward tracking.

compared to the single-hit resolution. The resolutions in Figure 5.21b are obtained by fitting Long tracks that have UT hits added. Yet, UT hits are not a requirement to form a Long track. Without them, the momentum resolution is 10% worse in the low-momentum regime and 25% for high-momentum tracks. The pronounced resolution degradation for high momenta can be explained by the single-hit resolution of the UT's silicon sensors, which is two to four times better than the single-hit resolution of the SciFi tracker. Hence, because the UT is located in an area with a non-negligible magnetic field, the UT hits add valuable information about the track momentum, especially for high-momentum particles.

5.3.2 Computational Performance

Besides providing good physics performance, the other factor driving the work on the Forward tracking and, in general, the whole reconstruction is the computational performance in terms of event throughput or execution time of the algorithms. The relevant metrics are defined in Section 4.4.2. A lot of effort was put into optimising the computational performance of the Forward tracking using SIMD instructions where appropriate, avoiding unnecessary work where possible and applying modern implementations of search and sort algorithms where needed. The numbers in this section only highlight the performance of the Forward tracking. It has to be noted, though, that eventually, only the throughput of the whole HLT2 sequence is relevant for the success of the real-time analysis strategy. And the performance, computational and physics, of the individual components strongly depends on the performance of the components before them in the sequence. This is to say that a lightning-fast Forward tracking can be counter-productive if it finds many fake tracks, which slow down the Kalman filter later in the sequence. Conversely, investing some more time executing a Forward tracking at a low ghost rate can pay off with a faster sequence in total as the

Kalman filter has to do less work.

The computational performance has already been revealed in Figure 4.5a: the Forward tracking takes about 11% of the reconstruction sequence, which runs at 388 evts/s on a single reference node. This translates to a Forward tracking throughput of around 3527 evts/s. Compared to the reference number from June 2019 from Section 5.1, this shows a speedup by more than a factor of 3.5 and now fits into the computing budget of HLT2, fulfilling the objective set in the beginning. Around 6% of the throughput increase must be credited with optimisation improvements by more recent compiler versions. The rest was achieved by the optimisations mentioned above. The effect of explicitly using SIMD instructions is evaluated by running the scalar version[1] of the component. The scalar version shows a 57.5% higher execution time than the vectorised version, demonstrating that the vectorisation is responsible for a speed-up factor of almost 3. Enabling[2] SIMD instructions from AVX512 but with 256-bit vectors, the Forward tracking gains another 5% in throughput. This is, however, not foreseen to be enabled for the application running in the event-filter farm because when using AVX512 instructions, the CPU dynamically scales down its frequency to a level where the whole HLT2 sequence becomes slower[3]. Aside from that, most CPUs in the HLT computing farm are not AVX512-compatible.

Known Bottlenecks The most time-consuming part of the Forward tracking is the hit projection during the Hough transform (`projectHitsToHoughSpace`), taking around 44% of the component's execution time, out of which 31% is spent on the projection of the x hits alone. There are two connected problems here which slow down the tight loop: there is an instruction-dependency chain after the bin number for a hit is calculated, which prevents efficient use of the CPU's pipeline, and there are memory operations necessary to fill the projected positions into the Hough histogram, *e.g.* scattering values from a SIMD vector to non-sequential locations in memory, which introduce some latency worsening the effect of the instruction dependencies. With AVX512, this could be implemented more efficiently but would nonetheless throttle the projection loop's performance. No way around this particular bottleneck was found. However, the problem can be mitigated by reducing the number of SciFi hits that are projected, *e.g.* by using the momentum-dependent search window when Upstream tracks or UT-filtered VELO tracks are given as input (see Table 5.6).

The second bottleneck is the selection of track candidates using the x hits taking 29% of the execution time (`selectXCandidates`). This is driven by the quality of candidates coming out of the Hough transform; to be able to reconstruct low-momentum tracks, the threshold scan over the Hough histogram takes into account up to three

[1] This is configured manually in the SIMDWrapper because otherwise, vectorisation using at least SSE4.2 is enabled by default. Also, this does not keep the compiler from auto-vectorising the code, *i.e.* only the explicit vectorisation is switched off.

[2] This was tested on an Intel Xeon Silver 4214 CPU.

[3] Which shows that the fraction of code (efficiently) using SIMD instructions is small.

Table 5.6: Average execution times per thread on the reference node for the Forward tracking with different input tracks and with or without the momentum-dependent search window.

Input tracks	$\langle t_{\text{VeloUT}} \rangle_{\text{ref}}$ [ms]	p search window?	$\langle t_{\text{Forward}} \rangle_{\text{ref}}$ [ms]	rel. Speed-up [%]
VELO	-	-	11.0	0
UT-filtered	0.4	no	9.3	16
		yes	4.0	64
Upstream	0.2	no	3.3	70
		yes	1.0	91

neighbouring bins with low thresholds which almost always pick up rogue hits that have to be cleaned out again.

In general, most of the algorithm's execution time is spent on reconstructing lower-momentum tracks or trying to find a SciFi extension to VELO tracks, which do not have one. Both are often done in the second loop, which itself can be seen as a bottleneck of the algorithm.

UT-filtered VELO tracks and Upstream Tracks Since typically half of the VELO tracks do not have a SciFi extension, it makes sense performance-wise to filter them upfront using the VeloUT tracking. The efficiencies of the Forward tracking using UT-filtered VELO tracks or Upstream tracks as input have been presented in the previous section. However, the primary purpose of these running modes is to speed up the Long track reconstruction, which only works because the VeloUT tracking is much faster than the Forward tracking. Execution times per event and thread are shown in Table 5.6. Reducing the number of input tracks by UT-filtering yields a timing improvement of 16% without unacceptably lowering the track finding efficiencies. Furthermore, the full reconstruction sequence profits from the reduced number of ghosts, the throughput effect of which was not quantified in this work. If the momentum-dependent search window is also used, the Forward tracking gains more than a factor of two in throughput, partially solving the hit-projection bottleneck mentioned above. The best computational performance is achieved when using Upstream tracks as input, but it also has the worst physics performance. The default VeloUT tracking configuration used here rejects low-momentum tracks and thus keeps most of the work away from the Forward tracking.

5.4 Comparison to Neural-Network-based Approach

LHCb's track-reconstruction design is modular, and the baseline track finding creates the most important track-segment types, *i.e.* VELO and T tracks, independently of each other (see Figure 4.3). This enables the possibility to create Long tracks by simply

combining VELO and T tracks using a machine learning classifier. This is done by the Matching algorithm, to which the author has contributed in the course of the natural overlap with the Forward tracking. In particular, the parameterisations to effectively model the passage through the magnetic field used in both algorithms are canonically similar.

5.4.1 The Matching Algorithm

The Matching takes VELO and T tracks as inputs with the objective to combine pairs of them into Long tracks. The basic idea is to define variables that quantify the level of agreement, *i.e.* a match, between the track segment found in the VELO detector and the segment reconstructed in the SciFi tracker, and evaluate a machine learning classifier on them to obtain a combined estimate of the matching quality.

The algorithm is organised as a loop over the VELO track state vectors with a nested loop over the state vectors of all T tracks, such that all possible combinations are covered. To reject combinations that are obviously wrong, the y component of both the T track and the VELO track are extrapolated by a straight line to a position behind the SciFi tracker at $z_{\text{match}} = 10\,\text{m}$, referred to as $y^{\text{T}}_{\text{straight}}$ and $y^{\text{V}}_{\text{straight}}$, respectively. Because the bending in the y-z plane is small, only T tracks with $y^{\text{T}}_{\text{straight}} < y^{\text{V}}_{\text{straight}} + 250\,\text{mm}$ are accepted. Subsequently, the matching variables used as input to a neural network are calculated. Analogously to what was used during the fake track rejection in the Forward tracking, the matching defines the x distance at the magnet kick position within the Optical Model method using Equation 5.26. Yet, a slightly different[1] parameterisation to determine the value of z_{mag} is used because, for T tracks, the slope difference is defined as $\Delta t_x^{\text{match}} := t_x^{\text{EndT}} - t_x$, which is different to Δt_x defined in the Forward tracking. The magnet kick position in the Matching is given by

$$z_{\text{mag}}^{\text{match}} = c_0 + c_2|x(z_{\text{EndT}})| + c_3 t_x^2 + |\Delta t_x^{\text{match}}|(c_1 + c_4|\Delta t_x^{\text{match}}|) \tag{5.27}$$

with coefficients listed in Table A.11. Similarly, the y distance is defined by Equation 5.25 but with a different correction because the z position where the two segments are matched is z_{match}. The correction is parameterised by

$$y_{\text{corr}}^{\text{match}} = t_y(c_0|\Delta t_x^{\text{match}}|^2 + c_1|\Delta t_y^{\text{match}}|^2) \tag{5.28}$$

with coefficients listed in Table A.12. This correction is less precise than the one given by Equation 5.24, however, it was found that the matching neural network performs better using this simple expression. The two matching distances are then used to define

[1] A different parameterisation also accounts for the fact that T tracks are not exactly the same as SciFi hit sets found by the Forward tracking because T tracks also cover tracks from long-lived particles and secondaries. It has been tested whether using the same parameterisation works. The answer is yes, but using a dedicated one works better.

Table 5.7: Input variables of the Matching MLP.

Variable	Preselection
χ^2_{match}	< 15
D_x	$< 250\,\text{mm}$
D_y	$< 250\,\text{mm}$
$\lvert\Delta t_x^{\text{match}}\rvert$	< 1.5
$\lvert\Delta t_y^{\text{match}}\rvert$	< 0.15
$t_x^2 + t_y^2$	

a combined quality measure mimicking a χ^2

$$\chi^2_{\text{match}} = \frac{D_x^2}{\delta x^2 + t_{\delta x}^2 |\Delta t_x^{\text{match}}|^2} + \frac{D_y^2}{\delta y^2 + t_{\delta y}^2 (t_x^2 + t_y^2)} + \frac{|\Delta t_y^{\text{match}}|^2}{\text{var}(t_y^{\text{EndT}})}$$

where $\text{var}(t_y^{\text{EndT}})$ is the variance of the T track's y slope at the end of the SciFi tracker, $\delta x = 8\,\text{mm}$ with $t_{\delta x} = 80\,\text{mm}$ configures the x uncertainty, and $\delta y = 6\,\text{mm}$ with $t_{\delta y} = 300\,\text{mm}$ configuring the y uncertainty.

The architecture of the neural network used to classify the track-segment pairs is almost the same as the one used for fake track rejection in the Forward tracking and trained on the same data samples (see Section 5.2.8); the MLP has two hidden layers with $N + 2$ and N neurons, respectively, where N is the number of input variables, listed in Table 5.7. It has been tested by the author whether adding the difference between the T-track momentum and the Long track momentum to the input variables improves the classification. No significant impact on the classification was achieved, which is why this variable is not used. Electrons are banned from the training data set as including them severely lowers the classifier's performance for other particles for the same reasons they were excluded when building the parameterisations for the Forward tracking. The input variables' distributions are shown in Figure A.10. The MLP's response is plotted in Figure 5.22 for the test and training sample. The neural network models the data well with a KS-test probability of 31.3% for the good pairs and 85.9% for the wrong pairs. Like the Forward tracking's ghost rejection, a cut on the classifier response controls the fake track fraction and track finding efficiency. The current working point chosen for the Matching is at a response of 0.215, where 84% of the wrong pairs are rejected while keeping 97% of the correct ones. The track-finding efficiency and fake track fraction are discussed in the next section. The pair with the highest response is selected. Other pairs can also be accepted if their response deviates less than 0.1 from the highest one, which is helpful for the reconstruction of conversion

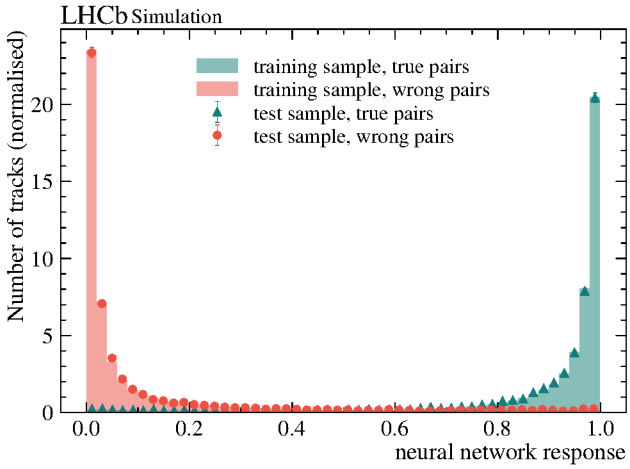


Figure 5.22: Output of the Matching MLP for the training (bar) and test (data point with error) sample. The distribution of true pairs is green with triangular markers, and the distribution of the wrong pairs is red with circular markers.

electron pairs as explained in Section 5.2.8.

Subsequently, the selected VELO-T track pairs are promoted to Long tracks; their momentum is estimated using Equation 5.23 but with a parameterisation different[1] to the one applied in the Forwrad tracking, and they are stored in the same data structure as the output of the Forward tracking. The final step is adding the optional UT hits using the tool discussed in Section 5.2.9.

The algorithm heavily uses SIMD instructions, processing eight T tracks per VELO track in parallel when AVX2 is enabled. Hence, the generic custom implementation of the neural network also used in the Forward tracking pays off here.

5.4.2 Performance Comparison

The first apparent difference between the two Long tracking algorithms is the Matching's striking simplicity compared to the Forward tracking. This is, however, only the case because the complexity is hidden in the algorithm finding the SciFi track segments. Therefore, a fair comparison of the computational performance must include the timing of the Hybrid Seeding algorithm. Adding the numbers given in Table 5.8 shows that the Matching finds Long tracks slightly faster than the Forward tracking. One reason is the efficient usage of SIMD parallelism in the Matching. But also, the Hybrid Seeding contributes to good computational performance because it does not re-use hits

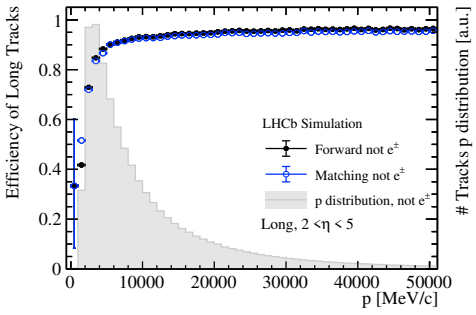
[1]The parameterisation is not the same because the author of this thesis had not had the time yet to unify all parameterisations used in the track reconstruction. In the future, all algorithms shall use the same parameterisations if possible. The priority in this specific case is low as the Kalman filter renders away any differences due to the parameterisations anyway.

Table 5.8: Average execution times per thread on the reference node for the Hybrid Seeding, Matching and the Forward tracking.

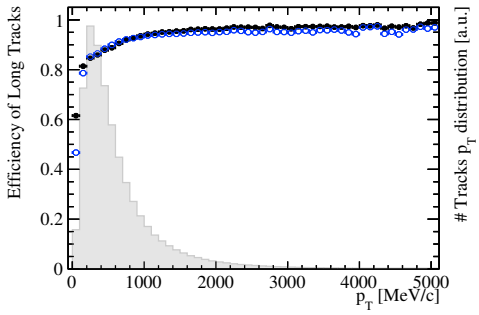
Component i	$\langle t_i \rangle_{\text{ref}}$ [ms]
Seeding	10.1
Matching	0.6
Forward	11.0

already associated with a found track. This makes the track-finding problem easier along the way. The efficiency comparison is shown in Figures 5.23 and A.11. The Forward tracking achieves slightly higher efficiencies in the high-momentum region, at low transverse momentum and high pseudorapidity, while the Matching performs slightly better in the low pseudorapidity region and around the beam pipe at $\phi = 0, \pm\pi$. The latter reflects a fundamental difference between the two approaches: the Forward tracking uses parameterisations that do not take multiple scattering into account to follow the VELO track into the SciFi tracker, while the Matching's MLP is trained on reconstructed T tracks and thus also models the effect of scattering before the SciFi tracker. Overall, the track finding efficiency of the two Long tracking algorithms is very similar, which is precisely the reason the fast reconstruction sequence avoids running both on the same data. The resulting Long track reconstruction efficiency is given in Table 4.1. Regarding the fake track fraction, however, the Forward tracking outperforms the Matching significantly, as shown in Figure 5.24. One reason is the different approach to using the information by the respective algorithm. While the Matching directly considers the T track state vector at the end of the SciFi tracker, the Forward tracking builds the entire SciFi track segment from scratch under the assumption that it is connected to the VELO track. This way, each SciFi hit is matched to the VELO track, instead of matching a whole track segment. Moreover, matching an entire track segment suffers from the fact that many T tracks are associated with secondary particles, which do not form a Long track. This increases the probability of matching up unrelated segments given that roughly 50% of the VELO tracks are not part of a Long track.

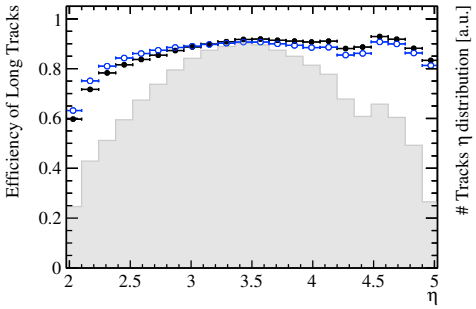
Last but not least, the electron track finding efficiencies of both algorithms are compared in Figures 5.25 and A.12. Because electrons were explicitly excluded from the training of the Matching's MLP it is no surprise that the Forward tracking performs better in finding electrons.



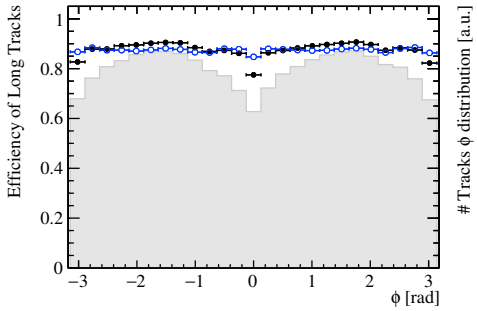
a: Efficiency *vs.* momentum.



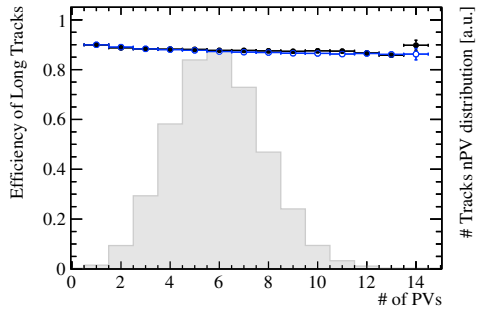
b: Efficiency *vs.* transverse momentum.



c: Efficiency *vs.* pseudorapidity.

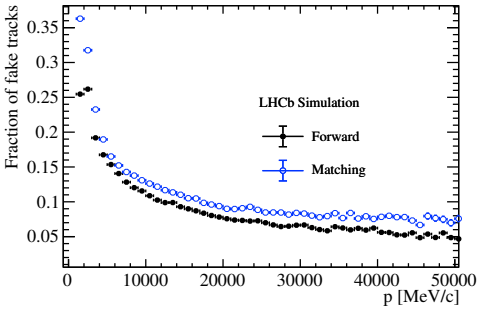


d: Efficiency *vs.* azimuthal angle.

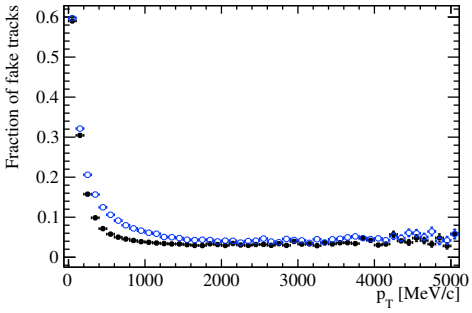


e: Efficiency *vs.* number of primary vertices.

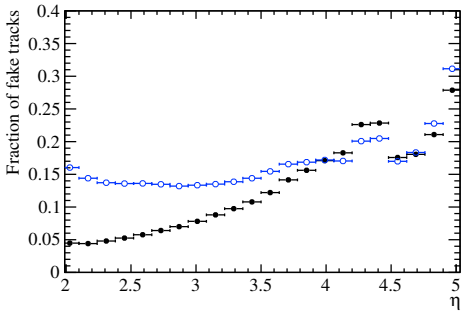
Figure 5.23: Track finding efficiency of the Forward tracking algorithm with VELO input tracks compared to the Matching algorithm for simulated Long tracks with $2 < \eta < 5$ in dependence on various kinematic variables and the number of primary vertices. The Matching is shown in empty blue circles and the Forward tracking is in full black circles.



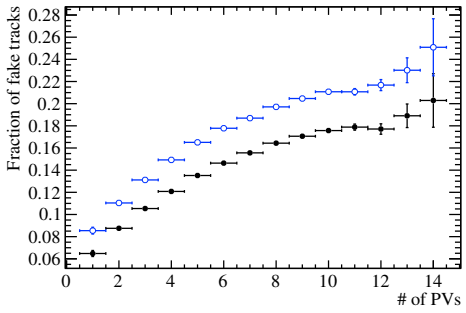
a: Fake track fraction *vs.* momentum.



b: Fake track fraction *vs.* transverse momentum.

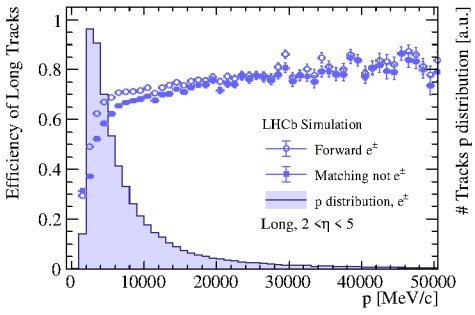


c: Fake track fraction *vs.* pseudorapidity.

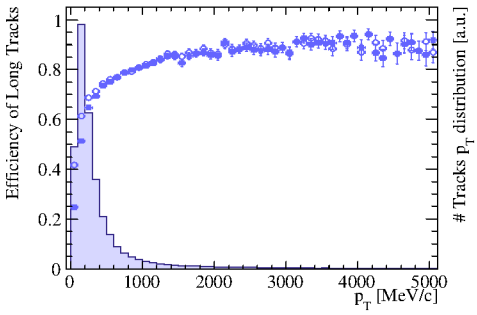


d: Fake track fraction *vs.* number of primary vertices.

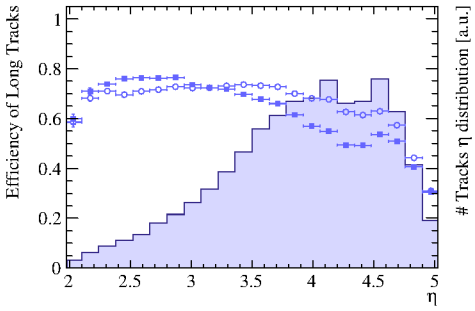
Figure 5.24: Fake track fraction of the Forward tracking algorithm with VELO input tracks compared to the Matching algorithm in dependence on kinematic variables and the number of primary vertices. The Matching is shown in empty blue circles and the Forward tracking is in full black circles.



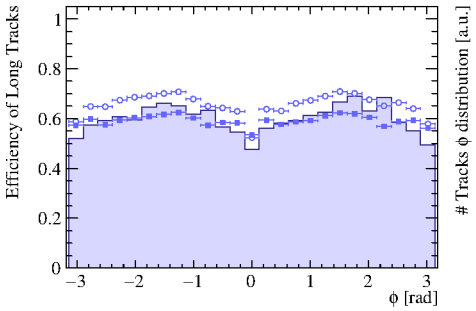
a: Efficiency *vs.* momentum.



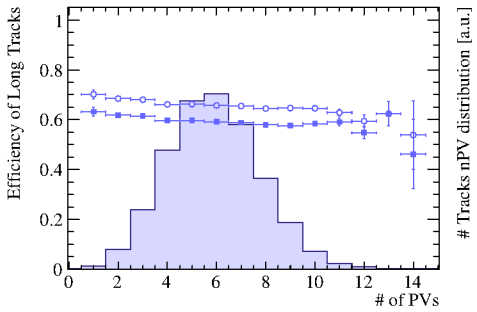
b: Efficiency *vs.* transverse momentum.



c: Efficiency *vs.* pseudorapidity.



d: Efficiency *vs.* azimuthal angle.



e: Efficiency *vs.* primary vertices.

Figure 5.25: Electron track finding efficiency of the Forward tracking algorithm with VELO input tracks compared to the Matching algorithm for simulated Long tracks with $2 < \eta < 5$ in dependence on various kinematic variables and the number of primary vertices. The Matching is shown in full blue rectangles, and the Forward tracking is in empty blue circles.

5.5 Summary and Future Prospects

The Forward tracking is one of the two algorithms finding Long tracks within LHCb's event reconstruction sequence running in HLT2. In simulation, it reaches track finding efficiencies of more than 90% for particles from heavy-flavour decays at an event throughput of 3527 evts/s, which fits into the computing budget of HLT2 for Run 3. This is made possible by extensively optimising the computational performance of the algorithm while keeping the physics performance high. As is typical for software-related work, the Forward tracking and, generally, track reconstruction is part of an endless cycle of planning, analysing, designing, implementing, testing and maintaining. The cycle is reignited every time new circumstances evolve: new or missing hardware - be it detector components like the absent UT or developments in computing architectures -, bugs that only occur when running in production over trillions of collision events, misalignment of the trackers; the list of caveats and problems under real-world conditions is long, some of which are discussed in Chapter 6. The development cycle has been triggered many times for the Forward tracking, which is why it is as complex of an algorithm as it is now. But for the same reason, it is also a flexible track reconstruction algorithm. If requirements on the track reconstruction change, *e.g.* during LHC operating as an ion-ion collider, the Forward tracking can be easily configured to serve a different momentum regime, to cope with larger fake track fractions due to high occupancies, or to be more resilient against poor detector conditions. In hindsight, it was a wise decision to keep the usage of UT hits optional in the HLT2 Forward tracking as the UT was still missing during the first data-taking in 2022. For 2023 when the UT is being commissioned, the Forward tracking is ready to use it, be it by only adding matching UT hits or by taking UT-filtered tracks as input. Nevertheless, to fill performance gaps of the reconstruction efficiencies and further optimise the computational performance more work is needed.

On the physics performance side, the electron finding efficiency leaves a lot to be desired, particularly with tests of lepton-flavour universality being a focus of the flavour-physics community, in addition to angular analyses of $b \to s\ell^+\ell^-$ transitions, LHCb's results of which being partially thwarted by the lack of electron statistics. A possible way towards better electron efficiencies could be to implement a dedicated electron reconstruction. It might run as a final stage of the Long track reconstruction over residual VELO tracks, and SciFi hits, similarly to how the Forward tracking is used in HLT2's fast reconstruction sequence. To account for the different kinematics of the electrons due to bremsstrahlung, all parameterisations could be re-calculated and trained on samples only containing electrons. To ensure these electrons survive the track fit, a specialised track parameter estimator like a Gaussian-sum filter [68, 103] could be applied.

On the computational performance side, the prospects are manifold, also in their scale of necessary changes. With a relatively small effort, the Forward tracking could gain around 5% more throughput by changing the binary search implementation to

using static B-trees [104]. A bit more involved, but with all necessary (software) components already in place, is reorganising the fast track reconstruction sequence to utilise the VeloUT algorithm as a pre-filter for the Long track finding. Instead of running the Matching first and subsequently the Forward tracking on residual hits and tracks, the Forward tracking could run first on UT-filtered VELO tracks using the momentum-dependent search window. As shown in Table 5.4, this would already find the bulk of reconstructible Long tracks at a low fake track fraction and almost three times the speed of the nominal Forward tracking or the Hybrid Seeding together with the Matching. Subsequently, the Hybrid Seeding would run on residual SciFi hits, followed by the Matching using residual VELO tracks. It is not expected that more computational improvements of the Forward tracking are necessary for the current data taking. However, if throughput gains of several factors again become necessary, *e.g.* for LHCb Upgrade 2 at the high-luminosity LHC, a completely new implementation will be necessary. It seems unlikely that CPU-only tracking can keep up with the increasing performance needs. A future HLT2 Forward tracking must use more accelerators like GPUs, FPGAs and generally heterogeneous computing chips to keep up with the performance demands within a real-time application.

The industry is already going in that direction, *e.g.* with Intel's oneAPI programming model aiming to unify interfaces to accelerators like GPUs, FPGAs and AI-specific circuits. Suppose this allows to easily develop the dispatching of a single algorithm's tasks to the accelerators, like performing the Hough-like transform of the Forward tracking on an FPGA. In that case, performing the full event reconstruction in a single high-level trigger stage might be within reach. Furthermore, the advances in machine learning combined with specialised hardware to evaluate the data-driven models pose opportunities to efficiently cope with the increasing complexity of track reconstruction in high-rate and high-resolution particle detectors. So far, modern machine learning techniques are hardly utilised in LHCb's track reconstruction software; geometric deep learning approaches like graph neural networks might be useful in the future to improve the physics and computational performance of the software. The fast and specialised developments in the computing domain and the accompanying software-maintenance burden will also make it more necessary in the future to share event-reconstruction software between large collaborations like ATLAS, CMS, ALICE and LHCb, and their potential successors at future colliders. The foundation is already there with *A Common Tracking Software*, and LHCb should contribute to and profit more from this in the future.

6 Commissioning of the Long Track Reconstruction

The track reconstruction developments for LHCb's real-time trigger described in the previous chapters have been proven to work well on simulated pp-collision samples, in which the detector is perfectly aligned and works as foreseen by its design. In reality, a complex apparatus like the LHCb experiment has to be carefully tuned during commissioning to approach the desired performance. This involves the sub-detector experts preparing and testing the apparatus for data taking and the trigger experts preparing and testing the data recording. For LHCb in Run 3, with its purely software-based trigger system performing the whole event reconstruction in real-time, it is crucial to validate before nominal operations that the software analysing the events works. This is particularly the case for track reconstruction, as it constitutes the bridge between the detector signals and the physics objects the data analysts want to study. This point cannot be stressed enough: LHCb needs to maintain an operational tracking system and real-time alignment at all times as in HLT1 it de facto picks up the task of the removed hardware trigger.

Therefore, this chapter gives an overview of the author's work as part of the commissioning team, particularly regarding the HLT2 Long track reconstruction. Information about the commissioning of HLT1 can be found elsewhere [105]. The computational performance of the track reconstruction on data has not yet been studied because in 2022 LHCb operated at a fifth of the foreseen instantaneous luminosity, and the event throughput of HLT2 is not a concern until it is proven that the new detector can deliver the data quality necessary to perform physics analyses.

6.1 The First Steps Toward Run 3 Data

The LHC started to ramp up its operations for Run 3 in April 2022 with experiment cavern closure and the first beam at the end of that month. After the commissioning of the beam, stable beams and the first collisions at the injection energy of 450 GeV were provided from the end of May on. The commissioning of LHCb's sub-detectors was ongoing - with all of them in place except for the UT- testing the data acquisition systems and the sub-detectors' time alignment, when HLT1 reached the first milestone by successfully triggering on the activity in the ECAL. The tracking detectors, however, were not yet able to take data useful for track reconstruction. Although progress was made, this had not changed when the LHC started regularly delivering stable beams

at the nominal beam energy of 6.8 TeV at the beginning of July. Then finally, the first success on the track reconstruction side was achieved toward the end of July when the VELO detector was able to see the first VELO tracks and vertices, albeit still in its safe, opened position with the modules separated by about 50 mm. Also, the Hybrid Seeding found its first tracks; however, the number was low and thus not distinguishable from fake tracks. Beginning to mid of August, the sub-detectors started to operate more and more together, and the SciFi detector took data with the calorimeters and the muon stations, and later with the VELO detector additionally. From a track reconstruction perspective, however, this data was merely valuable to mechanically test the reconstruction machinery because both the VELO and the SciFi detector had no stable coarse time alignment yet, and several detector modules were excluded from data taking because of problems with them. The coarse time alignment ensures that the sub-detector is synchronised with a specific isolated bunch in the LHC such that each recorded event can be assigned to a unique bunch-crossing identifier. Furthermore, also no fine time alignment was yet ready for the SciFi tracker, drastically reducing its hit efficiency to a level where hardly any tracks could be found. The fine time alignment optimises the electronics' integration window such that hits left by primary tracks are recorded with high efficiency. Without a reasonable set of tracks also no spatial alignment of the tracking system is possible (*cf.* Section 3.2.3). For the track reconstruction to use a spatial alignment, the conditions database storing information about the exact spatial positions of detector modules and the mapping of readout links must be available, which was only the case around the end of August. Then at the end of September, after a cooling incident had tied up the LHC for more than three weeks, the individual sub-detectors approached an operational state in which the track reconstruction should have been able to find its first real, confirmed Long tracks.

6.2 The Hunt for the First Mass Peak

In the beginning, when an essentially new particle physics detector with new reconstruction software is being commissioned, the best way to show that it works is to measure the decay of a particle which is known to be produced abundantly in pp collisions and which has some properties that make it easy to distinguish from the background. Throughout September and the beginning of October 2022, hardly any data with all tracking detectors operating together was available. The first[1] decay reconstruction attempts were therefore searching for the decay $J/\psi \to \mu^+\mu^-$, which in principle can be cleanly reconstructed because the muons leave clear signatures in the muon stations. Hence, the Hybrid Seeding and a simple muon track segment reconstruction were used to reconstruct "SeedMuon tracks". The disadvantage of this decay is that it is relatively

[1]The ECAL by then already had its first success by reconstructing the decay $\pi^0 \to \gamma\gamma$. The calorimeters are not a new sub-detector, though, and usually sit behind the track reconstruction in the processing chain.

rare. But with a J/ψ cross-section in LHCb's acceptance [106] of around 17 µb and a commissioning instantaneous luminosity of roughly $300\,\mu b^{-1}s^{-1}$, half an hour of data taking should have been enough to be able to reconstruct $\mathcal{O}(1000)$ $J/\psi \to \mu^+\mu^-$ candidates. Yet, the invariant mass distribution of the two reconstructed muons was perfectly consistent with combinatorial background. Later it turned out that the muon stations' tiles were not correctly ordered in the geometry used by the software. Therefore, no reasonable reconstruction was possible.

Then, towards the end of October 2022, things finally started to come together: the VELO and the SciFi detector had been sufficiently tuned, *i.e.* the number of hits they saw was of the order of magnitude of what was expected, and they were both running stably except for some modules with known problems. Everything looked ready to reconstruct some Long tracks. Without success in reconstructing $J/\psi \to \mu^+\mu^-$ because of problems with the muon stations, the decay of interest now was $K^0_S \to \pi^+\pi^-$. The K^0_S particle is produced in almost every pp collision and lives long enough to travel $\mathcal{O}(100\,\text{mm})$ in LHCb's laboratory reference frame. A clean selection is, therefore, possible by requiring the z position of the K^0_S vertex (z_{vtx}) to be displaced from the interaction region. Furthermore, the two pions are required to have a small distance of closest approach (DOCA), which ensures that a reasonable vertex can be formed. These variables only use information from the tracking system and are chosen because of their simplicity; after all, the detector is not aligned yet, and the selection can not be reasonably tuned. The full selection is given in Table 6.1. The implicit momentum cuts from the reconstruction ($p > 1.5\,\text{GeV}/c$ and $p_{\text{T}} > 50\,\text{MeV}/c$) are not listed. All tracks are reconstructed using the Forward tracking as described in Section 5.2, yet without adding UT hits. The output tracks of the Forward tracking are fitted using a Kalman filter, but no track quality cuts are applied. Moreover, using the PID system to identify the pions is unnecessary since the geometric cuts already reduce the background sufficiently. This was checked with simulated minimum bias pp collisions at $\sqrt{s} = 14\,\text{TeV}$ and $\nu = 7.6$ ($\mu \simeq 5$) shown in Figure 6.1. The fit model is an exponential function for the combinatorial background and a Gaussian for the signal component. The fit results are given in Table 6.2. The background in data is expected to be a bit lower because of the lower number of visible collisions during the commissioning compared to the simulation.

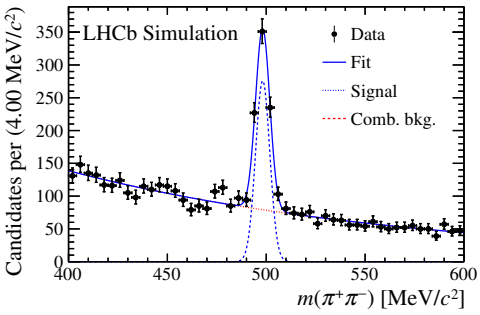
Table 6.1: Simple $K^0_S \to \pi^+\pi^-$ selection.

Variable	Selection
$\text{DOCA}(\pi^+, \pi^-)$	$< 0.3\,\text{mm}$
$z_{\text{vtx}}(K^0_S)$	$> 200\,\text{mm}$
$m(\pi^+\pi^-)$	$> 400\,\text{MeV}/c^2$
	$< 600\,\text{MeV}/c^2$

Table 6.2: Result of the fit shown in Figure 6.1. The last row gives the number of events processed by the reconstruction.

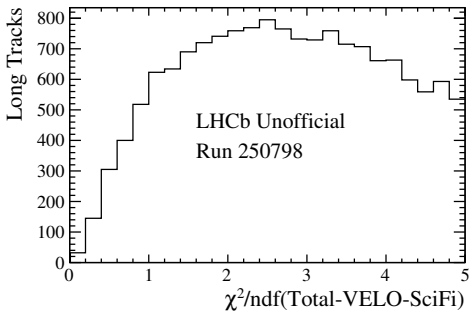
Parameter	Estimated Value
λ_{expo}	$(-5.7 \pm 0.3) \times 10^{-3} (\text{MeV}/c^2)^{-1}$
μ_{Gauss}	$(498.2 \pm 0.2)\,\text{MeV}/c^2$
σ_{Gauss}	$(3.6 \pm 0.2)\,\text{MeV}/c^2$
N_{sig}	620 ± 33
N_{bkg}	4138 ± 68
N_{events}	4800

Figure 6.1: Selected $K^0_S \to \pi^+\pi^-$ candidates using Long tracks from the Forward tracking with closed VELO after the Kalman filter.

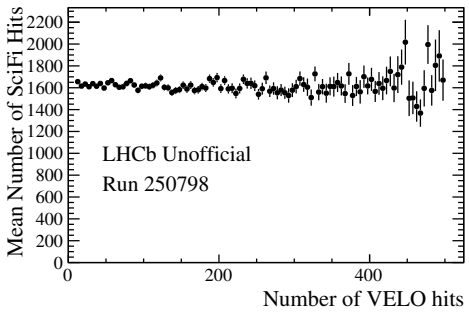


Although things looked promising on the sub-detector side, very few Long tracks were reconstructed from the data, on par with what was expected from random fake tracks. Thus hardly any candidates for the decay $K^0_S \to \pi^+\pi^-$ were found, let alone a visible mass peak. The crucial plots that lead to the underlying problem are shown in Figure 6.2. They are created using the HLT2 Forward tracking on raw detector data, passed through the first trigger stage without any selection but a pre-scale, and directly written to the buffer to be independent of the HLT1 commissioning. In a spatially and timely misaligned detector, the mode of the χ^2/ndf distribution is expected to be shifted towards higher values with a more pronounced tail to the right. The reason is the underestimated uncertainty on the hit positions and the fewer degrees of freedom due to the lower single-hit efficiency of the detector. However, the distribution shown in Figure 6.2a is consistent with random tracks. The χ^2/ndf shown there is calculated by subtracting the χ^2 values of the VELO and SciFi track segments from the one of the entire Long track, and the same for the degrees of freedom. The resulting χ^2/ndf gives information about how well the VELO track segment and the SciFi track segment match each other. Hence, the SciFi hit extension found by the Forward tracking is not a good match to the VELO track on average. Because the confidence in the tracking algorithms after years of studying them in simulation is relatively high, the findings of Figure 6.2a lead to Figure 6.2b, which shows the number of SciFi hits in dependence on the number of VELO hits. A linear correlation between these quantities should be visible if both sub-detectors see the same *pp*-collision event. The flat distribution hints at a problem with the coarse time alignment between the VELO and the SciFi detector. And indeed, subsequently, it was found that the VELO was off by four bunch crossings compared to the other sub-detectors.

On the 27th of October 2022, the sub-detector experts had fixed the relative time alignment. To verify that the problem indeed was fixed, the data taken afterwards was immediately analysed, the result of which is shown in Figure 6.3. Given that the



a: Matching χ^2/ndf between VELO and SciFi track segments.

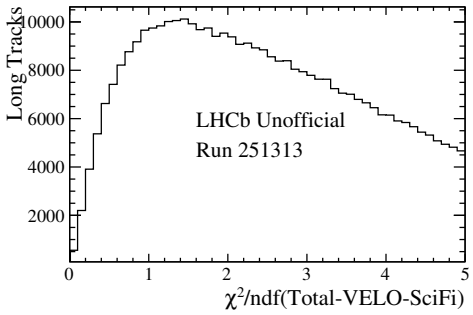


b: Profile of the number of SciFi hits versus the number of VELO hits.

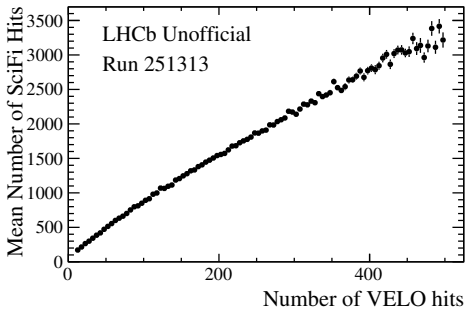
Figure 6.2: Distributions used to commission the track reconstruction. VELO and SciFi detector see different bunch crossings here.

detector was not aligned, the χ^2/ndf distribution in Figure 6.3a now looks as expected, with its peak slightly above a value of 1 and a pronounced tail to higher values. Also, the correlation between the number of SciFi and VELO hits is now clearly visible in Figure 6.3b.

Then, using the Forward tracking for track reconstruction and the same selection as used to reconstruct the decay $K^0_S \to \pi^+\pi^-$ in simulation for Figure 6.1, finally the desired mass peak is visible in Figure 6.4, marking a major milestone in the development of the LHCb experiment for Run 3, and a personal milestone for the author, as it proves that the developed Forward tracking algorithm finds real tracks under real-world conditions. After this data was taken, the VELO detector was put into its nominal position, *i.e.* closed around the interaction region. The number of tracks reconstructed in this position is much higher as the coverage of the high pseudorapidity region



a: Matching χ^2/ndf between VELO and SciFi track segments.



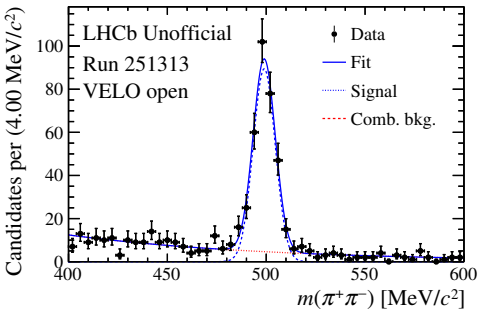
b: Profile of the number of SciFi hits versus the number of VELO hits.

Figure 6.3: Distributions used to commission the track reconstruction. VELO and SciFi detector see the same bunch crossings here.

Table 6.3: Result of the fit shown in Figure 6.4. The last row gives the number of events processed by the reconstruction.

Parameter	Estimated Value
λ_{expo}	$(-1.0 \pm 0.1) \times 10^{-2} (\text{MeV}/c^2)^{-1}$
μ_{Gauss}	$(499.0 \pm 0.4)\,\text{MeV}/c^2$
σ_{Gauss}	$(5.6 \pm 0.3)\,\text{MeV}/c^2$
N_{sig}	313 ± 19
N_{bkg}	270 ± 18
N_{events}	3×10^6

Figure 6.4: Selected $K^0_S \to \pi^+\pi^-$ candidates using Long tracks from the Forward tracking with open VELO after the Kalman filter.

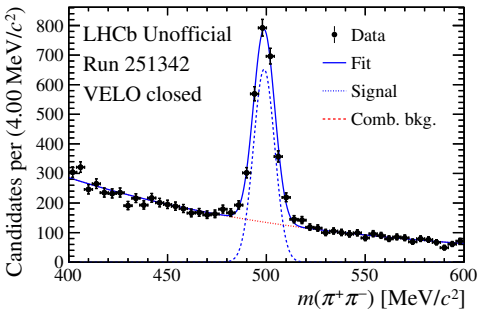


increases. As expected, this also increases the yield of reconstructed K^0_S candidates as shown in Figure 6.5. The width of the mass peak is larger compared to simulation which must be attributed to the misalignment of the detector. Even considering the lower instantanous luminosity, the number of events that had to be processed to reconstruct a similar amount of K^0_S decays as in simulation is three orders of magnitude higher, showing that without alignment the detector performance was nowhere near its design. However, the fact that still a significant number of K^0_S decays were reconstructed shows the robustness of the Forward tracking even with its default configuration.

Table 6.4: Result of the fit shown in Figure 6.5. The last row gives the number of events processed by the reconstruction.

Parameter	Estimated Value
λ_{expo}	$(-7.6 \pm 0.2) \times 10^{-3} (\text{MeV}/c^2)^{-1}$
μ_{Gauss}	$(498.8 \pm 0.2)\,\text{MeV}/c^2$
σ_{Gauss}	$(5.3 \pm 0.2)\,\text{MeV}/c^2$
N_{sig}	2185 ± 59
N_{bkg}	7372 ± 93
N_{events}	3×10^6

Figure 6.5: Selected $K^0_S \to \pi^+\pi^-$ candidates using Long tracks from the Forward tracking with closed VELO after the Kalman filter.



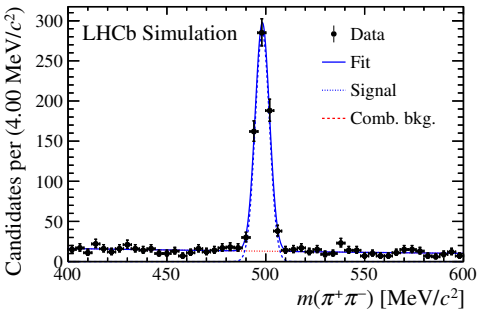
6.3 Different Configurations of the Long Track Reconstruction

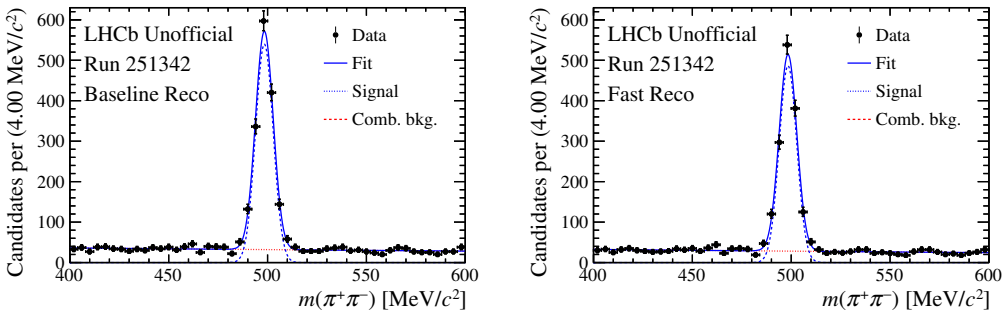
The alignment procedure for the SciFi detector depends on the minimisation of the residuals between a reconstructed track and the hits it is made of. This implies that as a first step, tracks must be reconstructed in a misaligned detector, and it is essential to find tracks in detector regions that are particularly misaligned so that these regions can be aligned. In the previous section, it has already been shown that the Forward tracking is robust enough to find real tracks in the misaligned detector. Nevertheless, it might be helpful to re-configure the track reconstruction sequence to increase the number of reconstructed true $K^0_S \to \pi^+\pi^-$ decays. In the track finding, the quantities most sensitive to the misalignment are the χ^2 cuts in the trajectory fits and the track classification neural network responses. Therefore, requirements on these quantities in the respective algorithms are relaxed as shown in Table A.13. Additionally, the required minimum number of hits on a track is lowered to account for the missed hits due to misalignment and the lower single-hit efficiency of the detector. It has to be noted that this configuration would severely increase the fake track fraction under nominal Run 3 conditions; for the commissioning with its lower instantaneous luminosity, this is not a concern. To test this configuration and the reconstruction sequences defined in Section 4.5, a preliminary alignment of the VELO detector is used, which makes it possible to select the K^0_S candidate with respect to reconstructed primary vertices instead of the simplistic selection from Table 6.1. The selection, which is more realistic to what would be used during nominal data taking, is given in Table 6.5 with its validation on simulation in Figure 6.6. The impact parameter (IP) is the smallest DOCA of the particle trajectory with any primary vertex (PV). The flight distance ($\mathrm{FD_{PV}}$) is the distance between the PV belonging to the particle and its decay vertex. Low-p_T tracks are excluded here because they are expected to be dominated by fake tracks (*cf.* Figure 5.20b). First, the baseline and the fast track reconstruction sequence

Table 6.5: $K^0_S \to \pi^+\pi^-$ selection using primary vertices.

Variable	Selection
$p_\mathrm{T}(\pi)$	$> 200\,\mathrm{MeV}/c$
$\mathrm{IP}(\pi)$	$> 0.4\,\mathrm{mm}$
$\mathrm{DOCA}(\pi^+,\pi^-)$	$< 0.3\,\mathrm{mm}$
$\mathrm{FD_{PV}}(K^0_S)$	$> 50\,\mathrm{mm}$
$m(\pi^+\pi^-)$	$> 400\,\mathrm{MeV}/c^2$
	$< 600\,\mathrm{MeV}/c^2$

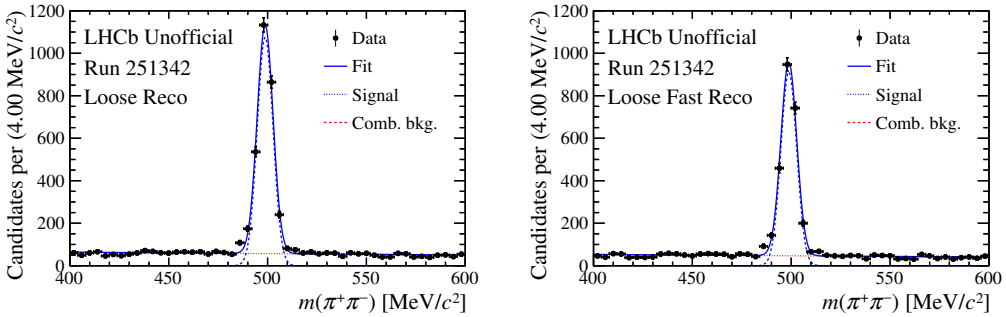
Figure 6.6: Selected $K^0_S \to \pi^+\pi^-$ candidates using Long tracks from the baseline sequence after the Kalman filter and the selection from Table 6.5.





a: Default baseline reconstruction.

b: Default fast reconstruction.



c: Loose baseline reconstruction.

d: Loose fast reconstruction.

Figure 6.7: Comparison of reconstructed and selected $K^0_S \to \pi^+\pi^-$ decays from the baseline and the fast reconstruction sequence with and without loosened Long tracking configuration as given in Table A.13. All candidates are built using Kalman filtered Long tracks.

are used to reconstruct the K^0_S candidates. The resulting mass peaks are shown in Figures 6.7a and 6.7b. With the baseline sequence, the fit estimates 1506 ± 42 K^0_S candidates and 1605 ± 43 random pion combinations. The fast sequence reconstructs tracks worth 1353 ± 40 K^0_S decays and 1415 ± 41 background candidates. From the reconstruction efficiencies for both sequences shown in Table 4.1 naively a smaller difference between the number of K^0_S candidates would be expected. It is, however, plausible that the simple Matching algorithm predominantly used in the fast sequence is less robust against a misaligned detector than the Forward tracking. This impression is confirmed by looking at the same mass peaks reconstructed using the loose tracking configuration in Figures 6.7c and 6.7d. Here the baseline and fast scenarios yield 2697 ± 56 and 2288 ± 51 K^0_S decays, respectively, with a background contribution of 2850 ± 57 and 2337 ± 52 candidates. Using the loose baseline configuration recovers a factor of 1.8 of K^0_S decays compared to the default. The signal-to-background ratio stays the same, indicating that not disproportionately more fake tracks are found. Yet the difference between the baseline and the fast reconstruction sequence increases

compared to the default, reflecting the value of the Forward tracking as a more complex algorithm with more configurable handles, compared[1] to the Matching.

6.4 Comparison of Data and Simulation

Fast-forward to the last weekend of November 2022, which marks the end of the data-taking period in 2022 and the start of the year's end shutdown. For the data taken this weekend, detector alignment constants for the tracking system are available, reflecting the best knowledge about the detector so far. This data is therefore used to compare to pp collisions simulated with conditions similar to the ones during data taking. This is $\mu \simeq 1.1$ at $\sqrt{s} = 13.6\,\text{TeV}$ and without a UT detector for data, and $\mu \simeq 1.5$ for simulation. The data is taken from the `NoBias` stream, *i.e.* no trigger selection is applied. The simulation was produced as a minimum bias sample. For better comparison, every event must have at least one reconstructed primary vertex. Tracks are reconstructed using the baseline sequence.

The χ^2/ndf of the Long tracks after the Kalman filter is plotted in Figure 6.8, which shows that the alignment of the detector does not yet sufficiently correct the residuals between the trajectory and its measurements. This is due to a convolution of multiple problems that need to be solved iteratively; to align particularly misaligned parts of the detector, first tracks must be found, which depends on the knowledge about the spatial position of the detector, which is what the alignment tries to improve. Initially, a coarse survey of the detector positions was made, which was found to have inaccuracies in some areas of the SciFi detector impeding the track reconstruction. Furthermore, the SciFi tracker has not yet reached its full hit efficiency and tracks thus often lack

[1] Arguably, the comparison is not fair because the Matching takes T tracks as input, the reconstruction of which has not been tuned here. However, as of writing this thesis, no loose configuration for the Hybrid Seeding is available.

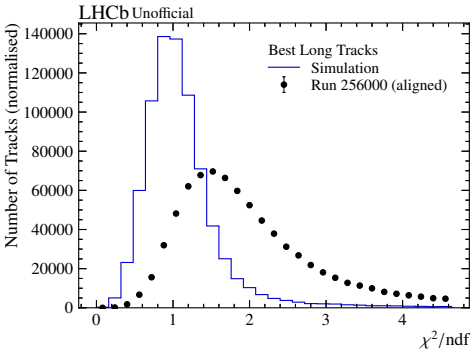
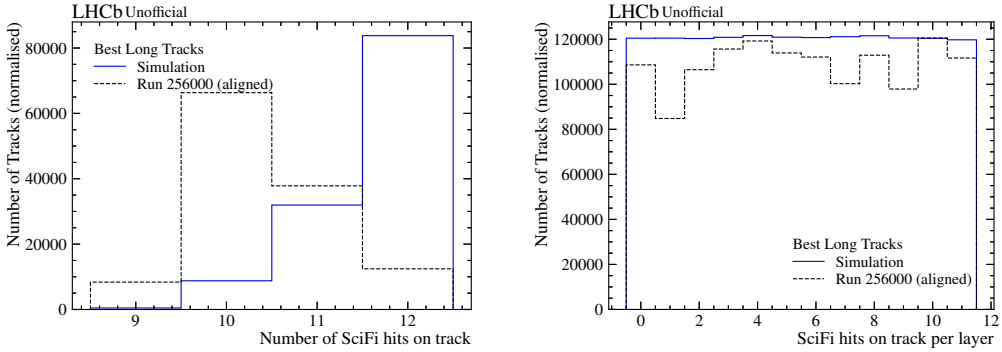


Figure 6.8: Comparison of the χ^2/ndf distribution between data and simulation. (Baseline reconstruction sequence)

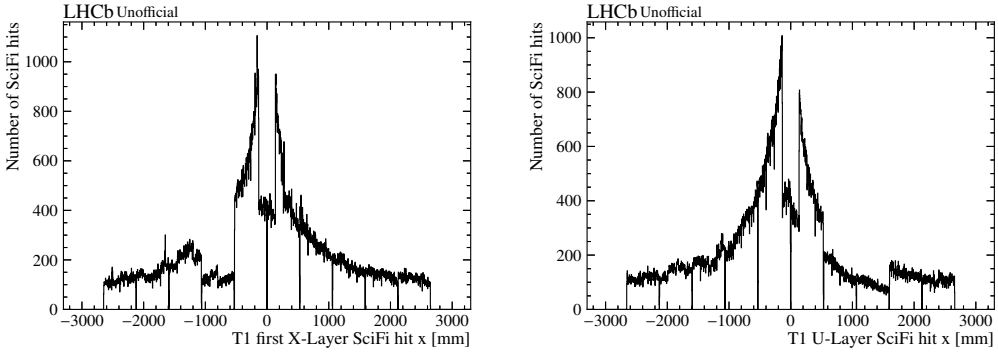


a: Number of SciFi hits on Long tracks.

b: Number of SciFi hits on a Long track per SciFi layer. Layers are numbered $0-11$ and ordered with increasing z positions.

Figure 6.9: Numbers of SciFi hits on Long tracks reconstructed with the baseline sequence for data (dashed black line) and simulation (solid blue line).

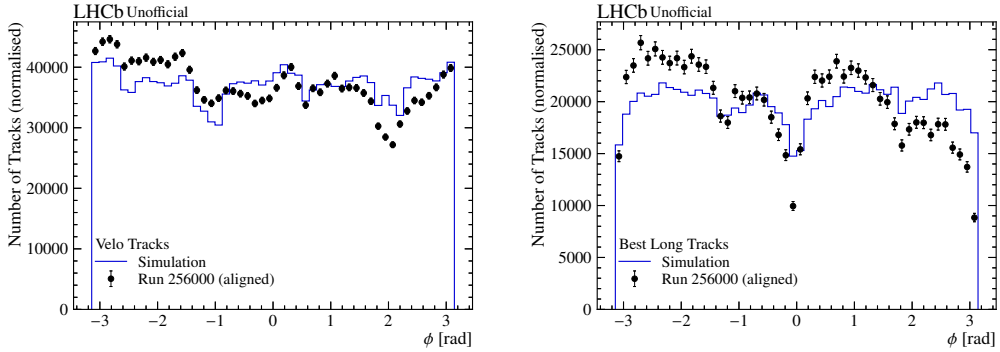
hits not only because they were not found due to misalignment but also because they were not detected in the first place. This is visible in the number of SciFi hits per Long track, shown in Figure 6.9a, which peaks at twelve hits in simulation, *i.e.* each layer provides a hit to the track, while in data most tracks have ten hits. If a module is inefficient and rarely provides a hit to a track, it can not be adequately aligned. Such inefficiency is, for example, observed in the number of SciFi hits on tracks from the first SciFi tracker station in Figure 6.9b. The SciFi hit distributions for the first two layers are shown in Figure 6.10. Then moreover, the VELO was not running with all its sensors operational, and the same misalignment argument holds. This results, for example, in underpopulated regions around $\phi \simeq 2\,\text{rad}$ as shown in Figure 6.11a. This inefficiency consequently is visible in the same distribution for the Long tracks,



a: First layer of the first SciFi station.

b: Second layer of the first SciFi station.

Figure 6.10: Distributions of SciFi hits' x positions from Run 256000.

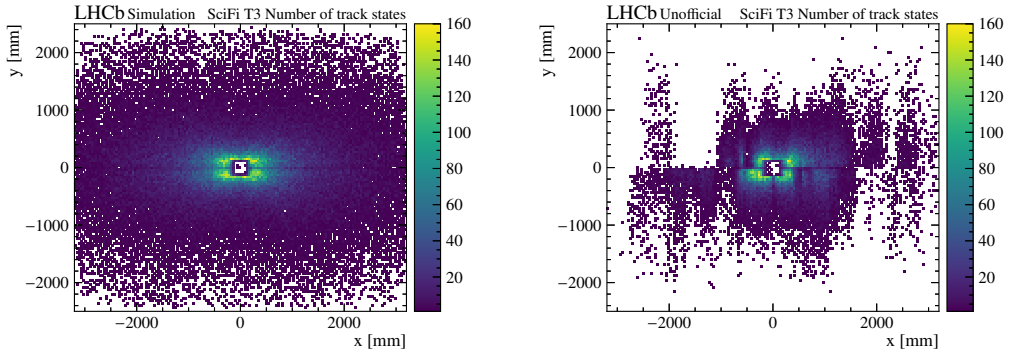


a: Azimuthal angle of VELO tracks.

b: Azimuthal angle of Long tracks.

Figure 6.11: Comparison between data taken during Run 256000 with aligned tracking system (black dots) and simulation (solid blue line) using the baseline reconstruction sequence.

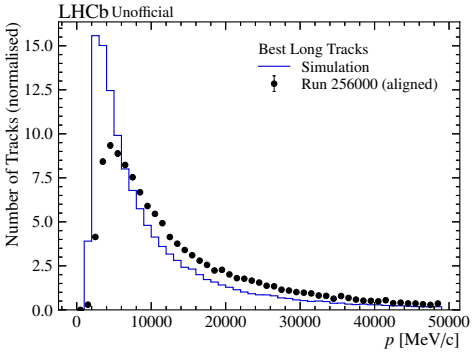
shown in Figure 6.11b, yet convoluted with inefficiencies of the SciFi tracker and its misalignment. The abovementioned problems can be seen in Figure 6.12b. The region around the beam pipe in the centre is populated well by tracks, while the edges of the detector hardly provide accurate hit information to build tracks. The second quadrant exhibits particularly few tracks with a large rectangular gap visible, hinting at a macroscopic module misalignment in this region, coinciding with the low-efficiency region observed in the azimuthal angle (Figure 6.11). Lastly, the momentum and the pseudorapidity distributions are compared to simulation in Figure 6.13. After the previous findings, the shapes shown here are straightforward to explain: the observed momentum spectrum is harder than expected from simulation because low-momentum tracks tend to be bent out of the central region around the beam pipe and thus traverse



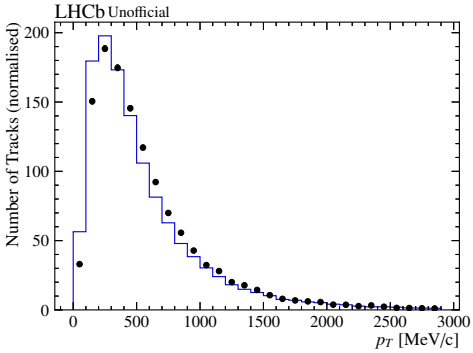
a: Simulation.

b: Data from Run 256000 with alignment applied.

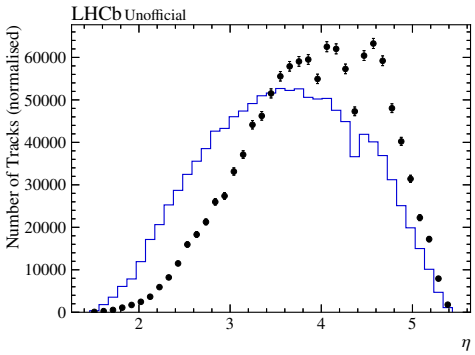
Figure 6.12: Distribution of reconstructed Long track positions in the last SciFi tracker station T3 using the baseline sequence.



a: Momentum distribution.



b: Transverse momentum distribution.



c: Pseudorapidity distribution.

Figure 6.13: Comparison between data taken during Run 256000 with aligned tracking system and simulation for basic kinematic variables

badly aligned SciFi tracker regions, penalising their reconstruction. Additionally, the low-momentum track finding depends heavily on the parameterisations in the pattern recognition, which assume an aligned detector. This causes a lower hit efficiency for low-momentum tracks from the reconstruction side, extra to the low hit efficiency on the detector side, and prevents these tracks from reaching the minimum number of hits. Steep tracks, *i.e.* tracks with small pseudorapidity, typically cross the SciFi tracker in its outer regions and are therefore less likely to be reconstructed in the current setup.

All of this shows that the commissioning of the detector and the alignment are incomplete. Further iterations of improving the detector performance and survey and initial track reconstruction followed by the alignment procedure are necessary to converge to a system that meets its design goals. For physics analysis, the data taken so far is merely helpful to mechanically test the selections and validate their output qualitatively. Selection efficiencies must be evaluated for quantitative results, relying on the simulation to model the data at least approximately correctly, which is not yet

the case. Also, calibration samples need to be ready and understood for data-driven efficiency evaluations, which is hardly possible under the current conditions.

6.5 Production of Charm Mesons

Although the detector is not yet in shape to perform robust physics measurements, this section summarises parts of the developments made to validate the entire data processing chain, from the new detector, over the online event reconstruction, to selections via trigger lines.

6.5.1 The Original Plan and its Status

Before the LHC's start for Run 3, LHCb made plans to measure various physics quantities as early as possible to prove that the new detector and trigger system can provide robust physics results. Among these early measurements is the determination of the prompt charm production cross-section in bins of p_{T} and rapidity y, using the decays $D^{*+} \to (D^0 \to K^-\pi^+)\pi^+$, $D^0 \to K^-\pi^-\pi^+\pi^+$, $D^+ \to K^-\pi^+\pi^+$, $D^+ \to K^-K^+\pi^+$, and $D_s^+ \to (\phi \to K^-K^+)\pi^+$. These measurements are particularly suited to validate the real-time analysis trigger and the new detector as the expected production rate of charm quarks at LHCb's interaction point is in the millions of pairs per second (*cf.* Figure 3.1) with many of them hadronising to one of the mesons as mentioned above. Hence, an integrated luminosity of $\mathcal{O}(10\,\mathrm{pb}^{-1})$ pp collisions is sufficient to conduct the measurement. Furthermore, the charm production cross-section was among the first measurements performed after the start of Run 2 in 2015 [107, 108], already using an early form of the real-time analysis strategy [37] to immediately analyse the data. The Run 2 measurements can thus be used as a reference to verify early Run 3 results. Besides the value for the LHCb experiment, the measurement is an important test of the predictions of perturbative quantum chromodynamics down to low values of the initial partons' momentum fraction accessible by LHCb's unique coverage of the high-rapidity region. With the LHC providing a centre-of-mass energy of $\sqrt{s} = 13.6\,\mathrm{TeV}$ for Run 3, the measurement also enters so-far unexplored energy territory. The plan is furthermore to extend the number of decays used to determine the cross-section. In particular, charm baryon decays[1] were not included in the measurement conducted in Run 2, as well as studies of the charm hadron production from double parton scattering [109, 110].

The entry point for every Run 3 measurement that directly uses the full capabilities of the real-time analysis trigger are the exclusive trigger lines selecting the decays of interest and persisting only the relevant reconstructed objects. The trigger lines must be prepared before the start of the data taking and thus rely on studies performed on samples simulating the expected experiment conditions. This poses a challenge

[1] $\Xi_c^+ \to pK^-\pi^+$, $\Xi_c^0 \to pK^-K^-\pi^+$, $\Lambda_c^+ \to pK^-\pi^+$

because the entire system, from the detector, over the data acquisition and event reconstruction software, to the software used by the data analysts to obtain the output of the trigger lines, is still dynamically changing. The writing of trigger lines for the early measurements, therefore, went along with actively developing and testing the necessary selection tools[1] for the framework, in which the author took an active part. By now, most selection tools the data analysts need are in place, and not only the trigger lines for early measurements are ready, but also more than a thousand exclusive trigger lines covering the whole physics programme of LHCb. Because 2022 was a commissioning year for LHCb, and the early data-taking period of 2023 will also be needed to iron out problems encountered in 2022 and to commission the UT, the measurement of the charm production cross-section has not been conducted yet. However, with the data taken in 2022, it was at least possible to verify that the processing chain, including the trigger lines, is, in principle, operational.

6.5.2 Trigger Lines

There are two design goals for the trigger lines selecting the charm mesons for the early measurement: they should be simple and loose enough to be able to select the desired decays even if the detector is not perfectly performing yet, and they should select the charm mesons over the whole p_{T} and y spectrum. The HLT2 selection cuts for $D^0 \to K^-\pi^+$ are listed in Table 6.6. All basic particles must be Long tracks, and

[1] These tools are called *ThOr functors* within LHCb. ThOr stands for "throughput oriented", which are functors designed to be able to use SIMD instructions to perform the selections with higher event throughput.

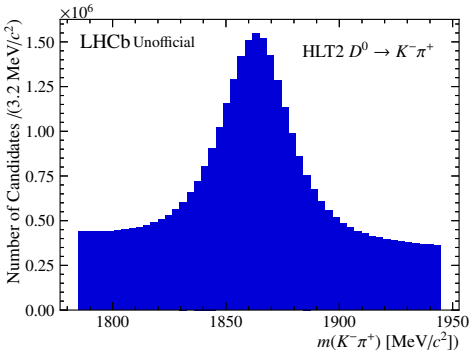
Table 6.6: HLT2 trigger selection for $D^0 \to K^-\pi^+$.

Particle	Selection
K^-, π^+	$p_{\mathrm{T}} > 250\,\mathrm{MeV}/c$
	$p > 2\,\mathrm{GeV}/c$
	$\chi^2_{\mathrm{IP}} > 16$
	$\mathrm{DOCA}(K^-, \pi^+) < 0.1\,\mathrm{mm}$
	$p_{\mathrm{T}}(K^-) + p_{\mathrm{T}}(\pi^+) > 1.5\,\mathrm{GeV}/c$
K^-	$\mathrm{DLL}_{K\pi} > 5$
π^+	$\mathrm{DLL}_{K\pi} < 5$
D^0	$m_{D^0} - 80 < m(K^-\pi^+) < m_{D^0} + 80\,\mathrm{MeV}/c^2$
	$\chi^2_{\mathrm{FD}} > 49$
	$\mathrm{DIRA} < 20\,\mathrm{mrad}$
	$\chi^2_{\mathrm{vtx}} < 10$

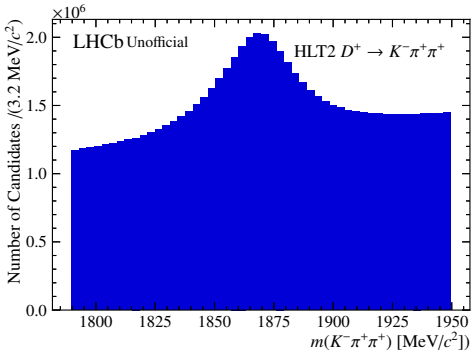
all events must have at least one reconstructed primary vertex (PV). After the Kalman filter, a common track χ^2/ndf cut is applied, set to $\chi^2/\text{ndf} < 4.2$ during commissioning (*cf.* Figure 6.8). The momentum requirements are chosen not to cut out the low-p_T region of the D^0 but still reject most of the low-momentum background. The variable χ^2_IP is the impact parameter significance, *i.e.* tracks with a large χ^2_IP are significantly displaced from any PV, a feature of tracks coming from the decay of a particle with measurable lifetime like the D^0. The variable $\text{DLL}_{K\pi}$ is the log-likelihood difference between the hypothesis that the particle is a kaon or a pion and is a combination of information from the PID system. The variable χ^2_FD is the significance of the distance between the PV and the secondary vertex (SV), where the D^0 decayed, a significant value of which is a crucial selection feature of the weakly decaying heavy-flavour hadrons. The direction angle (DIRA) is defined as the angle between the momentum vector of the D^0 and the direction vector obtained by connecting the PV and the SV and ensures that the decay is not only reconstructed partially. The quality of the SV is denoted by χ^2_vtx. The HLT2 trigger lines for the other charm meson decays apply similar cuts and are listed in Appendix A.7.

6.5.3 Mass Distributions

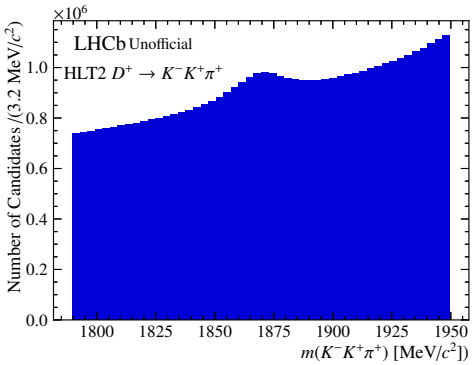
Parts of the data taken during commissioning have been processed by HLT2 using the selections to measure the charm production cross-section. As explained in the previous section, the data quality is not yet good enough for physics analysis. Yet, the selections were explicitly chosen to be loose to work under suboptimal conditions. The resulting mass distributions are shown in Figure 6.14. Each channel exhibits a mass peak, albeit dominated by background for all but the two-body D^0 decay. This is expected, and tighter selections are ready for nominal data taking, cutting harder on the impact parameter, the flight distance, and the position of the SV with respect to detector material to remove candidates from material interactions. The selections were tuned on simulation where the detector is perfectly aligned, so the mass windows were chosen a bit too narrow for the amount of misalignment present in the data. From a track reconstruction point of view, it is reassuring to observe the four-body D^0 decay, which would suffer the most from a suboptimal tracking performance. All in all, successfully reconstructing these decays with a not-fully commissioned detector shows that the Run 3 apparatus and its software, in principle, work.



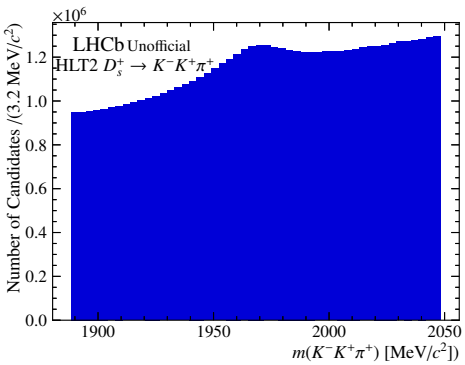
a: `Hlt2Charm_D0ToKmPip_XSec`.



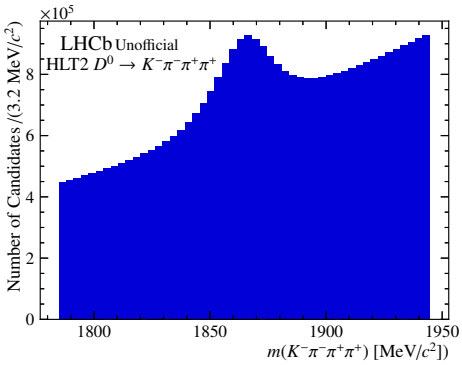
b: `Hlt2Charm_DpToKmPipPip_XSec`.



c: `Hlt2Charm_DpToKmKpPip_XSec`.



d: `Hlt2Charm_DspToKmKpPip_XSec`.



e: `Hlt2Charm_D0ToKmPimPipPip_XSec`.

Figure 6.14: D meson invariant mass distributions as selected by the corresponding HLT2 trigger lines.

7 Conclusion

The work presented in this dissertation highlights how classical pattern recognition can be implemented as a component of a cutting-edge, purely software-based trigger system on modern CPUs. For this, the track reconstruction software for LHCb's Run 3 HLT2 with a focus on the Forward tracking has been described, including its first commissioning using the decay $K^0_S \to \pi^+\pi^-$ and the prospect of further validation using charm-mesons decays.

By using the capabilities of modern CPUs and following best practices of high-performance computing, the track reconstruction's event throughput was increased by several factors such that particle collisions can be reconstructed with offline quality already in the trigger using the limited resources provided by the event-filter farm. The Forward tracking approach has been redesigned completely to gain a factor of 3.5 in throughput by utilising SIMD parallelism, efficient data structures and effective descriptions of the particles' passage through the magnetic field. This was achieved without trading off on the physics performance of the algorithm; the Forward tracking finds Long tracks using a Hough-like transform with an efficiency of 88% integrated over the whole momentum spectrum and is 95.8% efficient for high-momentum tracks originating from a B meson. The event throughput of the Forward tracking reaches 3527 events per second and computing node, corresponding to an average single-thread executing time of 11 ms per non-empty pp collision. The fake-track fraction is controlled using a neural network reducing it to 14%, significantly lower than the ghost rate of the presented neural-network-based Long track reconstruction algorithm, the Matching. Both algorithms achieve comparable track-finding efficiencies for all reconstructible particles except electrons, for which the Forward tracking is slightly more efficient. Furthermore, it was shown that the throughput of the Forward tracking increases by another factor of 2.7 by using the UT to filter the input tracks, which additionally halves the fake track fraction while reaching similar track-finding efficiencies.

Eventually, the Forward tracking and the entire track reconstruction software were demonstrated to work under challenging conditions in a not fully commissioned experiment with a misaligned tracking system. To that end, the author was able to reconstruct the first K^0_S mass peak for LHCb in Run 3. Finally, it was shown that also charm-hadron decays with up to four tracks in the final state can be reconstructed, which is part of the preparations for detector- and trigger-validation measurements aiming to determine the charm-production cross-section in the forward region at $\sqrt{s} = 13.6\,\text{TeV}$.

A Appendix

A.1 Equation of Motion w.r.t z Coordinate of a Charged Particle in the Magnetic Field

The goal is to transform the differential equation Equation 4.4 such that it depends on the z coordinate instead of the path length s. An infinitesimal piece of the path length in cartesian coordinates is given by

$$\mathrm{d}s^2 = \mathrm{d}x^2 + \mathrm{d}y^2 + \mathrm{d}z^2 \tag{A.1}$$

yielding the Jacobian determinant

$$J \equiv \frac{\mathrm{d}s}{\mathrm{d}z} = \sqrt{1 + \left(\frac{\mathrm{d}x}{\mathrm{d}z}\right)^2 + \left(\frac{\mathrm{d}y}{\mathrm{d}z}\right)^2} \tag{A.2}$$

The second derivative of $\mathbf{x}$ with respect to s can be written as

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}z} = J\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s} \implies \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s} = J^{-1}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}z} \tag{A.3}$$

$$\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}s^2} = \frac{\mathrm{d}}{\mathrm{d}s}\left[J^{-1}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}z}\right] = J^{-1}\frac{\mathrm{d}}{\mathrm{d}z}\left[J^{-1}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}z}\right] \tag{A.4}$$

$$= J^{-2}\left[\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}z^2} - J^{-1}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}z}\frac{\mathrm{d}J}{\mathrm{d}z}\right] \tag{A.5}$$

with

$$\frac{\mathrm{d}J}{\mathrm{d}z} = J^{-1}(x'x'' + y'y'') \tag{A.6}$$

where the prime indicates a derivative with respect to z. Plugging Equation A.3 and Equation A.5 into Equation 4.4 yields

$$\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}z^2} = J^{-2}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}z}(x'x'' + y'y'') + J\kappa\frac{q}{p}\left[\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}z} \times \mathbf{B}(\mathbf{x})\right] \tag{A.7}$$

which shows that the system of differential equations is coupled. The first component reads

$$x'' = \left(y' \left[\frac{x'y''}{J^2} + J\kappa \frac{q}{p} B_z \right] - \kappa \frac{q}{p} B_y \right) \left[1 - \left(\frac{x'}{J} \right)^2 \right]^{-1} \tag{A.8}$$

To find a useful expression for x'', the analogous expression

$$y'' = \left(x' \left[\frac{y'x''}{J^2} - J\kappa \frac{q}{p} B_z \right] + \kappa \frac{q}{p} B_x \right) \left[1 - \left(\frac{y'}{J} \right)^2 \right]^{-1} \tag{A.9}$$

is plugged into Equation A.8, which after simplifying becomes

$$x'' = \kappa \frac{q}{p} J \left[y'(B_z + B_x x') - B_y(x'^2 + 1) \right] \tag{A.10}$$

and analogously

$$y'' = \kappa \frac{q}{p} J \left[-x'(B_z + B_y y') + B_x(y'^2 + 1) \right] \tag{A.11}$$

A.2 Linear Least-Square Fits

Trajectory fits in the Forward tracking are performed using the linear least-square method. Following the description of the method from Ref. [111], the estimators for the parameters of the track model are given by

$$\hat{\boldsymbol{\theta}} = \underbrace{(\overbrace{A^T V^{-1} A}^{P})^{-1}}_{U} A^T V^{-1} \mathbf{x} \tag{A.12}$$

where A is the feature matrix with elements $A_{ij} = a_j(z_i)$ calculated from the track model's linearly independent monomials of each measurement at z_i, V is the covariance matrix of the measurements and $\mathbf{x}$ the corresponding measurement vector. The matrix V is easily inverted, assuming that the errors on the hits' x positions are independent of each other, in which case the matrix is diagonal. Calculating the inverse of the matrix P yields the covariance matrix U of the estimators and can be computationally expensive. That the inverse matrix exists is not guaranteed as the covariance matrix is only positive semi-definite, in praxis though the invertibility is rarely an issue[1]. For matrices up to rank three, which are by choice the only cases occurring in the Forward tracking, decently simple solutions can be written down using the fact that the inverse can be expressed as

$$U = \frac{\mathrm{adj}(P)}{\det(P)} \tag{A.13}$$

[1]Nevertheless, the implemented fits check if the determinant is different from zero to prevent division by zero, which, as we all know, would have catastrophic consequences. :)

i.e. using the determinant and the adjoint of the matrix [112]. To fit a straight-line trajectory, for example, the estimator covariance matrix is calculated by

$$U = \frac{1}{P_{11}P_{22} - P_{12}^2} \begin{pmatrix} P_{22} & -P_{12} \\ -P_{12} & P_{11} \end{pmatrix} \tag{A.14}$$

For 3×3 matrices, a similar expression can be found using the algebraic Leibniz formula.

The mathematics mentioned above should give the reader enough hints to understand the implementation of the fits in the Forward tracking (in fact, also some other places in LHCb's software). However, as usual with floating-point arithmetic, some caution has to be exercised regarding the numerical stability of calculation results. The whole pattern recognition, and so the Forward tracking, is implemented using single-precision floating-point numbers. It is therefore sound to not directly use the x coordinate measurements $\mathbf{x}$, the values of which might span three orders of magnitude, but to work with the distance $d = x_i - x_{\text{track}}$, where x_{track} is calculated using the prior best knowledge of the model parameters. The fit method then estimates the change of the parameters with respect to the prior knowledge instead of the absolute values, *i.e.* the resulting $\hat{\boldsymbol{\theta}}$ has to be added to the prior parameters and it is possible to fit only a subset of coefficients, while keeping others constant. Furthermore, when creating the matrix P for a fit with three parameters, some matrix elements are proportional to the third power of the z coordinate, which is a huge number for $\mathcal{O}(z) \simeq 1000\,\text{mm}$. Multiplying this with distances $\mathcal{O}(d) \simeq 1\,\text{mm}$ should be avoided. Hence for the matrix calculations, the z coordinates are scaled to the same order of magnitude as the distances. The resulting parameters must then be re-scaled.

A.3 Parameterisations

The parameterisations are a cornerstone of both the physics and the computational performance of the pattern recognition in LHCb. The latter is the driving force because effectively describing the particles' passage through the magnetic field by an approximation is calculated fast in contrast to expensively solving the equations of motion using numerical methods. The quality of the approximation then drives the physics performance. In a sense, the parameterisations are similar to a description by a neural network that has learnt to predict particle trajectories from a given initial state. Yet, because the underlying mechanics of the particle movement is not too complex, a neural network would be unnecessarily complicated to describe the desired quantities. Instead, simple linear regression is used to fit multi-dimensional dependencies and obtain a polynomial that describes the quantity of interest on average. Because these polynomials and neural networks, if necessary, occur in many of LHCb's pattern-recognition software components, I have implemented a Python-based software package

that provides a central and reproducible way to obtain the parametrisations used in LHCb's pattern recognition. As of writing this thesis, the `Reco-Parameterisation-Tuner` can be found in Ref. [113]. Still, the plan is to move it into the LHCb software stack, where it will be located inside the Rec project in the pattern recognition part. Wherever the package is to be found, the goal is to provide the code to reproduce every single parameterisation used in the track reconstruction. So far, the package features all parameterisations used in the Forward tracking and Matching, including the training of the neural networks. The machine-learning library *scikit-learn* [114] is used as the backend for the regressions. For training of the track reconstruction's MLPs, TMVA [102] is used.

Apart from documentation and reproducibility, reasons to implement, update or change the parameterisations might be changes in the magnetic field, the geometry or acceptance of the detector and better simulation. The intended workflow starts with the production of training data from simulated collisions. The package provides Moore option files that configure software components implemented to create ROOT files containing the necessary data. Currently, the distributions are taken directly from simulation, meaning that very high and very low momenta and very steep tracks are less well represented in the training data. This should improve in the future; the plan is to test sampling tracks equally in bins of $1/p$, t_x and t_y [69]. From the user's perspective, the next step is simply executing the predefined methods on the new data to obtain the supported parameterisations, which are directly written into C++ files from where the new coefficients can be directly copied or included into the software stack. The neural network training works the same way but takes more time than the linear regressions. The neural network weights are also written into C++ files in a format understood by the LHCb's custom implementation of the MLP's matrix evaluations.

If legacy parameterisations are to be ported to this package, I advise following the same code structure as the already implemented ones have. This is using scikit-learn's `PolynomialFeatures` to the desired degree and removing unused monomials. The coefficients are then found by employing `LinearRegression`. If entirely new parameterisations need to be found, a bit of trial-and-error is necessary to find the correct description. First, it can be helpful to study the distribution of the target feature as a function of the training features to see to which degree the polynomial needs to be defined. Additionally, it is helpful to know how precise and accurate the parameterisation needs to be to fulfil its purpose. In Section 5.2.4, the z_{mag} parameterisation, for example, needs to be precise but not very accurate, *i.e.* a small bias is acceptable as long as the variance is small. Only a rough estimate is necessary for estimating the momentum in the Forward tracking (Section 5.2.7) because the Kalman filter corrects deviations afterwards. When building the parameterisation, it should thus be monitored on a test sample if the desired performance is reached. Many different metrics are available; the mean-squared error or the R2 score are common for

quick checks. To identify biases and study variances in different parameter regions, it is, however, necessary to check the corresponding distributions, *e.g.* by regression plots such as Figure 5.7a or the underlying 2D distribution of the residuals and the (target) feature. The main problem with the polynomial features is that it is not obvious which monomials actually contain information about the target feature. The simple trial-and-error method is to systematically remove and add monomials until a minimal polynomial is found that describes the target feature. The maybe faster and smarter approach is to use *least absolute shrinkage and selection operator* (Lasso) regression. This regularises the loss function such that coefficients of monomials contributing little tend to zero. The regularisation parameter can be scanned to find the minimal parameterisation that fulfils the needs. The standard linear regression can be applied again to find the final coefficients as soon as the set of monomials, *i.e.* the polynomial is found. Most of the parameterisations documented in the following sections were found this way.

A.3.1 Trajectory Boundaries

Table A.1: Coefficients for Equation 5.1.

a: Linear coefficients (L).

Coefficient	Value
c_0	4018.90
c_1	6724.79
c_2	3970.90
c_5	4934.08
c_9	106069
c_{12}	−23936.5

b: Quadratic coefficients (Q).

Coefficient	Value
c_3	−4363.58
c_6	6985.25
c_{10}	−94446.8

c: Cubic coefficients (C).

Coefficient	Value
c_4	1421.11
c_7	−5538.28
c_{11}	26489.8

d: Quartic coefficients.

Coefficient	Value
c_8	1642.86

A.3.2 Magnet Centre

Table A.2: Coefficients for Equation 5.7.

Coefficient	Value
c_0	5205.14
c_1	-320.721
c_2	702.138
c_3	-316.364
c_4	441.599

A.3.3 Track Model

Table A.3: Coefficients for Equation 5.8.

a: Coefficients for c_x.

Coefficient	Value
c_0	2.335×10^{-5}
c_1	-5.394×10^{-8}
c_2	-1.135×10^{-6}
c_3	9.213×10^{-6}
c_4	-6.764×10^{-7}
c_5	-3.740×10^{-4}

b: Coefficients for d_x.

Coeffiecent	Value
c_0	-7.058×10^{-9}
c_1	1.052×10^{-11}
c_2	6.461×10^{-10}
c_3	2.596×10^{-9}
c_4	8.044×10^{-11}
c_5	9.934×10^{-8}

Table A.4: Coefficients for Equation 5.12.

a: Coefficients linear in Δt_x.

Stereo layer	c_0	c_2	c_5	c_6
1	1.91414	3719.30	41484.8	30544.7
2	1.98021	3767.00	42635.3	31435.0
5	2.60366	4236.34	52670.2	39380.5
6	2.68024	4296.65	53813.8	40299.1
9	3.38271	4875.42	63618.0	48278.8
10	3.46572	4946.54	64707.5	49179.4

b: Coefficients linear in $|\Delta t_x|$.

Stereo layer	c_1	c_3	c_4	c_7
1	154.619	−6981.57	−67.7612	211219
2	146.342	−7381.00	18.4078	218404
5	53.2313	−10844.8	986.150	281251
6	40.7583	−11234.8	1115.36	288432
9	−76.6132	−14585.2	2322.16	350658
10	−90.5897	−14962.3	2464.76	357681

Table A.5: Coefficients for Equation 5.12 targeting the reference plane.

a: Coefficients linear in Δt_x.

z [mm]	c_0	c_2	c_5	c_6
8520	2.54155	4187.53	51730.0	38622.5

b: Coefficients linear in $|\Delta t_x|$.

z [mm]	c_1	c_3	c_4	c_7
8520	63.2584	−10520.3	881.686	275326

Table A.6: Coefficients for $\hat{b}_y$ (Equation 5.18).

Coefficient	Value
c_0	0.93462
c_1	−0.46580
c_2	−4.11981
c_3	2.95148
c_4	12.5961
c_5	39.9847

Table A.7: Coefficients for $\hat{c}_y$ (Equation 5.19).

Coefficient	Value
c_0	-1.20348×10^{-5}
c_1	8.34465×10^{-5}
c_2	-3.92497×10^{-5}
c_3	0.00025
c_4	0.00019

Table A.8: Coefficients for $y_{\text{corr}}^{\text{EndT}}$, Equation 5.24.

Coefficient	Value
c_0	4227.70
c_1	49.5714
c_2	1771.73
c_3	1939.31
c_4	1536.75
c_5	287.174
c_6	1901.27
c_7	−1757.69
c_8	17704.0

A.3.4 Hough Histogram

Table A.9: Coefficients for Equation 5.15.

Coefficient	Value
p_0	576.971
p_1	0.57803
p_2	0.00067

A.3.5 Momentum Estimation

Table A.10: Coefficients for Equation 5.22.

Coefficient	Value
c_0	-1.20945
c_1	-2.78970
c_2	-0.359769
c_3	-0.47139
c_4	-0.56008
c_5	14.0093
c_6	-16.1628
c_7	-8.80799
c_8	-0.87532
c_9	2.98254
c_{10}	0.96254
c_{11}	0.10201

A.3.6 Parameterisations of the Matching Algorithm

Table A.11: Coefficients for $z_{\text{mag}}^{\text{match}}$ (Equation 5.27).

Coefficient	Value
c_0	5286.69
c_1	−3.25969
c_2	0.015616
c_3	−1377.32
c_4	282.982

Table A.12: Coefficients for $y_{\text{corr}}^{\text{match}}$ (Equation 5.28).

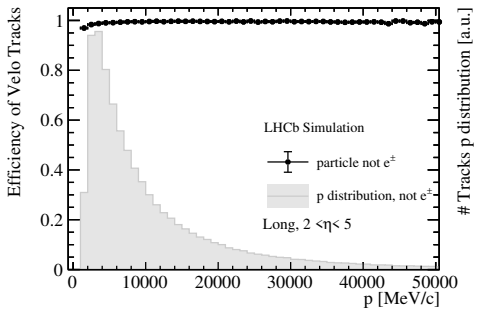
Coefficient	Value
c_0	1974.64
c_1	−35933.8

A.4 Reconstruction Performance

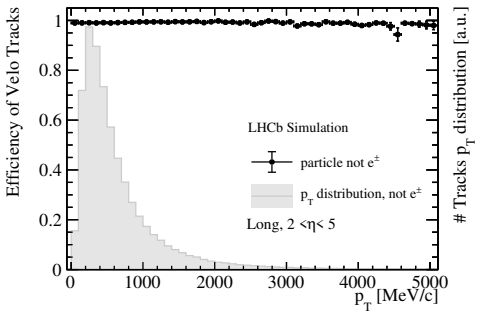
The appendix is related to Section 4.4.1.

A.4.1 VELO Tracking

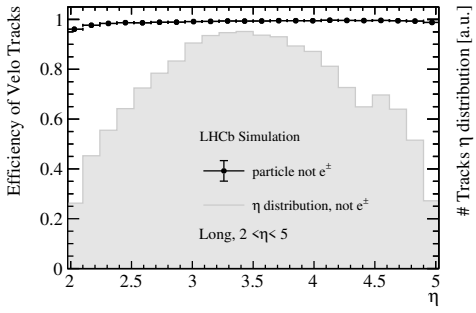
Track finding and efficiency plots for the VELO Tracking used as input to the Long tracking algorithms.



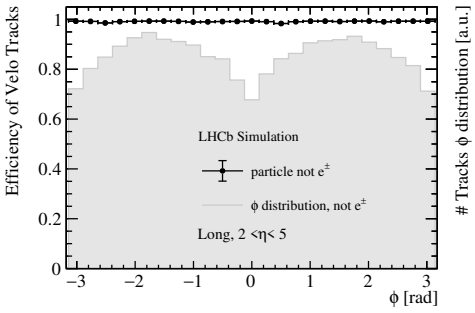
a: Efficiency *vs.* momentum.



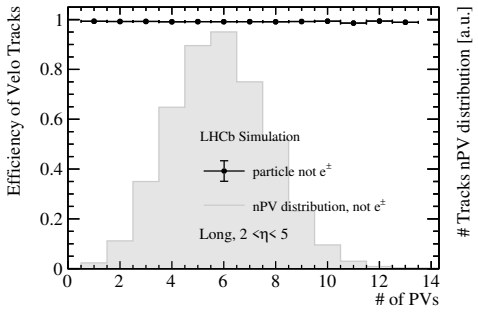
b: Efficiency *vs.* transverse momentum.



c: Efficiency *vs.* pseudorapidity.



d: Efficiency *vs.* azimuthal angle.

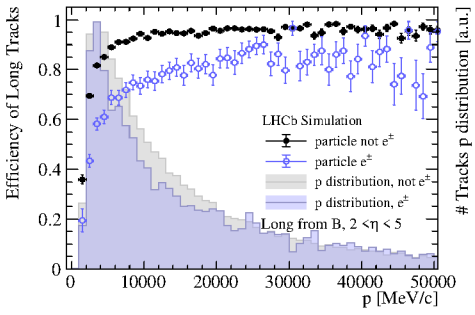


e: Efficiency *vs.* number of primary vertices.

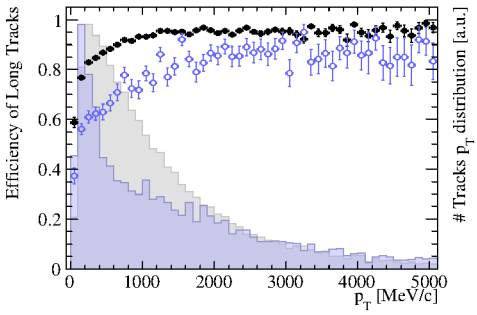
Figure A.1: Track finding efficiency of the VELO tracking algorithm for simulated Long tracks with $2 < \eta < 5$ in dependence on various kinematic variables and the number of primary vertices. The grey distributions show the true variable values of Long-reconstructible particles.

A.4.2 Forward Tracking on UT-filtered VELO Tracks

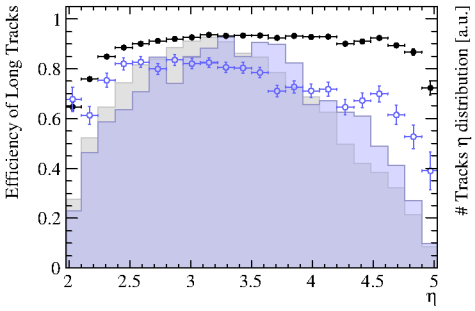
Track finding and efficiency plots for the Forward Tracking with UT-filtered VELO tracks as input. Two scenarios are shown, with and without the momentum-dependent search window.



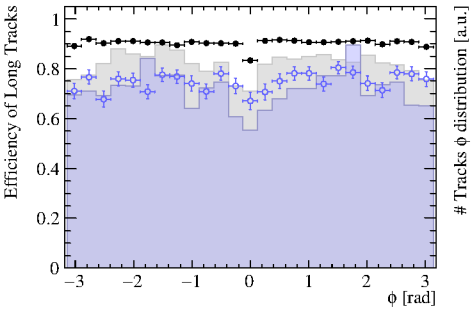
a: Efficiency *vs.* momentum.



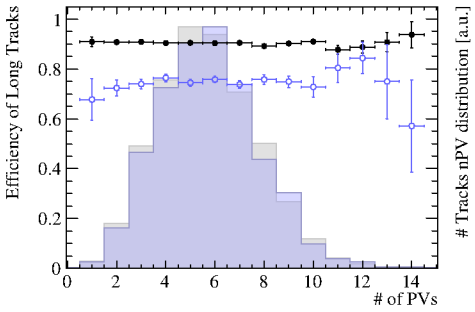
b: Efficiency *vs.* transverse momentum.



c: Efficiency *vs.* pseudorapidity.

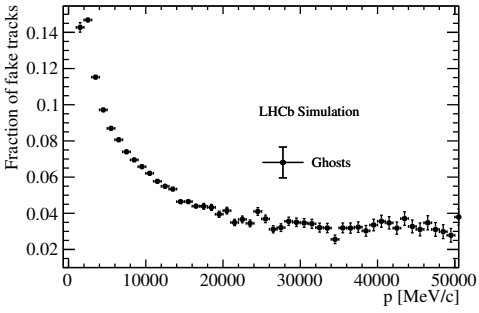


d: Efficiency *vs.* azimuthal angle.

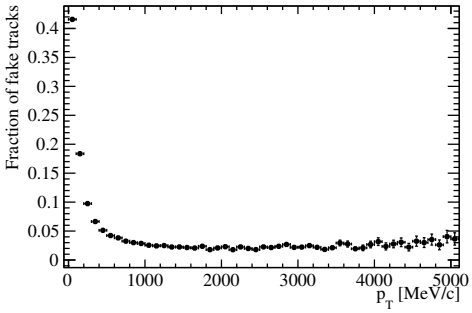


e: Efficiency *vs.* number of primary vertices.

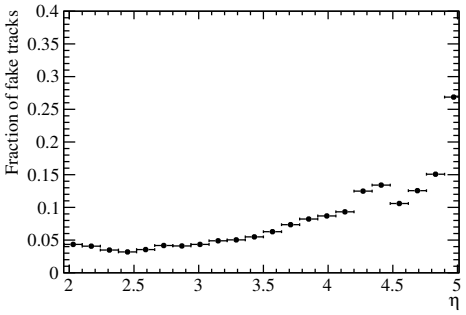
Figure A.2: Track finding efficiency of the Forward tracking algorithm with UT-filtered VELO tracks for simulated Long tracks with $2 < \eta < 5$ originating from a B meson in dependence on various kinematic variables and the number of primary vertices. No momentum-dependent search window is applied. The simulated distributions of the variables and the efficiencies are given for reconstructible electrons and other reconstructible particles separately; electrons in blue with empty-circle markers and other particles in black with full-circle markers. The same samples as for the training of the fake track rejection classifier (Section 5.2.8) are used to avoid a specific bug related to UT hit decoding.



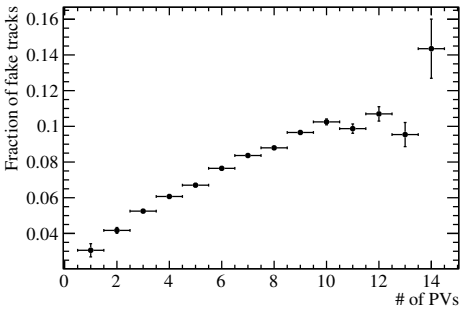
a: Fake track fraction *vs.* momentum.



b: Fake track fraction *vs.* transverse momentum.

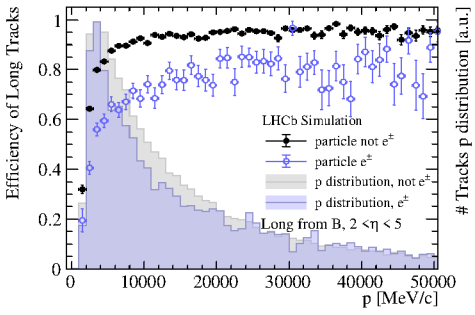


c: Fake track fraction *vs.* pseudorapidity.

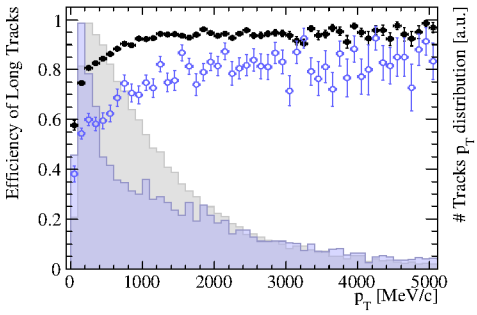


d: Fake track fraction *vs.* number of primary vertices.

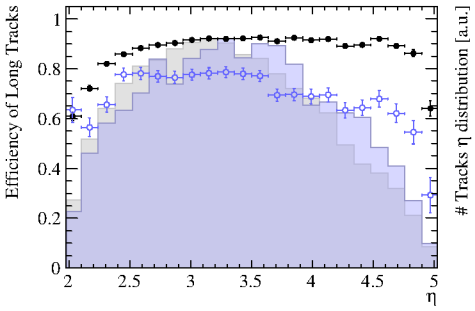
Figure A.3: Fake track fraction of the Forward tracking algorithm with UT-filtered VELO tracks as input in dependence on kinematic variables and the number of primary vertices. No momentum-dependent search window is applied.



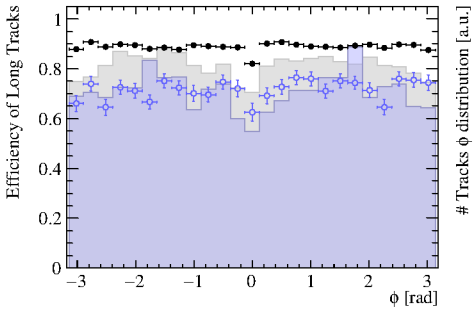
a: Efficiency *vs.* momentum.



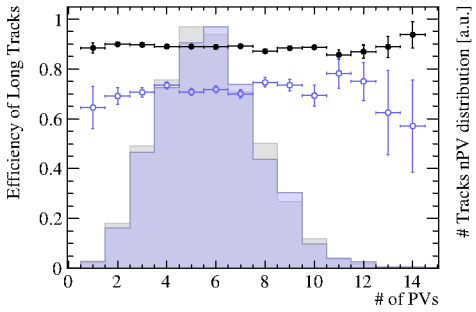
b: Efficiency *vs.* transverse momentum.



c: Efficiency *vs.* pseudorapidity.

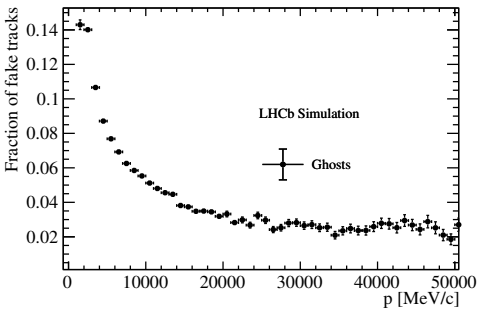


d: Efficiency *vs.* azimuthal angle.

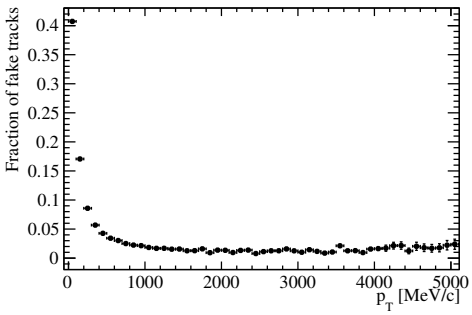


e: Efficiency *vs.* number of primary vertices.

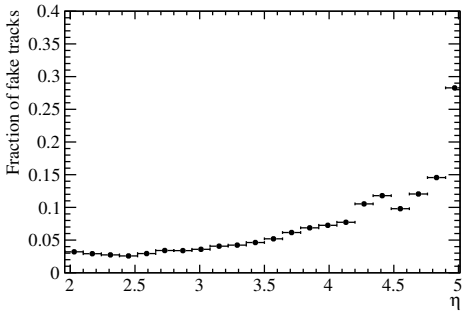
Figure A.4: Track finding efficiency of the Forward tracking algorithm with UT-filtered VELO tracks for simulated Long tracks with $2 < \eta < 5$ originating from a B meson in dependence on various kinematic variables and the number of primary vertices. The momentum-dependent search window is applied. The simulated distributions of the variables and the efficiencies are given for reconstructible electrons and other reconstructible particles separately; electrons in blue with empty-circle markers and other particles in black with full-circle markers. The same samples as for the training of the fake track rejection classifier (Section 5.2.8) are used to avoid a specific bug related to UT hit decoding.



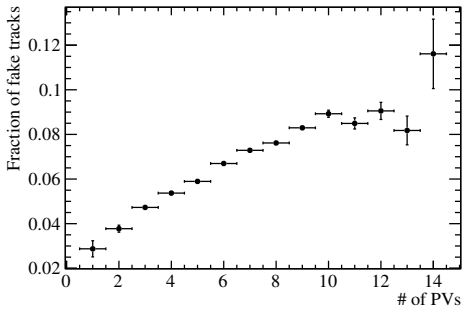
a: Fake track fraction *vs.* momentum.



b: Fake track fraction *vs.* transverse momentum.



c: Fake track fraction *vs.* pseudorapidity.

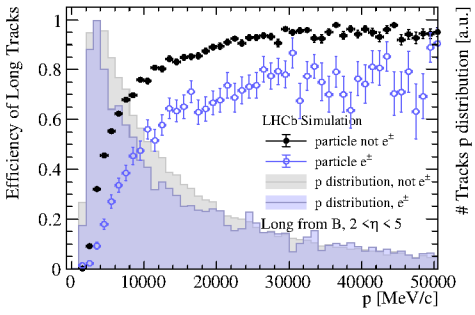


d: Fake track fraction *vs.* number of primary vertices.

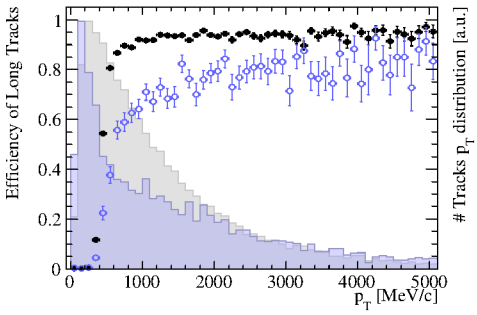
Figure A.5: Fake track fraction of the Forward tracking algorithm with UT-filtered VELO tracks as input in dependence on kinematic variables and the number of primary vertices. The momentum-dependent search window is applied.

A.4.3 Forward Tracking on Upstream Tracks

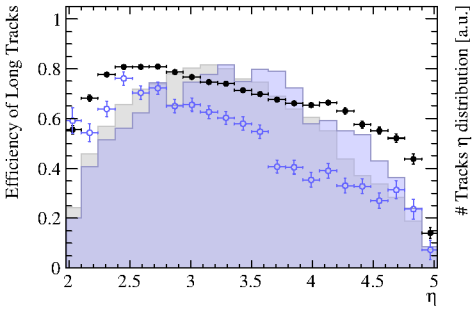
Track finding and efficiency plots for the Forward Tracking with Upstream tracks as input from the VeloUT algorithm with the default configuration. Two scenarios are shown, with and without the momentum-dependent search window. The appendix is related to Section 4.4.1.



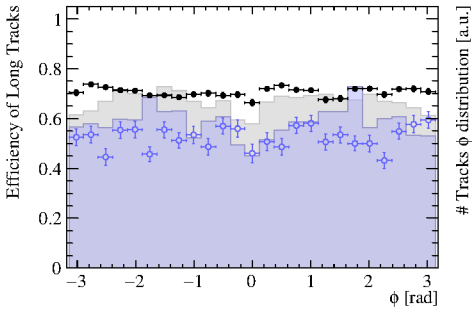
a: Efficiency *vs.* momentum.



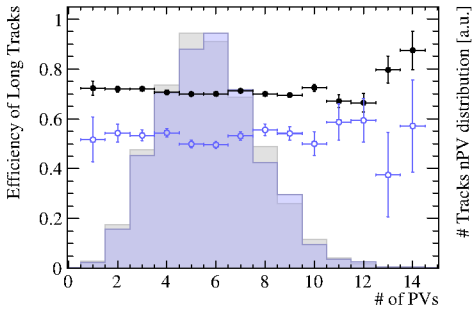
b: Efficiency *vs.* transverse momentum.



c: Efficiency *vs.* pseudorapidity.

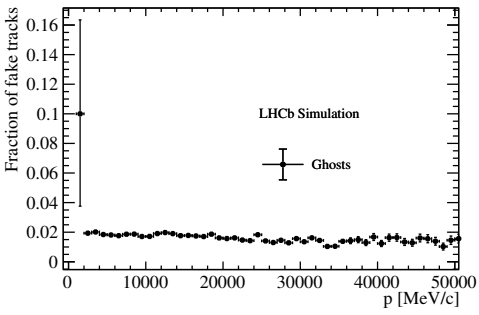


d: Efficiency *vs.* azimuthal angle.

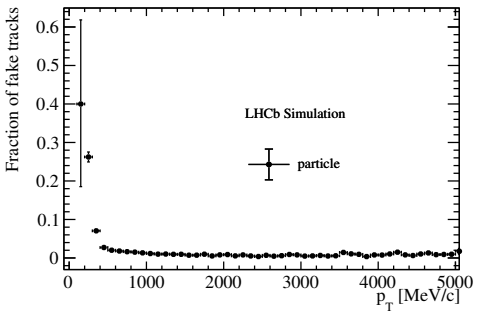


e: Efficiency *vs.* number of primary vertices.

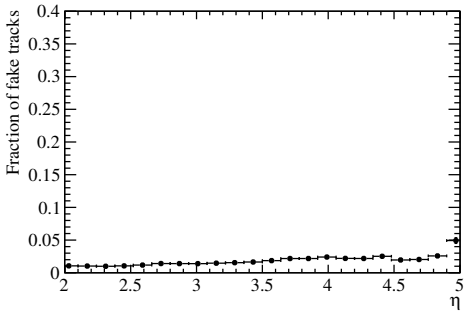
Figure A.6: Track finding efficiency of the Forward tracking algorithm with Upstream tracks for simulated Long tracks with $2 < \eta < 5$ originating from a B meson in dependence on various kinematic variables and the number of primary vertices. The momentum-dependent search window is applied. The simulated distributions of the variables and the efficiencies are given for reconstructible electrons and other reconstructible particles separately; electrons in blue with empty-circle markers and other particles in black with full-circle markers. The same samples as for the training of the fake track rejection classifier (Section 5.2.8) are used to avoid a specific bug related to UT hit decoding.



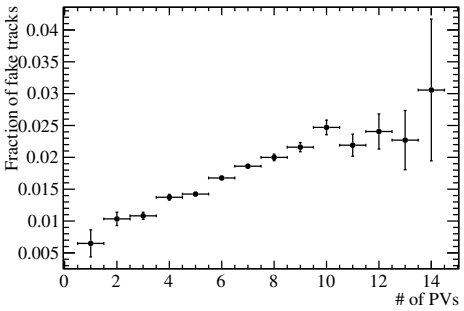
a: Fake track fraction *vs.* momentum.



b: Fake track fraction *vs.* transverse momentum.

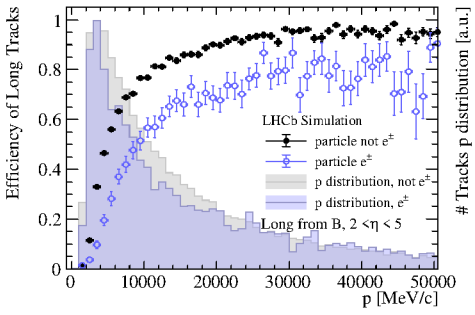


c: Fake track fraction *vs.* pseudorapidity.

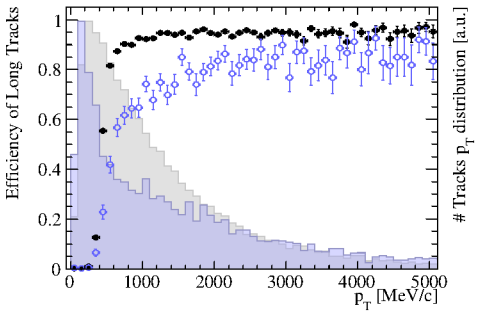


d: Fake track fraction *vs.* number of primary vertices.

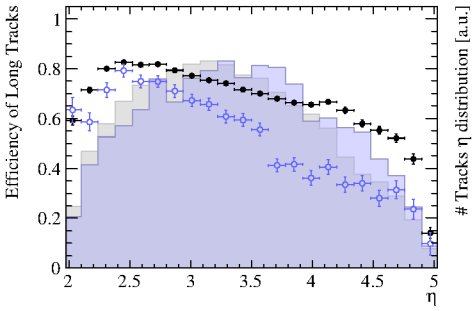
Figure A.7: Fake track fraction of the Forward tracking algorithm with Upstream tracks as input in dependence on kinematic variables and the number of primary vertices. The momentum-dependent search window is applied.



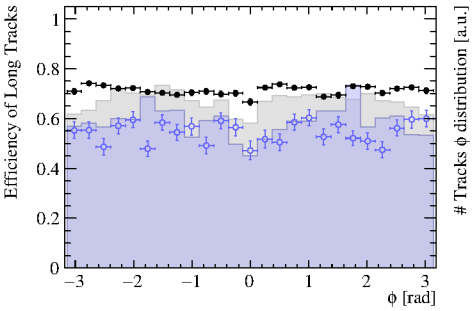
a: Efficiency *vs.* momentum.



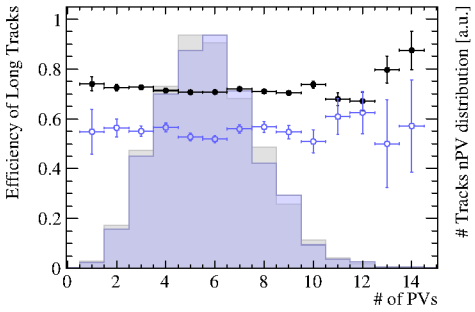
b: Efficiency *vs.* transverse momentum.



c: Efficiency *vs.* pseudorapidity.

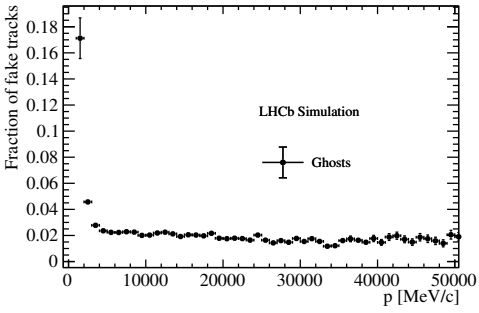


d: Efficiency *vs.* azimuthal angle.

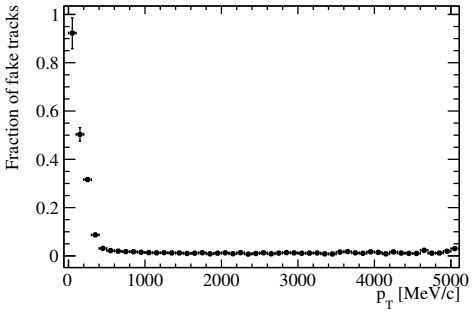


e: Efficiency *vs.* number of primary vertices.

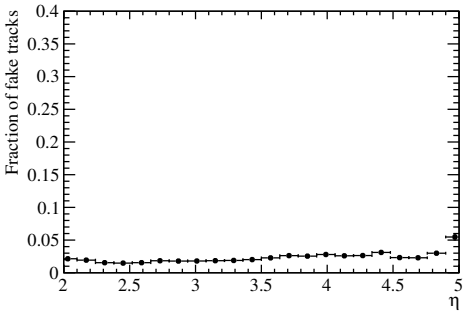
Figure A.8: Track finding efficiency of the Forward tracking algorithm with Upstream tracks for simulated Long tracks with $2 < \eta < 5$ originating from a B meson in dependence on various kinematic variables and the number of primary vertices. No momentum-dependent search window is applied. The simulated distributions of the variables and the efficiencies are given for reconstructible electrons and other reconstructible particles separately; electrons in blue with empty-circle markers and other particles in black with full-circle markers. The same samples as for the training of the fake track rejection classifier (Section 5.2.8) are used to avoid a specific bug related to UT hit decoding.



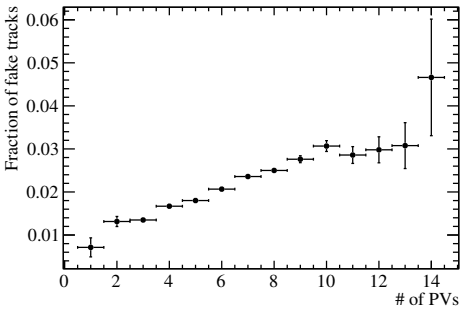
a: Fake track fraction *vs.* momentum.



b: Fake track fraction *vs.* transverse momentum.



c: Fake track fraction *vs.* pseudorapidity.



d: Fake track fraction *vs.* number of primary vertices.

Figure A.9: Fake track fraction of the Forward tracking algorithm with Upstream tracks as input in dependence on kinematic variables and the number of primary vertices. No momentum-dependent search window is applied.

A.4.4 Comparison to Matching Algorithm

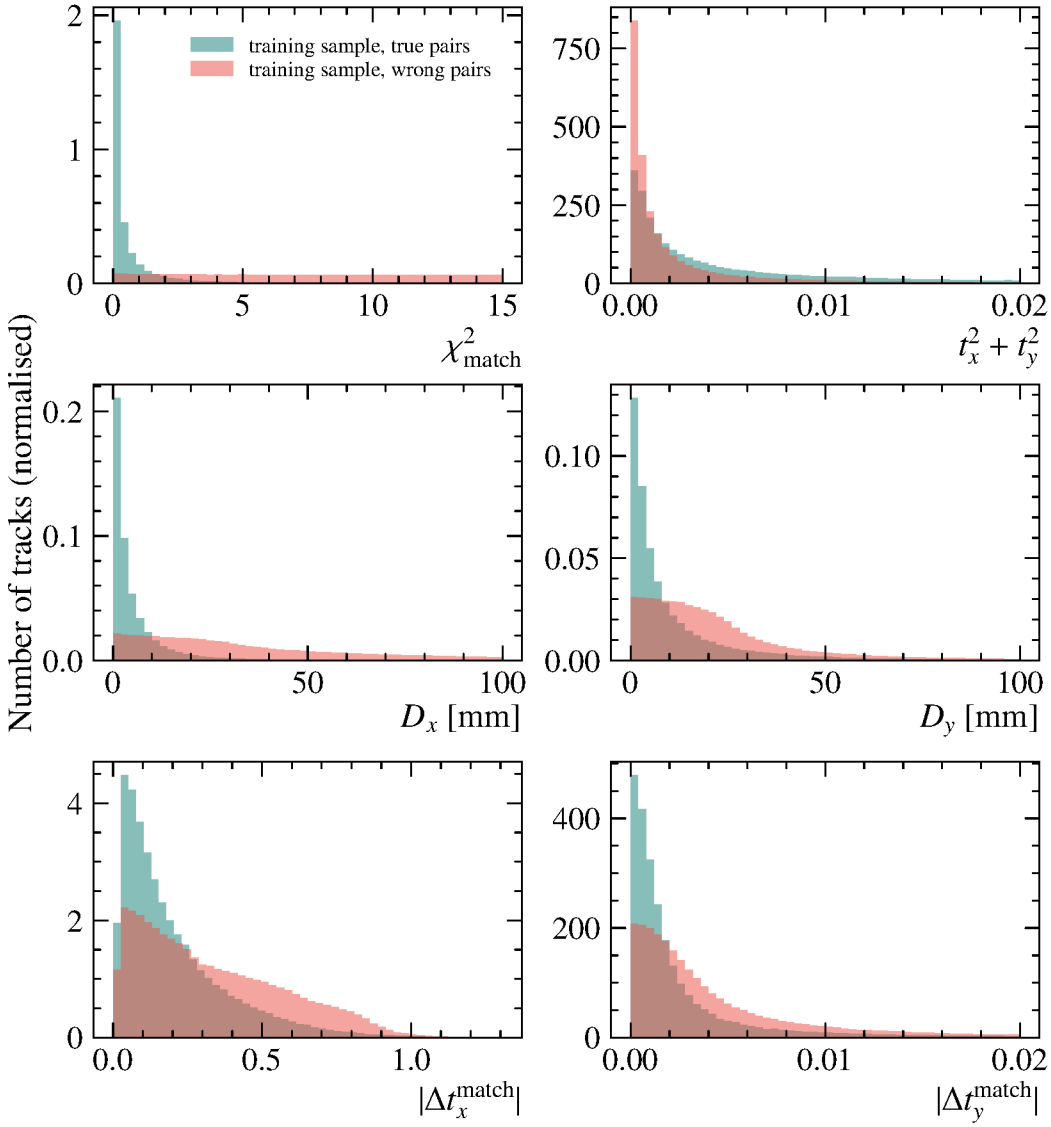
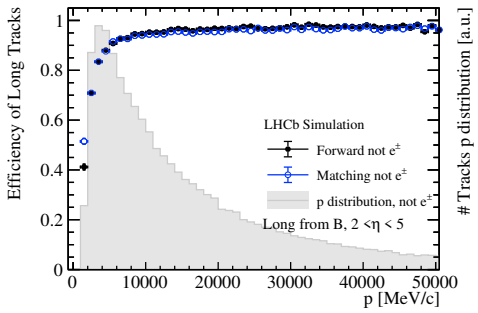
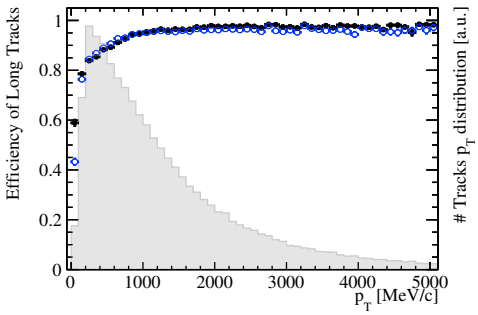


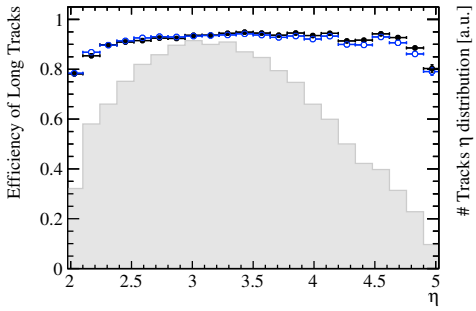
Figure A.10: Distributions of the Matching MLP's input variables listed in Table 5.7.



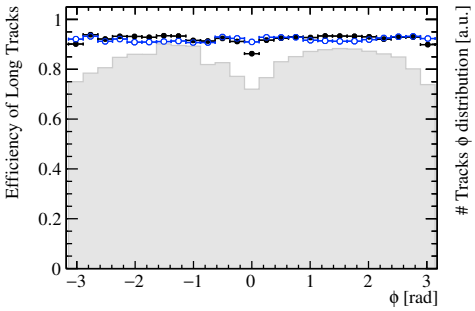
a: Efficiency *vs.* momentum.



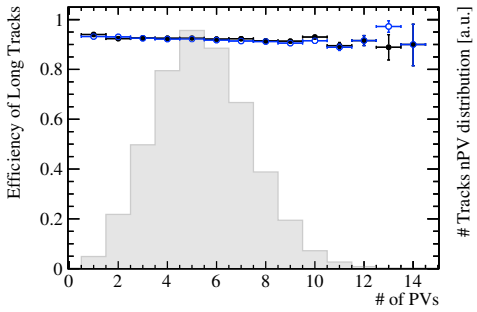
b: Efficiency *vs.* transverse momentum.



c: Efficiency *vs.* pseudorapidity.

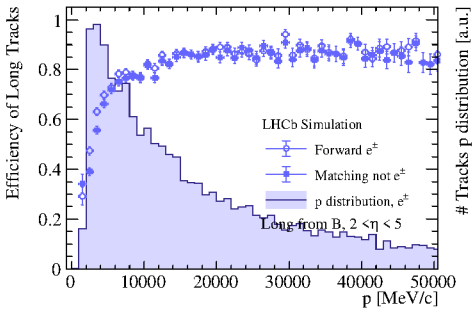


d: Efficiency *vs.* azimuthal angle.

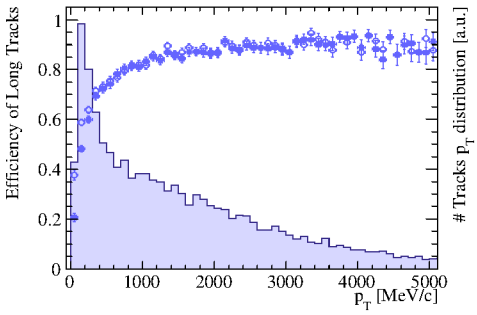


e: Efficiency *vs.* number of primary vertices.

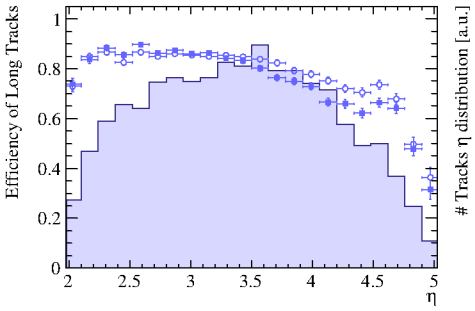
Figure A.11: Track finding efficiency of the Forward tracking algorithm with VELO input tracks compared to the Matching algorithm for simulated Long tracks with $2 < \eta < 5$ from a B meson in dependence on various kinematic variables and the number of primary vertices. The Matching is shown in empty blue circles and the Forward tracking is in full black circles.



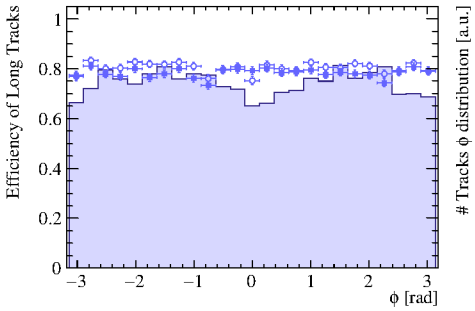
a: Efficiency *vs.* momentum.



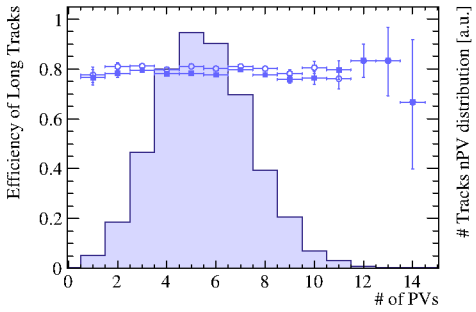
b: Efficiency *vs.* transverse momentum.



c: Efficiency *vs.* pseudorapidity.



d: Efficiency *vs.* azimuthal angle.



e: Efficiency *vs.* number of primary vertices.

Figure A.12: Electron track finding efficiency of the Forward tracking algorithm with VELO input tracks compared to the Matching algorithm for simulated Long tracks with $2 < \eta < 5$ from a B meson in dependence on various kinematic variables and the number of primary vertices. The Matching is shown in full blue rectangles and the Forward tracking is in empty blue circles.

A.5 Magnet, Beam Pipe, and the SciFi Tracker

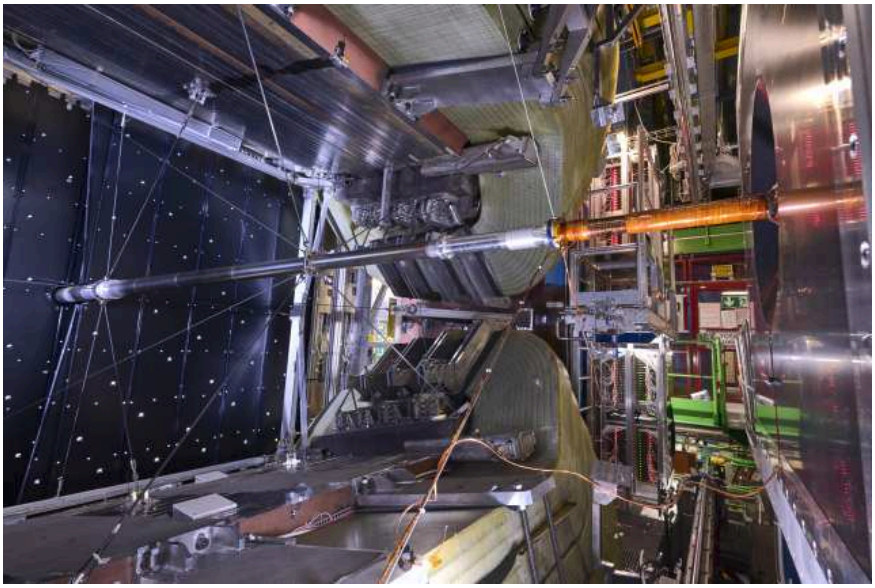


Figure A.13: Picture [115] of the beampipe and its support structures within the magnet. On the left, the first layer of the SciFi tracker is visible. The empty space on the right is where the UT and RICH1 detectors are located.

A.6 Tracking Configuration

Table A.13: Loose track reconstruction configuration used in Section 6.3.

Component	Property	Value
Forward	MLP response	> 0
	ΔMLP response	$= 0$
	$n_{\text{Hits}}^{\text{tot}}$	> 8
	single x hit χ^2	< 60
	single uv hit χ^2	< 16
	first fit χ^2/ndf	< 50
	last fit χ^2/ndf	< 28
Matching	χ^2_{match}	< 20
	MLP response	> 0
	ΔMLP response	< 0.3
	δy	$= 12\,\text{mm}$
	$t_{\delta y}$	$= 600\,\text{mm}$
Track Fit	χ^2/ndf	< 20
	outlier removal χ^2	> 25

A.7 Charm Meson HLT2 Trigger Lines

Table A.14: HLT2 trigger selection for $D^+_{(s)} \to h^-h^+h^+$ with $h^\pm \in \{K^\pm, \pi^+\}$.

Particle	Selection
$K^\pm$, π^+	$p_\mathrm{T} > 250\,\mathrm{MeV}/c$
	$p > 2\,\mathrm{GeV}/c$
	$\chi^2_\mathrm{IP} > 4$
	$\mathrm{DOCA}(h_i, h_j) < 0.2\,\mathrm{mm}$
	$\sum_i p_\mathrm{T}(h_i) > 1.5\,\mathrm{GeV}/c$
	$\sum_i \chi^2_\mathrm{IP}(h_i) > 64$
$K^\pm$	$\mathrm{DLL}_{K\pi} > 5$
π^+	$\mathrm{DLL}_{K\pi} < 5$
D^+/D^+_s	$m_D - 80 < m(h^-h^+h^+) < m_D + 80\,\mathrm{MeV}/c^2$
	$\chi^2_\mathrm{FD} > 50$
	$\mathrm{DIRA} < 30\,\mathrm{mrad}$
	$\chi^2_\mathrm{vtx} < 10$
	reconstructed lifetime $> 0.15\,\mathrm{ps}$

Table A.15: HLT2 trigger selection for $D^0 \to K^-\pi^-\pi^+\pi^+$.

Particle	Selection
K^-, $\pi^\pm$	$p_\mathrm{T} > 250\,\mathrm{MeV}/c$
	$p > 2\,\mathrm{GeV}/c$
	$\chi^2_\mathrm{IP} > 4$
	$\mathrm{DOCA}(h_i, h_j) < 0.2\,\mathrm{mm}$
	$\sum_i p_\mathrm{T}(h_i) > 1.5\,\mathrm{GeV}/c$
	$\sum_i \chi^2_\mathrm{IP}(h_i) > 50$
K^-	$\mathrm{DLL}_{K\pi} > 5$
$\pi^\pm$	$\mathrm{DLL}_{K\pi} < 5$
D^0	$m_{D^0} - 80 < m(h^-h^+h^+) < m_{D^0} + 80\,\mathrm{MeV}/c^2$
	$\chi^2_\mathrm{FD} > 49$
	$\mathrm{DIRA} < 20\,\mathrm{mrad}$
	$\chi^2_\mathrm{vtx} < 10$
	reconstructed lifetime $> 0.15\,\mathrm{ps}$

Table A.16: HLT2 trigger selection for $D^{*+} \to D^0\pi^+$. The D^0 is selected by Table 6.6.

Particle	Selection
π^+	$p_\mathrm{T} > 100\,\mathrm{MeV}/c$
	$p > 1.5\,\mathrm{GeV}/c$
D^{*+}	$m(D^0\pi^+) - m(D^0) < 160\,\mathrm{MeV}/c^2$
	$\chi^2_\mathrm{vtx} < 25$

Acknowledgments

I want to express my gratitude to everyone who supported and guided me as a PhD student at the Physikalisches Institut Heidelberg and CERN. In particular, I'd like to thank Stephanie Hansmann-Menzemer, who did an outstanding job supervising me throughout my entire particle physics studies, together with Michel De Cian, from whom I gained the most hands-on LHCb knowledge.

I want to thank Andre Schöning for being the second referee for this work and Fred Hamprecht and Björn Malte Schäfer, who kindly agreed to complete my PhD committee.

I'd like to thank Sascha Stahl, Peilian Li, Christoph Hasse, Arthur Hennequin, Louis Henry, Rosen Matev, Marian Stahl, Sevda Esen and Laurent Dufour, who all gave me lots of guidance and were always open to answering my questions or engaging in discussions with me about software, hardware, the management, physics and what not.

Furthermore, I am grateful for the support of the whole local Heidelberg LHCb group, of which I want to thank in particular Blake Leverington and Martino Borsato, who shared their detector and physics analysis knowledge with me, and the IT crew around Alexey Zhelezov and Joerg Marks, who provided me computing infrastructure and support making my work a lot easier.

Lastly, my gratitude goes to my friends and family; their support was indispensable. In addition to my parents, Gabi and Rolf, and my sister, Sarah, I want to explicitly thank Menno, Bernhard, and Arthur, who are the remaining fellow sufferers in Heidelberg, Basti and his flatmates who never fail to clear away the cobwebs of work, and Niko providing me shelter and asylum in Cologne whenever needed or wanted. The most special thanks go to Caroline, who became my most important mental, emotional and social supporter in the last two years.

Bibliography

[1] J. J. Thomson, "XL. Cathode Rays," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 44, no. 269, pp. 293–316, Oct. 1, 1897.

[2] C. D. Anderson, "The Positive Electron," *Physical Review*, vol. 43, no. 6, pp. 491–494, Mar. 15, 1933.

[3] S. H. Neddermeyer and C. D. Anderson, "Note on the Nature of Cosmic-Ray Particles," *Physical Review*, vol. 51, no. 10, pp. 884–886, May 15, 1937.

[4] S. Chen *et al.*, "Heavy flavour physics and CP violation at LHCb: A ten-year review," *Frontiers of Physics*, vol. 18, no. 4, p. 44 601, Mar. 21, 2023.

[5] B. Fang, "Recent LHCb results on heavy hadron spectroscopy," *Nuclear and Particle Physics Proceedings*, QCD 21 Is the 24th International Conference on Quantum Chromodynamics, pp. 66–72, Nov. 1, 2022.

[6] J. Albrecht, "Search for New Physics in rare decays at LHCb," *Nuclear Physics B - Proceedings Supplements*, Proceedings of the Fourth Workshop on Theory, Phenomenology and Experiments in Heavy Flavour Physics, pp. 49–54, Aug. 1, 2013.

[7] LHCb collaboration, "Test of lepton universality in $b \to s\ell^+\ell^-$ decays," *preprint*, LHCb-PAPER-2022-046, Dec. 18, 2022. arXiv: 2212.09152 [hep-ex].

[8] A. A. Alves Junior *et al.*, "The LHCb Detector at the LHC," *Journal of Instrumentation*, vol. 3, no. 08, S08005, Aug. 2008.

[9] F. Follin and D. Jacquet, "Implementation and experience with luminosity levelling with offset beam," *CERN Yellow Report*, CERN-2014-004, pp.183–187, 2014. arXiv: 1410.3667 [physics].

[10] C. Hasse. "Simple sketch of the LHC." (Feb. 22, 2023), [Online]. Available: https://github.com/hassec/LHC_Sketch (visited on 03/23/2023).

[11] Christian Elsässer. "$\bar{b}b$ Production angle plots." (2023), [Online]. Available: https://lhcb.web.cern.ch/speakersbureau/html/bb_productionangles.html (visited on 03/23/2023).

[12] "LHCb magnet: Technical Design Report," CERN, Geneva, LHCb-TDR-1, 2000.

[13] C. Hasse, "Alternative approaches in the event reconstruction of LHCb," Ph.D. dissertation, Tech. U., Dortmund, 2019.

[14] M. Losasso *et al.*, "Tests and Field Map of LHCb Dipole Magnet," *Applied Superconductivity, IEEE Transactions on*, vol. 16, pp. 1700–1703, Jul. 1, 2006.

[15] LHCb Collaboration, "The LHCb Upgrade I," LHCB-DP-2022-002, 2023.

[16] T. Poikela *et al.*, "The VeloPix ASIC," *Journal of Instrumentation*, vol. 12, no. 01, p. C01070, Jan. 2017.

[17] E. Buchanan *et al.*, "Spatial resolution and efficiency of prototype sensors for the LHCb VELO Upgrade," *Journal of Instrumentation*, vol. 17, no. 06, P06038, Jun. 2022.

[18] LHCb Collaboration, "LHCb VELO Upgrade Technical Design Report," CERN, Geneva, LHCB-TDR-013, 2013.

[19] LHCb Collaboration, "LHCb Tracker Upgrade Technical Design Report," CERN, Geneva, LHCB-TDR-015, 2014.

[20] LHCb Collaboration, "LHCb PID Upgrade Technical Design Report," CERN, Geneva, LHCB-TDR-014, 2013.

[21] "LHCb calorimeters: Technical Design Report," CERN, Geneva, LHCb-TDR-2, 2000.

[22] "LHCb muon system: Technical Design Report," CERN, Geneva, LHCb-TDR-4, 2001.

[23] S. Cadeddu *et al.*, "LHCb reoptimized detector design and performance : Technical Design Report," CERN, CERN-LHCC-2003-030, 2003.

[24] G. Bassi *et al.* "A FPGA-based architecture for real-time cluster finding in the LHCb silicon pixel detector." arXiv: arXiv:2302.03972. (Feb. 8, 2023), preprint.

[25] G. Barrand *et al.*, "GAUDI — A software architecture and framework for building HEP data processing applications," *Computer Physics Communications*, CHEP2000, vol. 140, no. 1, pp. 45–55, Oct. 15, 2001.

[26] R. Aaij *et al.*, "Allen: A High-Level Trigger on GPUs for LHCb," *Computing and Software for Big Science*, vol. 4, no. 1, p. 7, Apr. 30, 2020.

[27] M. Clemencic and B. Couturier, "LHCb Build and Deployment Infrastructure for run 2," *Journal of Physics: Conference Series*, vol. 664, no. 6, p. 062 008, Dec. 2015.

[28] LHCb Collaboration. "LHCb · GitLab," GitLab. (2023), [Online]. Available: https://gitlab.cern.ch/lhcb.

[29] M. Clemencic *et al.*, "The LHCb Simulation Application, Gauss: Design, Evolution and Experience," *Journal of Physics: Conference Series*, vol. 331, no. 3, p. 032 023, Dec. 2011.

[30] V. Gligorov, "Conceptualization, implementation, and commissioning of real-time analysis in the High Level Trigger of the LHCb experiment." arXiv: 1806.10912 [physics.ins-det].

[31] B. Todd, A. Apollonio, L. Ponce, D. Walsh, and C. Roderick, "LHC availability 2017: Proton physics-setting the scence," in *8th Evian Workshop on LHC Beam Operation*, 2019, pp. 35–46.

[32] Cisco Public, "Cisco Visual Networking Index: Forecast and Trends, 2017–2022."

[33] J. Stirling. "Parton Luminosity and cross-section plots." (2013), [Online]. Available: http://www.hep.ph.ic.ac.uk/~wstirlin/plots/plots.html (visited on 02/24/2023).

[34] T. Boettcher, "Allen in the first days of Run 3," LHCb-PROC-2022-010, 2022.

[35] A. Piucci, "The LHCb Upgrade," *Journal of Physics: Conference Series*, vol. 878, no. 1, p. 012 012, Jul. 2017.

[36] A. Bharucha *et al.*, "Implications of LHCb measurements and future prospects," *The European Physical Journal C*, vol. 73, no. 4, p. 2373, Apr. 26, 2013.

[37] R. Aaij *et al.*, "A comprehensive real-time analysis model at the LHCb experiment," *Journal of Instrumentation*, vol. 14, no. 04, P04006–P04006, Apr. 15, 2019. arXiv: 1903.01360 [hep-ex].

[38] "RTA and DPA dataflow diagrams for Run 1, Run 2, and the upgraded LHCb detector," LHCB-FIGURE-2020-016.

[39] LHCb collaboration, "Measurement of the track reconstruction efficiency at LHCb," *Journal of Instrumentation*, vol. 10, no. 02, P02007, Feb. 2015.

[40] R. Aaij *et al.*, "Selection and processing of calibration samples to measure the particle identification performance of the LHCb experiment in Run 2," *EPJ Techniques and Instrumentation*, vol. 6, no. 1, pp. 1–16, 1 Dec. 2019.

[41] R. Aaij *et al.*, "A Comparison of CPU and GPU Implementations for the LHCb Experiment Run 3 Trigger," *Comput. Softw. Big Sci.*, vol. 6, no. 1, p. 1, 2022.

[42] A. Scarabotto, "Tracking on GPU at LHCb's fully software trigger," CERN-LHCb-PROC-2022-011, 2022.

[43] LHCb Collaboration, "LHCb Upgrade GPU High Level Trigger Technical Design Report," CERN, LHCB-TDR-021, 2020.

[44] D. H. C. Pérez, N. Neufeld, and A. R. Núñez, "Search by triplet: An efficient local track reconstruction algorithm for parallel architectures," *Journal of Computational Science*, vol. 54, p. 101 422, Sep. 2021. arXiv: 2207.03936 [hep-ex].

[45] "Run3 Primary Vertex reconstruction description and performance," LHCB-FIGURE-2020-005, 2020.

[46] P. F. Declara, D. H. C. Pérez, J. Garcia-Blas, D. vom Bruch, J. D. Garcia, and N. Neufeld, "A parallel-computing algorithm for high-energy physics particle tracking and decoding using GPU architectures," *IEEE Access*, vol. 7, pp. 91 612–91 626, 2019. arXiv: 2002.11529 [hep-ex, physics:physics].

[47] B. K. Jashal, "Standalone track reconstruction and matching algorithms for GPU-based High level trigger at LHCb," LHCb-PROC-2022-013, 2022.

[48] P. Billoir, M. De Cian, P. A. Günther, and S. Stemmle, "A parametrized Kalman filter for fast track fitting at LHCb," *Computer Physics Communications*, vol. 265, p. 108 026, Aug. 2021. arXiv: 2101.12040 [hep-ex, physics:physics].

[49] L. Anderlini *et al.*, "Muon identification for LHCb Run 3," *Journal of Instrumentation*, vol. 15, no. 12, T12005–T12005, Dec. 10, 2020. arXiv: 2008.01579 [hep-ex, physics:physics].

[50] O. Kitouni, N. Nolte, and M. Williams. "Robust and Provably Monotonic Networks." arXiv: arXiv:2112.00038. (Nov. 30, 2021), [Online]. Available: http://arxiv.org/abs/2112.00038 (visited on 03/11/2023), preprint.

[51] N. V. Canudas, M. C. Gómez, X. Vilasís-Cardona, and E. G. Ribé, "Graph Clustering: A graph-based clustering algorithm for the electromagnetic calorimeter in LHCb," *The European Physical Journal C*, vol. 83, no. 2, p. 179, Feb. 25, 2023.

[52] M. Adinolfi *et al.*, "Performance of the LHCb RICH detector at the LHC," *The European Physical Journal C*, vol. 73, no. 5, p. 2431, May 2013. arXiv: 1211.6759 [hep-ex, physics:physics].

[53] R. Calabrese *et al.*, "Performance of the LHCb RICH detectors during LHC Run 2," *Journal of Instrumentation*, vol. 17, no. 07, P07013, Jul. 2022.

[54] R. Aaij *et al.*, "Tesla : An application for real-time data analysis in High Energy Physics," *Computer Physics Communications*, vol. 208, pp. 35–42, Nov. 2016. arXiv: 1604.05596 [hep-ex, physics:physics].

[55] LHCb Collaboration, "Computing Model of the Upgrade LHCb experiment," LHCB-TDR-018, 2018.

[56] N. Nolte, "A Selection Framework for LHCb's Upgrade Trigger," Ph.D. dissertation, Tech. U., Dortmund, 2020, 139 pp.

[57] S. Esen, A. M. Hennequin, and M. De Cian, "Fast and flexible data structures for the LHCb Run 3 software trigger," LHCb-PROC-2022-012, 2022.

[58] F. Reiss, "Real-time alignment procedure at the LHCb experiment for Run 3," LHCb-PROC-2023-001, 2023.

[59] D. Bakhvalov, *Performance Analysis and Tuning on Modern CPUs: Squeeze the Last Bit of Performance from Your Application.* Independently Published, Nov. 16, 2020, 237 pp. Google Books: oJUfzgEACAAJ.

[60] Daniel Lemire. "Daniel Lemire's blog," Daniel Lemire's blog. (2023), [Online]. Available: http://lemire.me/blog.

[61] "CppCon | The C++ Conference." (2022), [Online]. Available: https://cppcon.org/.

[62] Bjarne Stroustrup and Herb Sutter. "C++ Core Guidelines." (2023), [Online]. Available: https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines.

[63] R. Matev, N. Nolte, and A. Pearce, "Configuration and scheduling of the LHCb trigger application," *EPJ Web of Conferences*, vol. 245, p. 05 004, Jan. 1, 2020.

[64] M. Clemencic, B. Hegner, and C. Leggett, "Gaudi Evolution for Future Challenges," *Journal of Physics: Conference Series*, vol. 898, no. 4, p. 042 044, Oct. 2017.

[65] BeeOnRope. "Answer to "About the branchless binary search"," Stack Overflow. (Jan. 20, 2019), [Online]. Available: https://stackoverflow.com/a/54273248 (visited on 02/08/2023).

[66] A. Hennequin, "Performance optimization for the LHCb experiment," Ph.D. dissertation, Sorbonne Université, Jan. 31, 2022.

[67] W.-C. Ma and C.-L. Yang, "Using Intel Streaming SIMD Extensions for 3D Geometry Processing," in *Proceedings of the Third IEEE Pacific Rim Conference on Multimedia*, Mar. 16, 2003.

[68] R. Frühwirth and A. Strandlie, *Pattern Recognition, Tracking and Vertex Reconstruction in Particle Detectors* (Particle Acceleration and Detection). Cham: Springer International Publishing, 2021.

[69] R. Frühwirth and R. K. Bock, *Data Analysis Techniques for High-Energy Physics Experiments*, H. Grote, D. Notz, and M. Regler, Eds. Cambridge University Press, 2000, vol. 11.

[70] X. Ai *et al.*, "A Common Tracking Software Project," *Computing and Software for Big Science*, vol. 6, no. 1, p. 8, Apr. 13, 2022.

[71] H. Dijkstra, H. J. Hilke, T. Nakada, and T. Ypsilantis, *LHCb Letter of Intent, LHCb Collaboration* (LHCC). 1995.

[72] P. Krizan, R. Mankel, D. Ressing, S. Shuvalov, and M. Spahn, "HERA-B, an experiment to study CP violation at the HERA proton ring using an internal target," *Nucl. Instrum. Meth. A*, vol. 351, pp. 111–131, 1994.

[73] R. Aaij *et al.*, "Updated measurement of time-dependent CP-violating observables in $B_s^0 \to J/\psi K^+K^-$ decays," *The European Physical Journal C*, vol. 79, no. 8, p. 706, Aug. 2019. arXiv: 1906.08356 [hep-ex].

[74] I. Belyaev, G. Carboni, N. Harnew, C. Matteuzzi, and F. Teubert, "The history of LHCb," *The European Physical Journal H*, vol. 46, no. 1, p. 3, Mar. 19, 2021.

[75] P. Li, E. Rodrigues, and S. Stahl, "Tracking Definitions and Conventions for Run 3 and Beyond," LHCb-PUB-2021-005, Feb. 24, 2021.

[76] Particle Data Group *et al.*, "Review of Particle Physics," *Progress of Theoretical and Experimental Physics*, vol. 2022, no. 8, p. 083C01, Aug. 8, 2022.

[77] V. L. Highland, "Some practical remarks on multiple scattering," *Nuclear Instruments and Methods*, vol. 129, no. 2, pp. 497–499, Nov. 15, 1975.

[78] G. R. Lynch and O. I. Dahl, "Approximations to multiple Coulomb scattering," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 58, no. 1, pp. 6–10, May 2, 1991.

[79] A. Lechner, "Particle interactions with matter," *CERN Yellow Rep. School Proc.*, vol. 5, B. Holzer, Ed., p. 47, 2018.

[80] J. Linsley, *The Radiation Length in Air*. France: Commissariat a l'Energie Atomique, 1982.

[81] LHCb collaboration, "Long-lived particle reconstruction downstream of the LHCb magnet," CERN, CERN-LHCb-DP-2022-001, Nov. 20, 2022. arXiv: 2211.10920 [hep-ex].

[82] E. Bowen and B. Storaci, "VeloUT tracking for the LHCb Upgrade," CERN, LHCb-PUB-2013-023, 2014.

[83] R. Aaij *et al.*, "Design and performance of the LHCb trigger and full real-time reconstruction in Run 2 of the LHC," *Journal of Instrumentation*, vol. 14, P04013–P04013, Apr. 24, 2019.

[84] A. Hennequin, B. Couturier, V. V. Gligorov, S. Ponce, R. Quagliani, and L. Lacassagne, "A fast and efficient SIMD track reconstruction algorithm for the LHCb upgrade 1 VELO-PIX detector," *Journal of Instrumentation*, vol. 15, no. 06, P06018, Jun. 2020.

[85] S. Aiola *et al.*, "HybridSeeding: A standalone track reconstruction algorithm for scintillating fibre tracker at LHCb," *Computer Physics Communications*, vol. 260, p. 107 713, Mar. 2021. arXiv: 2007.02591 [hep-ex, physics:physics].

[86] A. Davis, M. De Cian, A. M. Dendek, and T. Szumlak, "PatLongLivedTracking: A tracking algorithm for the reconstruction of the daughters of long-lived particles in LHCb," CERN-LHCb-PUB-2017-001, 2017.

[87] P. A. Günther. "LHCb's Forward Tracking algorithm for the Run 3 CPU-based online track-reconstruction sequence." arXiv: arXiv:2207.12965. (Oct. 31, 2022), preprint.

[88] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1, 1960.

[89] R. Frühwirth, "Application of Kalman filtering to track and vertex fitting," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 262, no. 2, pp. 444–450, Dec. 15, 1987.

[90] E. Bos and E. Rodrigues, "The LHCb Track Extrapolator Tools," CERN-LHCb-2007-140, 2007.

[91] LHCb Collaboration, "Selected HLT2 reconstruction performance for the LHCb upgrade," LHCB-FIGURE-2021-003.

[92] M. Benayoun and O. Callot, "The forward tracking, an optical model method," LHCb-2002-008, 2002.

[93] O. Callot and S. Hansmann-Menzemer, "The Forward Tracking: Algorithm and Performance Studies," CERN-LHCb-2007-015, 2007.

[94] Y. Amhis, O. Callot, M. De Cian, and T. Nikodem, "Description and performance studies of the Forward Tracking for a scintilating fibre detector at LHCb," CERN-LHCb-PUB-2014-001, Apr. 22, 2014.

[95] P. V. C. Hough, "Machine Analysis of Bubble Chamber Pictures," *Conf.Proc.C*, vol. 590914, pp. 554–558, 1959.

[96] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, Third edition. Cambridge, Massachusetts ; London, England: MIT Press, 2009, xix, 1292 Seiten.

[97] Agner Fog. "Software optimization resources. C++ and assembly. Windows, Linux, BSD, Mac OS X." (2022), [Online]. Available: https://www.agner.org/optimize/ (visited on 02/15/2023).

[98] O. Green, S. Odeh, and Y. Birk. "Merge Path - A Visually Intuitive Approach to Parallel Merging." arXiv: arXiv:1406.2628. (Jun. 20, 2014), preprint.

[99] A. Watkins and O. Green, "A Fast and Simple Approach to Merge and Merge Sort Using Wide Vector Instructions," in *2018 IEEE/ACM 8th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*, Nov. 2018, pp. 37–44.

[100] H. Inoue and K. Taura, "SIMD- and cache-friendly algorithm for sorting an array of structures," *Proceedings of the VLDB Endowment*, vol. 8, pp. 1274–1285, Jul. 1, 2015.

[101] A. Krishnamoorthy and D. Menon, "Matrix inversion using Cholesky decomposition," in *2013 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, Sep. 2013, pp. 70–72.

[102] A. Hocker *et al.*, "TMVA - Toolkit for Multivariate Data Analysis," Mar. 2007.

[103] W. Adam, R. Frühwirth, A. Strandlie, and T. Todorov, "Reconstruction of electrons with the Gaussian-sum filter in the CMS tracker at LHC," *Journal of Physics G: Nuclear and Particle Physics*, vol. 31, no. 9, N9–N20, Sep. 1, 2005. arXiv: physics/0306087.

[104] S. Slotin. "Static B-Trees - Algorithmica." (2022), [Online]. Available: https://en.algorithmica.org/hpc/data-structures/s-tree/ (visited on 03/02/2023).

[105] C. Agapopoulou, "Commissioning LHCb's GPU high level trigger," *Journal of Physics: Conference Series*, vol. 2438, no. 1, p. 012 017, Feb. 2023.

[106] R. Aaij *et al.*, "Measurement of forward J/ψ production cross-sections in pp collisions at $\sqrt{s} = 13$tev," *Journal of High Energy Physics*, vol. 2015, no. 10, p. 172, Oct. 2015. arXiv: 1509.00771 [hep-ex].

[107] R. Aaij *et al.*, "Measurements of prompt charm production cross-sections in pp collisions at $\sqrt{s} = 13\,\text{TeV}$," *Journal of High Energy Physics*, vol. 2017, no. 5, p. 74, May 2017. arXiv: 1510.01707 [hep-ex].

[108] R. Aaij *et al.*, "Measurements of prompt charm production cross-sections in pp collisions at $\sqrt{s} = 5\,\text{TeV}$," *Journal of High Energy Physics*, vol. 2017, no. 6, p. 147, Jun. 2017. arXiv: 1610.02230 [hep-ex].

[109] R. Aaij *et al.*, "Observation of double charm production involving open charm in pp collisions at $\sqrt{s} = 7\,\text{TeV}$," *Journal of High Energy Physics*, vol. 2012, no. 6, p. 141, Jun. 2012. arXiv: 1205.0975 [hep-ex].

[110] I. Helenius and H. Paukkunen, "Double D-meson production in proton-proton and proton-lead collisions at the LHC," *Physics Letters B*, vol. 800, p. 135 084, Jan. 2020. arXiv: 1906.06971 [hep-ph].

[111] G. Cowan, *Statistical Data Analysis*. Clarendon Press, Mar. 26, 1998, 214 pp. Google Books: D_v4ckw24gwC.

[112] H. Fischer and H. Kaul, *Mathematik für Physiker Band 1: Analysis, Lineare Algebra, Vektoranalysis, Funktionentheorie* (Springer eBook Collection), 8. Aufl. 2018. Berlin, Heidelberg: Springer Spektrum, 2018, Online-Ressource (XII, 575 S, online resource).

[113] P. A. Günther. "Reco-Parametrisation-Tuner." (2022), [Online]. Available: https://gitlab.cern.ch/gunther/prforwardtracking-parametrisation-tuner/-/tree/master (visited on 04/05/2023).

[114] Pedregosa, F. et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[115] M. Brice, "The LHCb beam pipe," CERN-PHOTO-202204-063-3, 2022.

Self References

P. Billoir, M. De Cian, P. A. Günther, and S. Stemmle, "A parametrized Kalman filter for fast track fitting at LHCb," *Computer Physics Communications*, vol. 265, p. 108 026, Aug. 2021. arXiv: 2101.12040 [hep-ex, physics:physics].

P. A. Günther. "Reco-Parametrisation-Tuner." (2022), [Online]. Available: https://gitlab.cern.ch/gunther/prforwardtracking-parametrisation-tuner/-/tree/master (visited on 04/05/2023).

P. A. Günther. "LHCb's Forward Tracking algorithm for the Run 3 CPU-based online track-reconstruction sequence." arXiv: arXiv:2207.12965. (Oct. 31, 2022), preprint.