Faculty of Physics and Astronomy

University of Heidelberg

Diploma thesis
in Physics

submitted by
Manuel Tobias Schiller
born in Heidelberg

July 2007

Standalone track reconstruction for the Outer Tracker of the LHCb experiment using a cellular automaton

This diploma thesis has been carried out by Manuel Tobias Schiller at the
Physikalisches Institut
under the supervision of
Prof. Dr. Ulrich Uwer

Kurzfassung

In dieser Diplomarbeit wird die Leistungsfähigkeit der Mustererkennung im Spursystem studiert. Zuerst wird der gegenwärtige Stand existierender Algorithmen zusammengefasst. Dann werden Zelluläre Automaten und ihre Anwendung zur Spurfindung detailliert diskutiert. Der Prototyp einer Implementierung eines solchen Algorithmus für das äußere Spurkammersystem vom LHCb erweist sich als schnelle und effiziente Alternative zu existierenden Algorithmen. Seine Leistungsfähigkeit mit über 93 Prozent Effizienz und einem Anteil von 12 Prozent fehlrekonstruierten Spuren ist vergleichbar mit den bereits existierenden Algorithmen. Bei hoher Belegunsdichte im Detektor erweist er sich aber als deutlich robuster.

Abstract

In this thesis, the performance of the pattern recognition in the LHCb tracking system is studied. First, the status of the existing algorithms is summarised. Then, the concept of the cellular automaton and its application in track reconstruction is discussed in detail. A prototype implementation of such an algorithm for the LHCb Outer Tracker proves to be a fast and efficient alternative to existing algorithms. Its performance of above 93 percent efficiency with a fraction of 12 percent wrongly reconstructed tracks is comparable with existing algorithms. It is significantly more robust with increasing detector occupancy.

Contents

Introduction

Physics aims at describing our world on scales of very different orders of magnitude. The observations range from distances as large as the radius of the universe of about 10^{26} m down to scales as tiny as 10^{-18} m which is the length down to which the electron substructure has been probed to be point-like.

Particle physics is concerned with observations on the end of extremely tiny scales, attempting to explain them and derive predictions. Usually, such observations are made using collisions in particle accelerators which effectively act as "microscopes" to the sub-atomic world. Models capturing the present understanding of particle physics are devised and used to predict future observations, putting them to test. A good model should be as simple as possible.

The so-called Standard Model of Particle Physics was formulated about three decades ago and describes the interaction of the constituents of matter through the electro-magnetic, weak and strong forces on the subatomic scale. It has proven to withstand scientific scrutiny and experimental evidence collected during these three decades almost unchanged, making it one of the most widely accepted and long-lived models in the history of particle physics.

While predictions made by the Standard Model have been found to be in excellent agreement with experiment, it leaves open a number of questions. For example, it provides a mechanism to generate an asymmetry between particles and antiparticles which has been tested at collider experiments. However, it is unable to explain the amount of matter-antimatter asymmetry seen in the universe which is several orders of magnitude larger than the Standard Model prediction.

The fact that the Standard Model describes the data observed in experiments so far, yet leaves unanswered very basic questions is usually interpreted as evidence that there must be physics beyond the Standard Model, at energies higher than those probed by past and present experiments. Therefore, new particle accelerators are being built, colliding elementary particles like

electrons or protons with ever-increasing energies. At the same time, the resolution of the detectors and the statistical size of the data samples are increased to improve measurement precision so that even small deviations from the Standard Model can be spotted in the search for signs of new physics.

The Large Hadron Collider (LHC), a proton-proton collider currently being completed in the tunnel of the former Large Electron Positron collider (LEP) at the Organisation Européen pour la Recherche Nucléaire (CERN) in Geneva, will be the dominating accelerator at the high-energy frontier in the next decade. It is due to start operation around the end of the year 2007 and will deliver collisions at a centre-of-mass energy of 14 TeV when fully operational.

One of the six experiments[1] located along the 27 km circumference of the tunnel is the Large Hadron Collider beauty experiment (LHCb). It seeks to gain a more thorough understanding of the particle-antiparticle asymmetry in the B-meson sector by performing precision measurements. Such precision measurements require the reconstruction of tracks of particles produced in collisions with high efficiency and very good spatial accuracy. This task, commonly referred to as tracking, is challenging, especially in the high track multiplicity environment generated by proton-proton collisions as delivered by the LHC. Figure 1 shows the reconstructed tracks of a typical simulated event.

In addition to the issue of the large combinatorial burden, the tracking has to meet severe timing constraints to make it possible to process the vast amount of data produced at LHCb. The CPU time available to the tracking at trigger level is even more restricted. Typically the tracking has to be faster than 1 ms per subdetector.

For this thesis, a tracking approach based on cellular automata is investigated, a technique that has been proven to be efficient and fast in previous experiments. A prototype implementation has been written for the LHCb Outer Tracker (see Chapter 1 for a description of the detector) which is presented here.

The remainder of the thesis is structured as follows: Chapter 1 briefly describes the LHCb experiment. Terms, definitions and some of the techniques commonly used in tracking are described in Chapter 2. In Chapter 3, the performance of the LHCb tracking in its present state is presented. Chapter 4 gives a brief introduction to cellular automata in general and outlines their application in tracking, while Chapter 5 describes the implementation written for this thesis. Chapter 6 investigates the performance of the algorithm presented and attempts a comparison with the standard package. A

[1] These six experiments are ALICE, ATLAS, CMS, LHCb, LHCf and TOTEM.

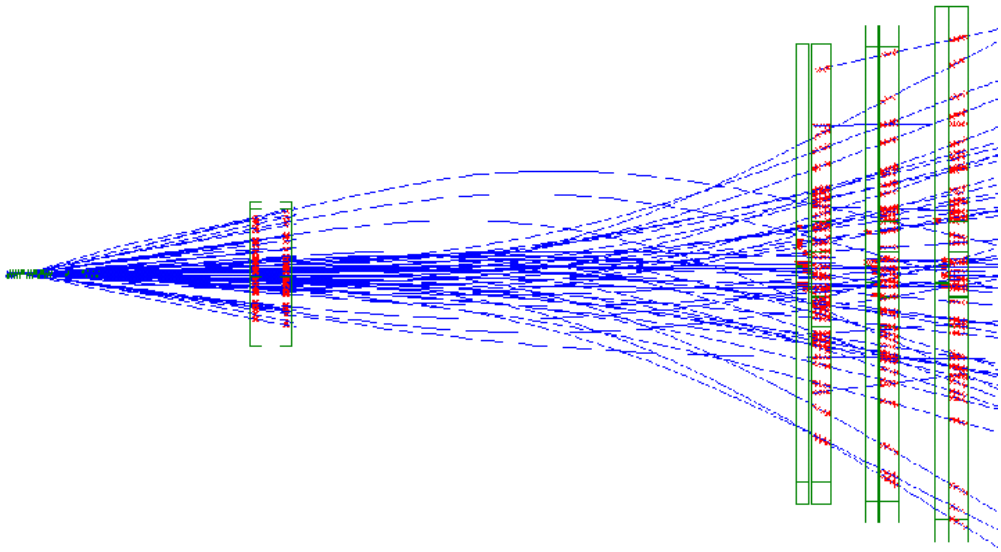


Figure 1: *Reconstructed tracks of a typical simulated proton-proton collision in the LHCb detector.*

summary of the results of this thesis is given in Chapter 7.

Chapter 1

The LHCb experiment

The LHCb experiment ([1], [2]) is designed to study properties of B mesons, with emphasis on precision measurements of CP violating processes and rare decays. It will run at $\sqrt{s} = 14$ TeV using the p-p collisions provided by the LHC. To perform precision B physics analysis, it is crucial to reduce the number of multiple interactions. Therefore, the beam is defocused before entering the LHCb experiment, reducing the average number of multiple interactions per bunch crossing from 17 to 1.5. The luminosity is also decreased from 10^{34} cm^{-2}s^{-1} to $2 \cdot 10^{32}$ cm^{-2}s^{-1}.

CP violating processes are known in the standard model and have been observed in kaon and beauty systems, although the precision of existing measurements for the beauty system is not as good as what LHCb is expected to achieve. The CKM mechanism which is the only source of CP violation in the Standard Model is several orders of magnitude too weak to explain the imbalance observed in the universe. LHCb will help improve the understanding of CP asymmetry in the standard model and may be able to discover evidence of CP-violating contributions from new physics.

As the $b\bar{b}$ pairs of interest in the experiment are emitted mostly either in forward or backward direction, LHCb has been designed as a forward spectrometer. Figure 1.1 shows a side view of the detector. It consists of:

- **Velo**: The Vertex Locator (Velo) is a silicon microstrip detector designed to identify primary and secondary vertices with good precision. This is necessary to recognise secondary vertices from B decays.

- **RICH1**: RICH1 is a Ring Imaging Cherenkov detector designed to provide particle identification for low-momentum particles. It contains two radiators covering a momentum range up to 10 GeV and from 10 GeV to 60 GeV, respectively.

- **Trigger Tracker**: The Trigger Tracker (TT) is a silicon microstrip detector located in the fringe field in front of the dipole magnet. It provides an initial momentum estimate for trigger applications.
- **Dipole magnet**: The dipole magnet permits access to the momentum of charged particles by measuring their deflection in its field. Particles traversing the magnet can see an integrated field of up to 4.2 Tm.
- **Tracking system**: The main tracking system (T) consists of Inner and Outer Tracker. The Inner Tracker is a silicon microstrip detector, the Outer Tracker is based on the drift chamber principle, realised in modular straw tube technology.
- **RICH2**: A second Ring Imaging Cherenkov Counter provides particle identification for high momentum charged particles up to 100 GeV.
- **ECAL and HCAL**: The Electromagnetic and Hadronic Calorimeters measure energies of electrons, photons and hadrons.
- **Muon stations**: The muon stations (M1-M5) identify muons.

The tracking system is considered in more detail, with special emphasis on the Outer Tracker because it is relevant to the reconstruction algorithm presented in Chapter 5.

1.1 Vertex Locator

The Velo ([3], [2]) consists of 21 stations which are in turn composed of two modules enclosing the beam pipe from both sides. Two half-disc-shaped silicon sensors mounted back to back form a module. One sensor type measures the radial distance of particle traversal (called r type sensors), the other the azimuthal angle (also called ϕ type sensors). Figure 1.2 contains a drawing illustrating the two Velo sensor types.

1.2 Trigger Tracker

The Trigger Tracker ([4], [2]) consists of two stations, each of which consists of two layers. The first layer of the first station and the last of the second one measure the x coordinate of a passing charged particle, while the two layers in the middle have their measurement direction rotated by $\pm 5\,^\circ$ around the z axis with respect to the x axis. These rotated layers are called u and v layers. The spatial resolution is about $50\,\mu$m in x, u and v direction. This design

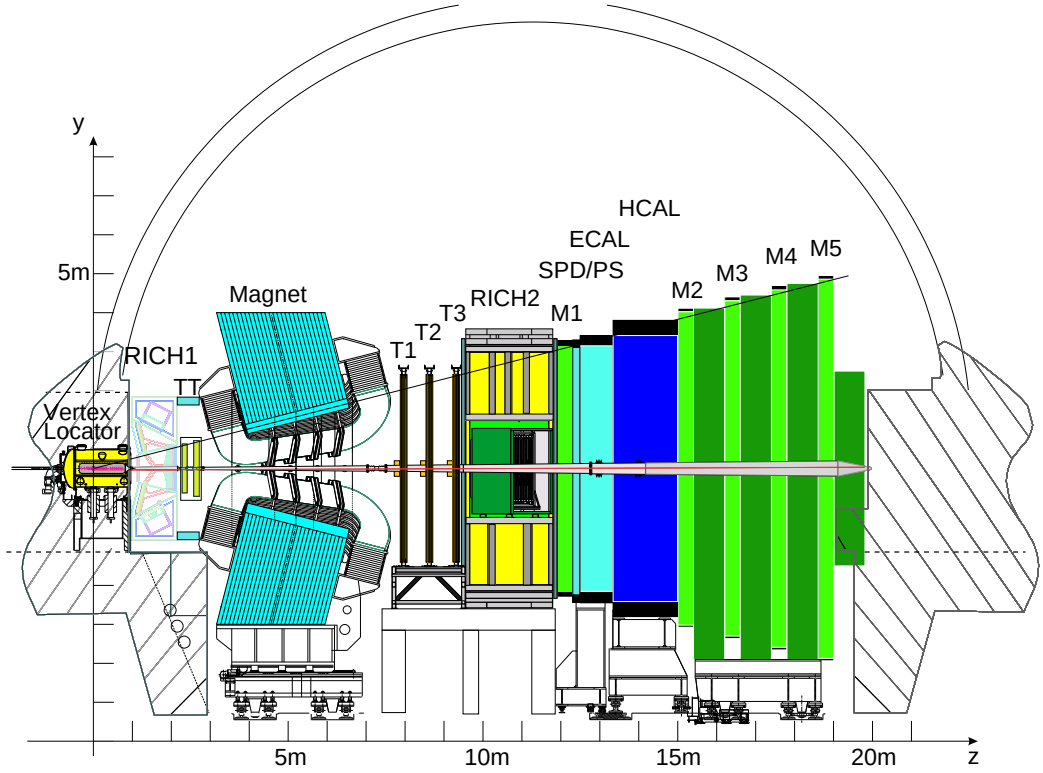


Figure 1.1: *Side view of the LHCb detector. The positive z axis points in the direction of the proton beam (to the right in the picture), the y axis up, and the x axis points into the drawing plane.*

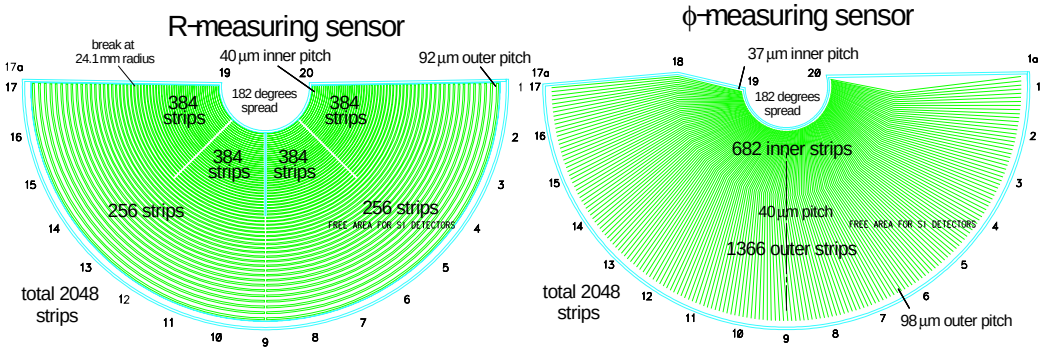


Figure 1.2: *Schematic view of the two types of Velo sensors. The one shown on the left measures the radial distance of a particle to the beam, the right one measures the azimuthal angle.*

permits inferring the y coordinate of a track from the three projections while maintaining excellent resolution in x direction in which most of the deflection in the magnetic field happens. Figure 1.3 contains a sketch of the TT layout.

1.3 Inner Tracker

The Inner Tracker (IT, see [4], [2]) is used to obtain position measurements of charged particles behind the magnet. It covers the area around the beam pipe where the highest particle fluxes are expected. It consists of three stations. Each of these stations contains four layers in an $xuvx$ configuration, just like in the Trigger Tracker. The resolution in x, u and v direction is about 50 μm. Figure 1.4 sketches an x layer.

1.4 Outer Tracker

Just like the Inner Tracker, the Outer Tracker (OT, see [5], [2]) consists of three stations, each with four layers in a $xuvx$ configuration as shown in Figure 1.5. The expected resolution is 200 μm in x, u and v direction.

Each OT layer consists of so-called straw tube modules. The straw tubes contained therein are essentially small drift chambers. In the centre of each straw, a wire is kept at about 1.5 kV positive potential while the conductive-coated wall of the straw is tied to ground. The straws themselves are filled with a counting gas[1]. When a charged particle passes through a straw, it ionises the counting gas. The primary ionisation electrons are accelerated towards the wire by the electric field, producing secondary ionisation electrons on their way towards the wire. This is especially true in the strong field gradient close to the wire, where the arriving electrons cause a charge avalanche, amplifying the small initial amount of charge produced by the primary ionisation (gas gain).

The charge pulse travels along the wire towards the read-out where it is amplified, shaped and discriminated. The time of arrival of such a pulse above a configurable detection threshold is measured with respect to the bunch clock of the machine. It can be decomposed into

$$t_{measured} = t_{TOF} + t_{drift} + t_{prop}$$

where t_{TOF} is the time of flight of the particle to the straw in question, t_{drift} is the time the ionisation electrons drift in the gas, and t_{prop} is the signal propagation time from the moment the charge avalanche hits the wire to the

[1] A mixture of 70 % (of the total volume) Argon and 30 % CO_2 is used.

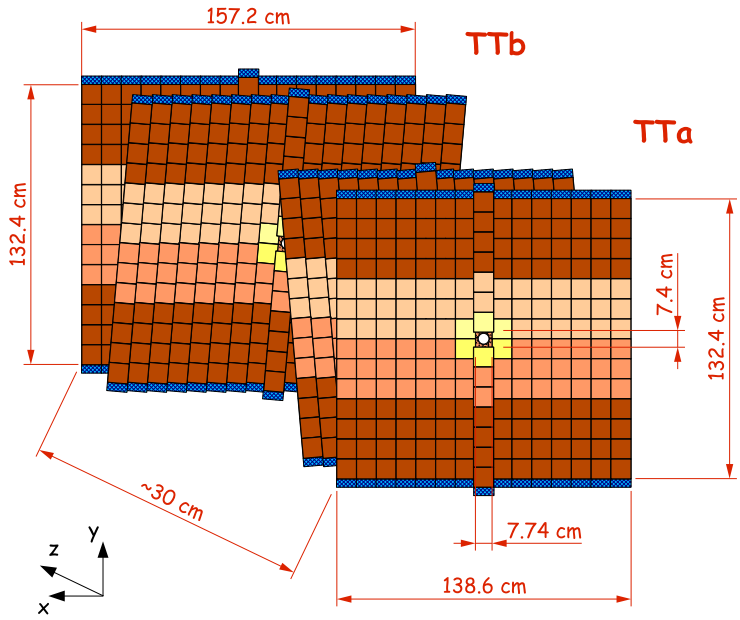


Figure 1.3: *Layout of the TT stations. The two stereo layers in the middle are called u and v layers.*

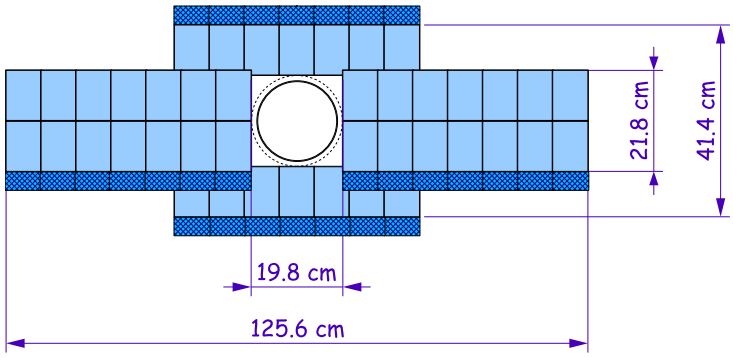


Figure 1.4: *Sketch of an x layer in the IT. It consists of several sensors located around the beam pipe.*

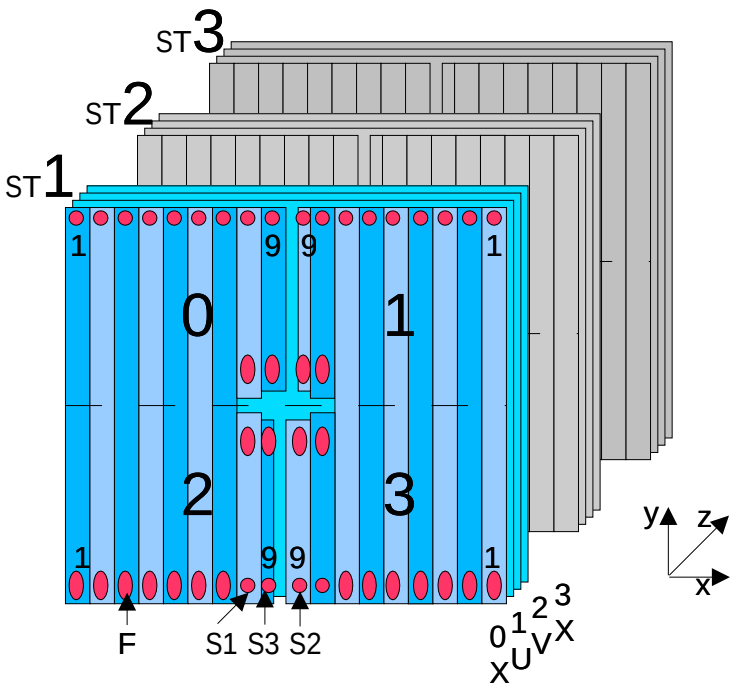


Figure 1.5: *Schematic layout of the OT: It consists of three stations, each of which contains four layers in an xuvx configuration. A layer can further be subdivided into left and right halves which are mounted on aluminium frames that can be retracted from the beam pipe for maintenance. Each half consists of nine modules. Separate readouts for the top and the bottom half of each module make it feasible to partition an OT layer into quarters, as shown in the picture.*

moment it is registered by the electronics. Correcting for t_{TOF} and t_{prop}, it is possible to infer the distance which the electrons had to drift in the straw using a so-called $r-t$ relation which describes the drift radius as a function of t_{drift}. While for a simple model, a constant drift velocity is a usable ansatz, it will have to be determined from data in the experiment because the relation is not exactly linear, especially near the walls of the straw and near the wire.

There are 128 straw tubes per module, further subdivided into two monolayers which have been staggered to compensate for the insensitive area between two straws. Figure 1.6 shows a cross section of such a module. Most modules are five metres long, but there are also four short ones in the middle of each layer to leave space for the beam pipe.

The detector is read out at the top and at the bottom; to do this, the long modules are electrically subdivided in the middle. The drift times are measured by the front end electronics attached to the top and bottom of each module. From there, the data is transmitted in digital form over optical fibres for further processing.

1.4.1 Spillover

There is one further complication: Bunches delivered by the LHC are only 25 ns apart while the drift time in the gas is about 44 ns for a full straw radius. Allowing for time of flight and wire propagation time as well, this means that the data taken during three bunches must be used to retain all the ionisation pulses from a single event. Drift electrons from different events can also arrive in this three bunch window, giving rise to so-called *spillover* measurements. Figure 1.7 sketches the situation. Figure 1.8 shows the distribution of drift time measurements in the simulation, the black curve is for all measurements in the three bunch window, the red one is the part of the spectrum caused by particles from the bunch crossing one is interested in.

The reconstruction has to cope with these spillover measurements in addition to noise and crosstalk.

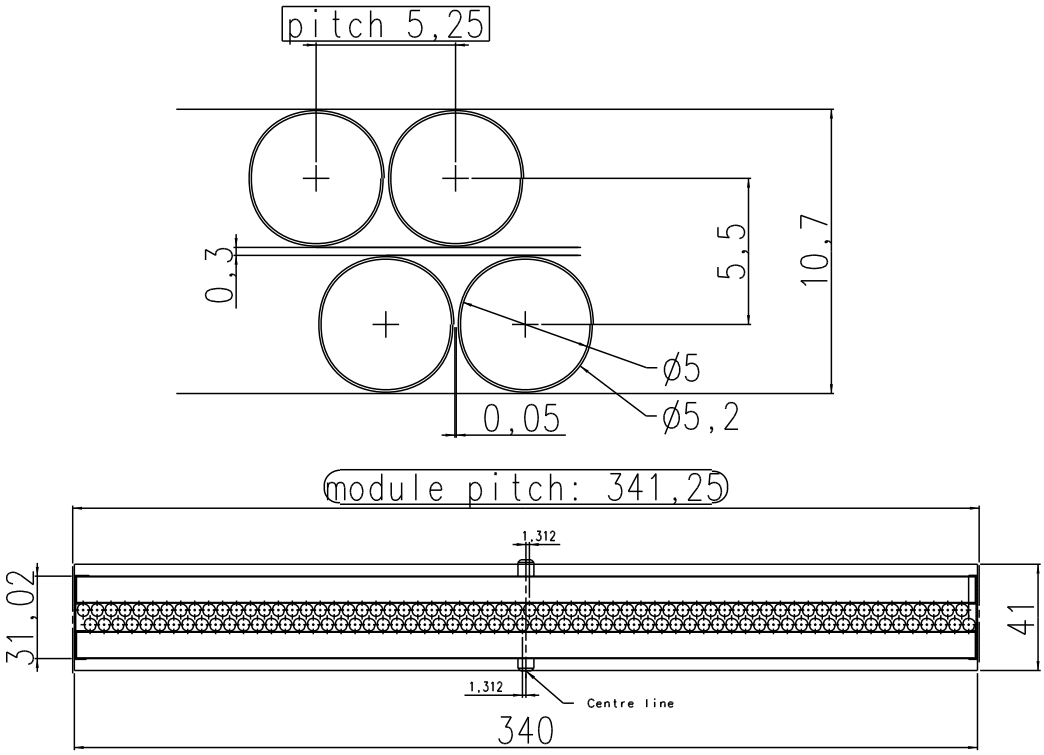


Figure 1.6: *Cross section through a module used in the Outer Tracker. A module consists of two monolayers of straws which are staggered such that the insensitive areas between the straws in two monolayers do not coincide. The upper part of the drawing shows a magnified part, illustrating how straws are mounted in detail, the lower part gives an impression of the overall module dimensions (all lengths are measured in millimetres). A typical module is about 5 metres long.*

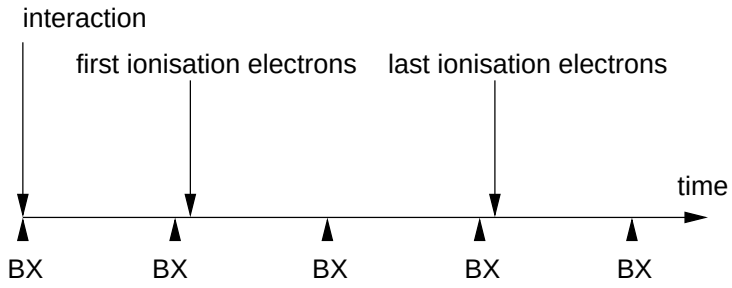


Figure 1.7: *Sketch illustrating the arrival of drift electrons at the wires in the OT with respect to the time when the primary interaction happened. The corresponding bunch crossing is the first one shown; the small arrows labelled "BX" below the time axis also indicate the following ones.*

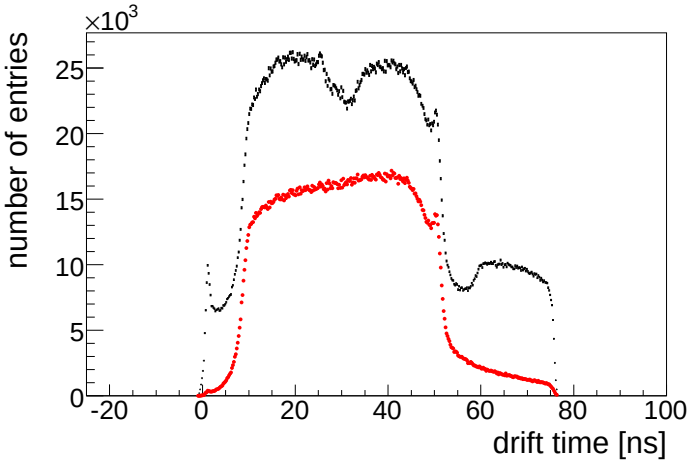


Figure 1.8: *Drift time spectra in the simulation: The black curve shows all measurements inside a three bunch crossing window, the red one is the part of the spectrum caused by particles from the bunch crossing one is interested in. A time of flight correction has been applied to the data. The tail for long times in the red curve may is due to low momentum particles for which the time of flight correction is biased.*

Chapter 2

Tracking in the LHCb experiment

2.1 Introduction

Like most high energy physics experiments, LHCb relies on reconstructing the trajectories of particles produced at the interaction point by proton-proton collisions of the LHC. This is a challenging task, especially in the high track multiplicity environment generated by p-p collisions. In this section, the general ideas underlying the tracking algorithms in the LHCb software are presented.

As usual, there are two distinct stages, pattern recognition and track fitting. The former provides collections of measurements which are likely to form tracks while the latter attempts to give the most accurate estimate of the track parameters.

The focus of this thesis is on pattern recognition, the track fit will only be mentioned briefly and appropriate references are given.

2.2 Track model

The reconstruction needs a software model to represent particle trajectories of interest. Such a track model is used in several places, for example to match a track based on pattern recognition information to RICH rings providing particle identification information, to calculate invariant masses or to find primary and secondary vertices during the analysis stage of the experiment.

The direction of most particle trajectories of interest in the LHCb experiment has large components parallel (or antiparallel) to the z axis. Therefore,

this choice resulted in a track model representing a track as a collection of tangents to the trajectory of a particle at several points along the z axis.

Each of those tangents (and the point in space to which they are attached) is represented as a vector of five real numbers called a *track state* (at a given z):

$$\text{Track state}(z) = \begin{pmatrix} x \\ y \\ t_x \\ t_y \\ q/p \end{pmatrix}$$

Here, $t_x = \partial x/\partial z$ and $t_y = \partial y/\partial z$ are the track slopes in xz and yz projection, and q/p is an estimate of charge divided by momentum for the reconstructed trajectory. It is introduced in addition to the first four components because one wants to be able to extrapolate a track through the field of the magnet if the momentum is known.

To give the most accurate estimate of these parameters along the track, an iterative fitting procedure called Kalman filtering ([6]) is used which is based on minimising the χ^2-contribution of each measurement on the track, thus making it equivalent to a least-squares fit. The advantages of the Kalman formalism are numerous: The method provides iterative updates of the track parameters and their estimated uncertainties for each new measurement added to the fit. This makes it possible to decide to add or not to add a measurement to a track depending on its predicted χ^2 contribution[1], for example. It can also account for noise-like effects[2] like multiple scattering without great difficulty.

The reader is referred to the literature for further details (for example, see [6], [7] or [8]).

2.3 LHCb Track types

It is useful to have names for tracks which satisfy some common criteria, so a few track types are introduced. The names follow the mode of speech used by the collaboration. Figure 2.1 gives a graphical impression of the track types.

[1]The χ^2 contribution of a measurement is the square of the difference between the measurement and its prediction from the track parameters divided by the measurement uncertainty squared.

[2]From the track fit's point of view, multiple scattering increases the uncertainty in the track parameters, just like noise would.

- Velo tracks: tracks which have measurements in the Velo only (in both r and ϕ sensors)
- VeloR tracks: tracks which have measurements in the r sensors of the Velo only (The Velo reconstruction, described in 2.4.1, reconstructs tracks using only r sensors first — if the track is not found in the ϕ sensors, it remains a VeloR track.)
- T track: tracks with measurements in the T stations only
- Long tracks: tracks which go through the whole detector, from Velo to T stations
- Upstream tracks: tracks with hits in the Velo and the TT stations only
- Downstream tracks: tracks with hits in the TT and T stations only

2.4 Pattern recognition

It is the task of the pattern recognition to find the set of measurements that correspond to particles traversing the detector among all measurements.

One very illustrative example of pattern recognition is the tracks to be seen in a bubble chamber or on a photographic plate used in a particle physics experiment. The measurements are either little bubbles in a liquid that can

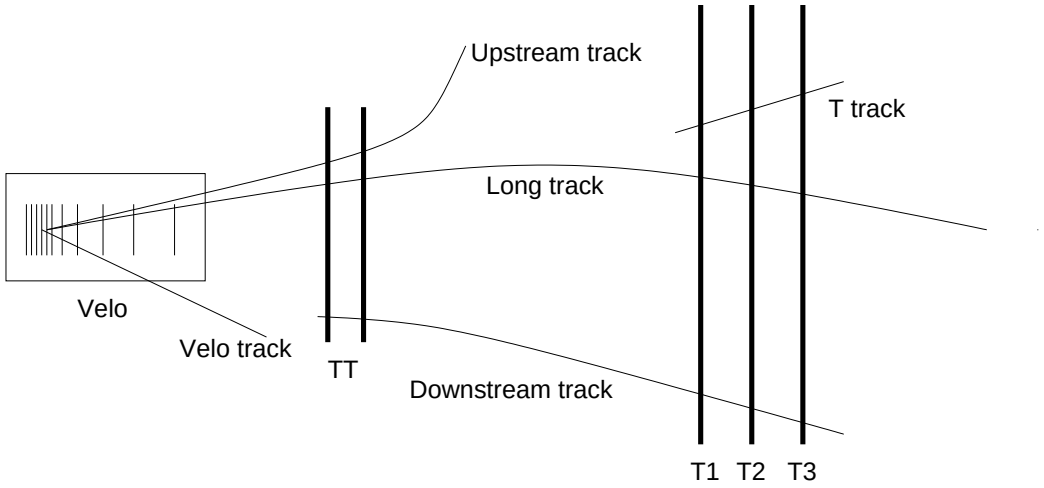


Figure 2.1: *Schematic view of the track types in the LHCb tracking system.*

be observed or metallic silver in very small quantities, rendering the plate black. The human brain does the pattern recognition and lets us see tracks.

In most modern particle physics experiments, the detectors used are more complex, so the pattern recognition works differently as well. In this section, the ideas behind the algorithms which are designed to find Long tracks are described to give an impression how the pattern recognition code works internally. While Long tracks are only a subset of all tracks expected to be found in the experiment, they are the most useful tracks for physics analysis (these are the ones about which one has the most precise information because they leave most measurements in the detector). The techniques used for other track types are similar.

There are two concepts implemented in the LHCb reconstruction software to find Long tracks: One is to start in the Velo, propagating the Velo seeds obtained to the T stations to find Long tracks there, the other is to reconstruct Velo and T stations separately and match the resulting tracks on one side of the magnet to those on the other side. Both have their advantages and their disadvantages, thus complementing each other. In fact, there are several reconstruction algorithms, each implementing one of the two concepts, so that at the moment, there is a wealth of complementarity in the reconstruction software.

In this thesis, the Forward tracking (`PatForward`, see [9]) will be introduced as an example of a strategy starting in the Velo, extending tracks found there to the T stations. The combination of a seeding algorithm (`TsaSeeding`, see [10]) and a track matching algorithm will serve as an example for the second concept mentioned above.

Usually, both strategies will find a track for a given particle. Therefore, it becomes necessary to choose among the two alternatives. It is the task of the `CloneKiller` to recognise such a situation and suppress redundant tracks (see [11] for details).

To introduce the different algorithms, we start with the Velo close to the interaction point and follow the particles through the detector, respecting the dependencies of the algorithms on each other.

2.4.1 Velo tracking

Velo tracking proceeds in two stages: first, tracks are searched for in rz projection (r being distance measured perpendicular to the z axis), then, in a second step, angular information is added using the ϕ sensors and the 2D tracks found in the previous stage (the angle ϕ describes a rotation around the z axis here).

The Velo consists of half-disc shaped sensors (cf. Figure 1.2). Half the

sensors are on the right side of the beam pipe, the other half is on the left side. Each r type sensor is divided into four sectors with each sector covering about 45° of azimuthal angle. By using pairs of r type sensors on the same side of the beam pipe with one or two sensors between them, straight line segments are constructed by combining a cluster in the first sensor with one in the second sensor of the pair. Cluster pairs must be in the same sector to be considered. A line segment is kept if there is a confirming cluster in the intermediate r sensor(s) within a certain search window around that line segment. These line segments are extended, adding additional clusters in other sensors close to the projected position, thus forming tracks in rz space.

The second stage takes these tracks in rz space and attempts to find the corresponding ϕ clusters. This is done by taking all clusters in ϕ sensors compatible with the ϕ range of the rz track. Once again, line segments are formed, this time in ϕz space, from two sensors as far away from the interaction point as possible. These segments are used to add compatible ϕ clusters in the other sensors, just as in the xz case.

In the end, there may be several 3D track candidates for an rz track because of the possibility to have several ϕz tracks in the sector of the rz track. Therefore, one has to choose. This is done by using a "longest track wins" logic (here, "longest" means most ϕ clusters). If one still has to decide among several tracks, the χ^2 (from a straight line fit in 3D) is used to find the best one.

For further details on this method, please see [12].

2.4.2 Momentum estimation

A Velo track does not provide a momentum estimate because there is insufficient magnetic field in the Velo. Information from other detector systems has to be used for this purpose. There are two ways to do this: Either measurements in the Trigger Tracker (TT) located right in front of the magnet are used (tracks have seen a $\int Bd\ell$ of 0.15 Tm), or tracks in the T stations after the magnet are reconstructed and matched to the corresponding Velo part.

As it is difficult to pick TT clusters belonging to a particle based on its Velo track alone[3], the offline reconstruction mainly relies on matching Velo and T parts of a track, adding TT clusters in a second step to improve momentum resolution. The use of Velo and TT alone is still interesting for

[3] Low momentum particles may have scattered in RICH 1 and the amount of their deflection in the fringe field is not precisely known without a momentum estimate. Moreover, there is not much redundancy in the trigger tracker which might help to confirm or reject the TT clusters selected for addition to a Velo track.

application in the trigger where high p_T tracks are of interest and time does not permit to decode the T stations.

There are two strategies which use Velo and T measurements of a particle to get a momentum estimate, Forward Tracking and Track Matching. The former will be described next while the latter, needing tracks in the T stations, will be postponed a little until it has been described how to obtain the tracks behind the magnet.

2.4.3 Forward Tracking

The Forward Tracking algorithm employs a *Hough transform*: Basically, one seeks a method to transform the space of observables (i.e. measurements) to a more abstract space, the *Hough space*. The transformation is picked such that the mutual distances of measurements from the same particle are small in the transformed space[4]. The concept will be illustrated using the Forward Tracking as example.

The idea behind the Forward Tracking is that in the absence of scattering and energy loss, the trajectory of a particle in a magnetic field is completely determined by the equation of the Lorentz force. Only the direction of the particle at one position along its trajectory and its momentum need to be known, or, alternatively, the direction at a position in front the magnet of and a second position behind the magnet. Once these quantities are known, the position and heading of a particle can be calculated at any point along its trajectory.

Given the dipole field of the LHCb magnet, one observes that the linear extrapolations of a Velo track and a T track intersect in xz projection at $z \approx 5300$ mm. This plane is called the bending plane of the magnet.

The algorithm exploits these facts by calculating the x position of the trajectory defined by a Velo track and a T station measurement in the bending plane of the magnet. If the Velo track and a T measurement were produced by the same particle, the x position calculated will be close to the true x position of that particle in the bending plane, if not, the result of the calculation is more random. Thus, the method will group close together the calculated bending plane x positions for T measurements belonging to the same particle, provided that the correct Velo track was used for the calculation.

Hence, T measurements belonging to a particle with a Velo track may be found by selecting the T measurements which produced a cluster of x positions in the bending plane[5].

[4]Depending on the problem at hand, the Hough space and the notion of a distance in that space can be virtually anything.

[5] This is often accomplished by histogramming the data in Hough space and identifying

One can focus on the x component of the trajectory because the main part of the deflection is in that direction. The field acting on the y component is negligible for this discussion.

An effective parametrisation is derived which takes track parameters of a Velo seed and the x and z positions of a measurement in one of the T stations, giving the x position in the bending plane of the magnet. Details on how such an effective parametrisation is obtained and how the algorithm proceeds in detail can be found in [9], for example.

A parametrisation is used instead of starting from first principles because the additional complexity would slow down the algorithm considerably: To obtain the x position in the bending plane from first principles, one needs to integrate numerically the set of equations:

$$\frac{d\vec{p}}{dt} = q\vec{v} \times \vec{B}$$

$$\frac{d\vec{x}}{dt} = \frac{\vec{p}}{\gamma m}$$

Here, $\vec{x}$, $\vec{v}$ and $\vec{p}$ are position, three-velocity and three-momentum of a hypothetical test particle starting from the coordinates and direction of the Velo track, and $\gamma = \frac{1}{\sqrt{1-\vec{v}^2/c^2}}$. From the Velo track alone, only the direction of the momentum of the test particle and not the momentum itself is apparent. Therefore, the integration must be done several times, varying a momentum assumption until the coordinates of the x measurement in the T stations are reproduced.

2.4.4 T station seeding

T station seeding is a strategy to find tracks in the T stations without relying on information from the Velo[6]. These tracks can be extrapolated to other detector components to pick up additional information, e.g. to match them to Velo tracks or to look for particle identification and other information in RICH2, the calorimeters or the muon stations. The algorithm looks for tracks first in xz projection and then in yz projection.

For the description, we focus on the case of the OT because it is more challenging in terms of reconstruction: The cell size is relatively big, an OT

peaks. The algorithm described uses a list of bending plane x positions instead because the histogram based approach can cause problems when the positions calculated for a single particle are in different bins.

[6]There are two algorithms implementing this strategy, `TsaSeeding` and `PatSeeding`. The former will be discussed.

straw is 5 mm in diameter and about 2.5 metres long, the occupancy is high due to spillover[7], and each measurement still has a drift time ambiguity[8].

Searching for tracks in xz projection proceeds as follows: First, one constructs lines between two x measurements in stations T1 and T3. To reduce combinatorics, only pairs of measurements are considered which pass selection cuts, e.g. on the slope t_x of the line (for details, see [10]).

All x measurements in a window around such a line are selected; the size of the window is on the order of the OT cell size as drift times are not taken into account (cf. Figure 2.2). Inside this window, a measurement in T2 is chosen. The measurements chosen in T1, T2 and T3 define a parabola in xz space [9]. There are eight possibilities to resolve ambiguities for the three hits forming the parabola. For each of these combinations, the number of measurements falling into a tighter window around the parabola are counted, this time resolving ambiguities of the remaining measurements towards the parabola. The ambiguity combination which gives most measurements in the second, tighter window is kept (cf. Figure 2.3).

A refit is done during which ambiguities can change if it improves the fit. The point having the largest χ^2 contribution is removed if it is more than 3σ away. The candidates obtained with this procedure have to pass another set of cuts (e.g. on the number of measurements, the slope and the curvature) before they are passed on to the stereo search.

The stereo search uses the track candidate in xz to make a collection of compatible stereo hits (only straws can contribute which are close enough to the candidate in xz, see Figure 2.4). y estimates are calculated for each stereo measurement from the stereo coordinates and the candidate in xz. The stereo search continues in a very similar manner to the x search. The main difference is that the trajectory in yz projection is modelled with a straight line because the effects of the magnetic field in that direction are much weaker. This makes it possible to have more stringent requirements for the hits from which such a straight line is formed: Good combinations still point roughly into the direction of the primary vertex. Of course one has to take into account that the conversion from u/v coordinates to y introduces some additional uncertainty, therefore wider search windows have to be chosen. Ambiguities are resolved just like above, the combination with most measurements in the search window is retained.

Again, some cuts have to be satisfied (e.g. a certain minimum number of

[7] cf Section 1.4.1

[8] It is not clear from the point of view of a measurement on which side of the wire the particle passed.

[9] A parabola was chosen due to the non-negligible fringe field within the T stations, deflecting particles in the xz plane.

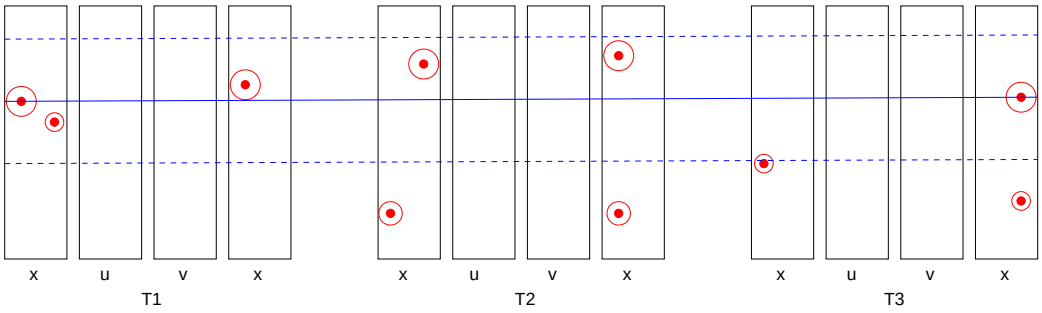


Figure 2.2: *Forming search windows. Wires are indicated with a solid red dot, the red circles are the corresponding drift circles. Using two x measurements, one in T1 and one in T3, a straight line is constructed. All measurements inside a window around the line (dashed lines) are examined further by the algorithm. (Only measurement in x layers are shown. Not to scale.)*

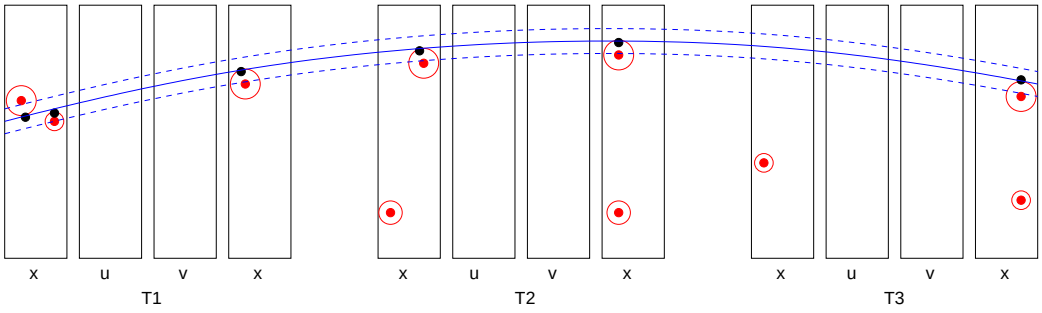


Figure 2.3: *Selecting an x measurement in T2 in addition to those selected in T1 and T3, eight parabolas are constructed, one for each combination of ambiguities of the three selected measurements. One such parabola is shown, the ambiguities chosen are symbolised by black dots. The measurements inside a tighter window around the parabola (dashed lines) are counted. (Only measurement in x layers are shown. Not to scale.)*

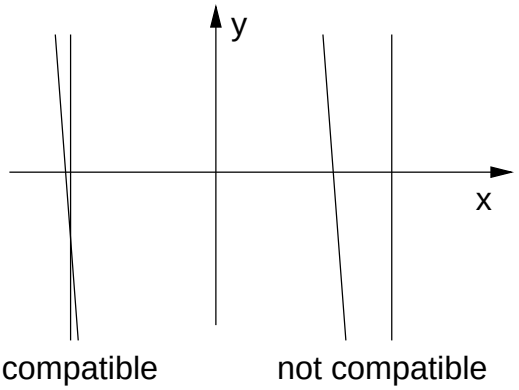


Figure 2.4: *Straws in x and stereo layers are compatible if they overlap in xy projection (left) and incompatible if they do not (right). Since x and stereo layers are separated by a small distance in z, the change in x coordinate of tracks over that distance needs to be taken into account as well.*

stereo hits), otherwise the candidate is not kept.

From the way the algorithm proceeded so far, it is apparent that it may find several candidates for the same particle. There will also be a certain amount of candidates which do not match any physical particle. To eliminate these, an additional selection phase is needed. For this purpose, each track candidate is assigned a likelihood into which several quantities enter, such as the χ^2 probability of the fit and if the candidate was built from all the measurements it should have produced in the detector (based on the assumption that each straw touched by the trajectory defined by the fit parameters also produced a measurement).

The details are a bit more involved; the authors of [10] account for the dependence of the detection efficiency on the drift radius and introduce a weighting of the contributions to the likelihood which improves performance. For details, please refer to that note.

The candidates are sorted by decreasing combined likelihood. The algorithm selects the most likely one, checks that the candidate does not share too many measurements with candidates already selected, and flags its measurements as used. The algorithm continues with candidates with lower likelihood. Candidates failing the cut on the shared measurement fraction or falling below a likelihood threshold are dropped.

2.4.5 Momentum estimation using the p_T-kick method

Assuming that a track in the T stations comes from the primary vertex, it is possible to obtain a momentum estimate from slope change ("kick") in the dipole magnet of the experiment. Figure 2.5 depicts the situation.

The momentum change in a magnetic field is given by

$$\frac{d\vec{p}}{dt} = q\,\vec{v} \times \vec{B}$$

Integrating this along the trajectory of a particle (arc length along the trajectory is denoted by s), one obtains

$$d\vec{p} = q \int_{s_1}^{s_2} \vec{v}(s) \times \vec{B}(s) \frac{dt}{ds} ds$$

Except for very low momentum particles, the initial and final direction of $\vec{v}$ can be estimated using the T track and the line connecting the origin with the extrapolation of the T track into the bending plane of the magnet. The particles of interest are relativistic, so $v \approx c$ is assumed. As x is the main deflection direction of the magnet, one obtains the best accuracy by using the slope change in x direction. Thus,

$$\Delta p_x = p \left(\left(\frac{t_x}{\sqrt{1 + t_x^2 + t_y^2}} \right)_T - \left(\frac{t_x}{\sqrt{1 + t_x^2 + t_y^2}} \right)_{Velo} \right) = q \int_{s_1}^{s_2} (\vec{v}(s) \times \vec{B}(s)) \cdot \vec{e}_x \frac{dt}{ds} ds$$

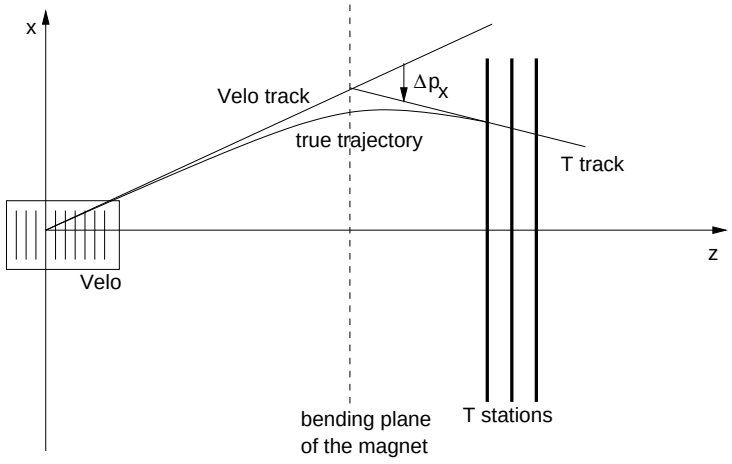


Figure 2.5: *It is possible to obtain a momentum estimate from the "kick" that a particle receives in the field of the magnet.*

where $\vec{e}_x$ is the unit vector in x direction. Solving for p, one obtains

$$p = \frac{q \int_{s_1}^{s_2} (\vec{v}(s) \times \vec{B}(s)) \cdot \vec{e}_x \frac{dt}{ds} ds}{\left(\frac{t_x}{\sqrt{1+t_x^2+t_y^2}}\right)_T - \left(\frac{t_x}{\sqrt{1+t_x^2+t_y^2}}\right)_{Velo}}$$

Details can be found in [13].

2.4.6 Track Matching algorithm

Once tracks in the T stations have been reconstructed, a momentum estimate can be obtained either from the track curvature in the T stations due to the fringe field of the magnet or with the p_T-kick method described above.

Having obtained a momentum estimate, the track is propagated through the magnet using numerical integration. It is matched to a Velo track in a matching plane at $z = 830$ mm. This matching is done using a quantity called χ^2_{match}:

$$\chi^2_{match} = (x_{Velo} - x_T)^T (C_{Velo} + C_T)^{-1} (x_{Velo} - x_T)$$

Here, x_{Velo} and x_T are the track parameters of the Velo track and the extrapolation of the T track into the matching plane, C_{Velo} and C_T are the corresponding covariance matrices. The T station part of the track is fitted using the Kalman Fitter before the Matching algorithm runs, so x_T and C_T are available. For x_{Velo} and C_{Velo}, one has to rely on estimates provided by the Velo pattern recognition. So in principle, χ^2_{match} just measures how well the two sets of track parameters agree in the matching plane. If the pair passes selection cuts, a single Long track is formed from the two tracks. A search for compatible TT clusters follows. Details can be found in [14].

2.5 Tracking performance indicators

To assess the quality of the pattern recognition and tracking algorithms used in the experiment, there are several performance indicators which can be studied[10]. This section gives the general ideas and concepts, Section 2.6 will give exact definitions as used in the experiment.

[10]This is usually done in a Monte Carlo simulation where the "correct" solution to the pattern recognition problem is known.

2.5.1 Reconstruction efficiency

An important question in track reconstruction is clearly how many particle tracks are reconstructed for particles in the detector acceptance.

Let F be the set of particles which was actually reconstructed (or "found") and let R be the set of particles that is reconstructible, i.e. the set of particles which is expected to be reconstructed. Then the reconstruction efficiency is defined as

$$\varepsilon = \frac{\#(F \cap R)}{\#R}$$

Here and in the following text, the symbol # denotes the operator which returns the number of elements in the set following it, i.e. $\#\{17, 42, 78\} = 3$. The exact definitions of "reconstructible" and "reconstructed" will be given below (c.f. Section 2.6), for now, only the general idea matters.

2.5.2 Ghost fraction

Apart from tracks that originate from particles travelling through the detector, tracks may be reconstructed which have no connection to a particle in the detector. These so-called *ghost tracks* may consist of measurements from several different particles, noise and spillover measurements[11] which happen to look like a track to the pattern recognition.

The ghost fraction is the fraction of such tracks among all reconstructed tracks:

$$\text{ghost fraction} = \frac{N_{\text{ghost}}}{N_{\text{total}}}$$

Here, N_{total} is the total number of tracks and N_{ghost} is the number of ghost tracks.

2.5.3 Clone fraction

The clone fraction measures how often you find several tracks for a given particle. The track containing most correct measurements is considered as the best, the others are clones.

There are two definitions of a clone, one involves Monte Carlo truth information, the other does not and is therefore more widely applicable (see [15] for details):

- A track is a *clone* if it shares more than 70% of its measurements in both the Velo and the T stations with another track. If the track in

[11] cf. Section 1.4.1

question does not have hits in one of these detectors, the requirement for that subdetector is dropped.

- A track is a *MC-clone* if more than one track is matched[12] to the same Monte Carlo particle.

The clone fraction is then defined as

$$\text{clone fraction} = \frac{N_{\text{clones}}}{N_{\text{tracks}}}$$

Currently, the reconstruction software runs more than one reconstruction algorithm for some subdetectors. Therefore, a particle can be found more than once in a subdetector[13] , and a dedicated clone killing algorithm is used to select the best alternative. As the clone killer will have to work with real data, the first clone definition has to be used.

This thesis is dealing with the development of an algorithm using a Monte Carlo simulation, so the MC-clone definition will be used to make plots, but clone recognition inside the algorithm will be based on criteria similar to those used in the Monte Carlo-independent definition.

2.5.4 Purity

In the ideal case, a reconstructed track would only consist of measurements of the particle that produced the track. This is fairly optimistic, especially in cases of high detector occupancy. A certain amount of measurements has to be tolerated which do not belong to the particle which produced the track. If the fraction of measurements belonging to the particle in question is not too low, it is still possible derive the track parameters with reasonable accuracy, but the aim is clearly to have tracks as pure as possible.

Thus, the *purity* of a given track is defined as

$$\text{purity} = \frac{N_{\text{correct}}}{N_{\text{total}}}$$

where N_{correct} is the number of measurements that originate from the particle which produced the track and N_{total} is the total number of measurements on that specific track.

[12]The matching is described in 2.6.1.

[13]In fact, even when running only one single reconstruction algorithm per subdetector, clones remain possible: In some cases, an algorithm might still find more than one track per particle.

2.5.5 Collection efficiency

Apart from the considerations above, it is also interesting to ask how efficiently the measurements are used to find a track. An algorithm which only uses half the measurements produced by a particle to estimate track parameters will usually not perform as well as an algorithm which manages to use all available measurements.

To quantify how much of the available information is captured by a pattern recognition algorithm, the *collection efficiency* (sometimes also referred to as hit efficiency) is defined:

$$\varepsilon_{coll} = \frac{N_{\text{on track}}}{N_{\text{total}}}$$

$N_{\text{on track}}$ is the number of correct measurements that was used by the reconstruction software to form the track and N_{total} is the total number of measurements which the particle left in the detector(s).

2.6 Tracking definitions used in LHCb

While the last section focused on the idea of performance indicators, this section provides the precise definitions used in the LHCb experiment and throughout this thesis. Unless specified otherwise, these definitions follow those given in [15].

2.6.1 Matching Monte Carlo particles and tracks

Until now, the question which track goes with which particle has been neglected. In a Monte Carlo simulation, it is known which measurements a particle leaves in the detector and therefore it should be easy to tell if a track belongs to a particle or not. In reality, however, the pattern recognition algorithms will rarely yield tracks which consist only of measurements from a single particle, therefore some more involved matching is required. This is accomplished on the measurement level, by introducing a weight w with which a particle p_i contributes to a track t_j. This weight is simply

$$w(p_i, t_j) = \frac{N_{\text{measurements on track } t_j \text{ caused by particle } p_i}}{N_{\text{measurements on track } t_j}}$$

If such a weight $w(p_i, t_j)$ is at least 0.7, the track t_j is said to be *matched* or *associated* to the particle p_i. So, a track matches a particle if the particle contributes at least 70 % of the measurements on the track. Note that in

principle, several particles can hit the same detector channel(s) in a single event, thus, a measurement may be caused by more than one particle.

Depending on the type of track, this criterion is slightly modified:

- For tracks of types Velo and T, the description above is applied without changes.
- For Long tracks, both the Velo and the T part must match separately (and match the same particle). Note that there is no requirement for measurements in the TT stations.
- For Upstream and Downstream tracks, the Velo (or T, respectively) part must match, and there must be at least one measurement in one of the TT stations from the same particle that produced the track in the Velo (or T, respectively).

A track is said to be reconstructed correctly if it matches a particle according to the criteria given above.

2.6.2 Definition of efficiency denominator

For the efficiency definitions in Section 2.5 to make sense, it still needs to be clarified which particles are reconstructible, i.e. which ones should be reconstructed by the pattern recognition. For this purpose, the following set of definitions of reconstructibility has been chosen:

A particle is reconstructible in

- the Velo, if there are at least three r and three ϕ Velo sensors with clusters from that particle
- the TT, if there is at least one cluster in the first two TT planes (TT1) and one cluster in the last two TT planes (TT2)
- the T stations, if there is at least one x and one stereo cluster in each of the three T stations

A track is said to be reconstructible as

- Velo track, if it is reconstructible in the Velo
- T track, if it is reconstructible in the T stations
- Long track, if it is both reconstructible as Velo and T track
- Upstream track, if it is reconstructible in the Velo and in the TT

- Downstream track, if it is reconstructible in the T stations and in the TT

In addition to these standard definitions, we include a specialised definition for particles and tracks which are reconstructible in the Outer Tracker (OT) as these will be of special interest later on:

- A particle or track is reconstructible in the OT, if it has at least one x and one stereo cluster in each of the three OT stations.

The difference between T station and OT reconstructible particles or tracks is that the definition for the T stations does not distinguish between the measurements in the IT and in the OT.

2.6.3 Event-weighted versus track-weighted quantities

While the tracking quality indicators have been defined now, there are still two weighting procedures which are in common use:

- A quantity can be given as *track-averaged* quantity which basically means that each track or particle enters unweighted. The formulae above are given in that formulation.

- These quantities can also be given as *event-averaged* quantities. Here, one calculates a quantity per event and then averages over the per-event results for that quantity, e.g.:

$$\text{event-averaged ghost fraction} = \frac{1}{N_{\text{Events}}} \sum_{\text{Events}} \frac{N_{\text{Ghosts in current event}}}{N_{\text{Tracks in current event}}}$$

The reason to have two weighting methods is that for very hot events, many more tracks are found by the reconstruction than for events with average occupancy. These hot events will show increased ghost fractions and tend to show decreased reconstruction efficiencies due to larger combinatorics. They also contribute to the track-averaged quantities with increased weight because they have many more tracks than the average event. Event-averaged quantities are introduced to give equal weight to events with different occupancies.

Unless specified otherwise, event-averaged quantities will be quoted.

Chapter 3

Tracking performance in the LHCb experiment

3.1 Track quality monitoring tool

As part of this thesis, a tool for monitoring the quality of tracking algorithms on Monte Carlo simulated data was developed, partly because such a tool was still missing, but also to gain familiarity with the software and definitions. Starting from existing but not very flexible code, an ntuple writing algorithm was designed which allows to check not only reconstruction and collection efficiencies, ghost fractions and purities but also track parameters and properties of individual measurements. Also, Monte Carlo information of the event is stored in some detail: All Monte Carlo particles, their properties and information linking measurements and tracks to Monte Carlo truth are written. This makes it possible to look only at B decay products, or only at particles one does not reconstruct for some reason. This ntuple can be read using ROOT[16] macros, producing the plots the user has in mind. Macros were written to plot efficiencies, ghost fractions, collection efficiencies and purities versus p, p_T or occupancy, for example. Most of the plots in this thesis showing these quantities have been produced using those ntuple-based track quality analysis macros.

3.2 Tracking performance

In this section, the performance of the tracking code, especially that of the algorithms designed to find Long tracks, is presented. Starting from Velo and T tracks, the performance of Forward Tracking and Track Matching is studied. These studies have been carried out on a Monte Carlo sample

consisting of 15000 $J/\psi(\mu^+\mu^-)K_s$ events[1]. The sample was generated such that the average number of interactions per event corresponds to the value expected at a luminosity of $2 \cdot 10^{32}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$.

For a particle to go into the set of reconstructible particles defining the efficiency denominator, we demand:

- reconstructibility in the applicable subdetector(s):
  - Velo track reconstructibility for the Velo tracking
  - T track reconstructibility for T station seeding
  - Long track reconstructibility for Forward Tracking and Track Matching
- the momentum at the production vertex of the particle is > 1 GeV
- it is produced close to the primary vertex: $|z_{prod.}| < 15$ cm
- it is not an electron nor a positron

Electrons and positrons are excluded because many of them are secondaries produced by a photon converting into an electron-positron pair. Besides, they tend to be more difficult to reconstruct due to their increased energy loss compared to heavier particles (Bremsstrahlung is much more pronounced for such light particles).

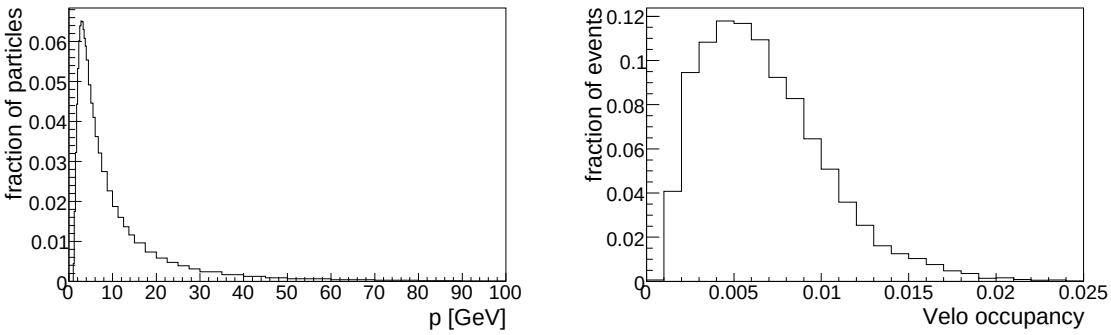
In the following subsections, plots of reconstruction efficiency and ghost fraction versus momentum and occupancy will be shown. The first choice has been made because one expects a momentum-dependence for ghost fraction and reconstruction efficiency: Due to the increased multiple scattering for low momentum tracks, the pattern recognition algorithms are more likely to not pick up all the correct measurements which leads to lower efficiency and, as a consequence, to a higher ghost fraction. The plots against occupancy have been made to provide a means of estimating the effect of noise, spillover or increased luminosity on the performance of the pattern recognition and tracking (which is likely to be worse in real data due to not simulated effects). To this end, the occupancy has been defined such that it includes noise and spillover measurements (where applicable): The occupancy is the fraction of channels per subdetector which registered a measurement (irrespective if it is noise, spillover or a real measurement).

[1] The sample used was produced in the DC'06 environment. The reconstruction software used was Brunel v30r14.

Before the performance of the different strategies is described, it is necessary to show momentum and occupancy distribution in the Monte Carlo sample used so that the plots shown later can be interpreted accordingly.

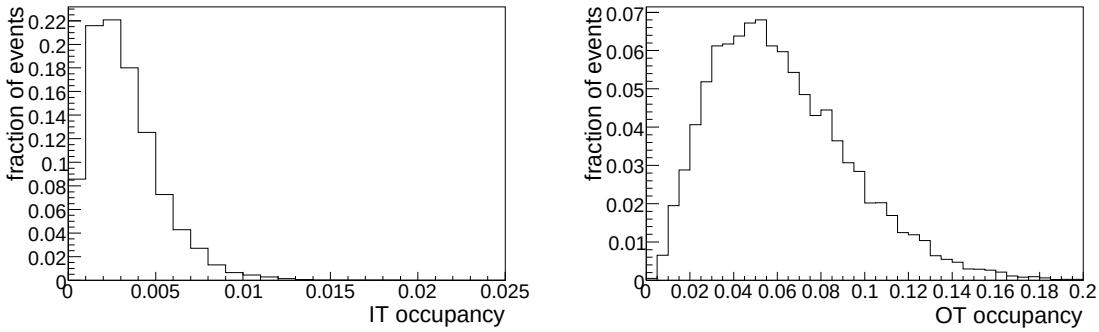
In Figure 3.1, the left plot shows a normalised momentum spectrum for particles reconstructible as Long tracks (the cuts described above have been applied, though), which shows a peak for momenta around 3 GeV and falls off rapidly for higher momenta. Note that the variable bin widths have been accounted for, so what is shown is the event-averaged particle density per unit momentum interval, normalised such that the integral of the histogram gives unity.

The right plot of Figure 3.1 gives the distribution of Velo occupancies. Figure 3.2 shows IT (left plot) and OT (right plot) occupancy distributions, one can see that for the silicon detectors with their high granularity, the occupancy is much lower than for the OT. Additionally, the OT is affected by spillover which gives rise to more measurements.



(a) **(b)**

Figure 3.1: *Normalised momentum spectrum for Long reconstructible particles (a), normalised Velo occupancy distribution (b).*



(a) **(b)**

Figure 3.2: *Normalised occupancy distribution for IT (a) and OT (b).*

3.2.1 Performance of Velo tracking

The reconstruction efficiency for tracks reconstructible in the Velo is $(90.4 \pm 0.1)\%$ with an event-averaged ghost fraction of $(4.7 \pm 0.1)\%$. The behaviour of the Velo reconstruction with momentum and Velo occupancy is shown in Figures 3.3 and 3.4. There is no plot of ghost fraction against momentum, because for ghosts, one would need to plot against reconstructed momentum (they can not be matched to a Monte Carlo particle by definition) which can not be determined due to the Velo being practically free of magnetic field.

As expected, low momentum particles are more difficult to reconstruct due to the larger effect of multiple scattering on their trajectory. The reconstruction efficiency is relatively flat for the occupancies present in the data sample, meaning that the Velo reconstruction code is relatively robust with respect to the measurement density. Of course, the ghost fraction increases appreciably with the number of measurements due to the additional combinatorics.

3.2.2 Performance of Forward Tracking

Forward Tracking is one of the strategies designed to find Long tracks. Its reconstruction efficiency in the DC'06 framework is $(87.0 \pm 0.2)\%$, the event-averaged ghost fraction is $(11.2 \pm 0.1)\%$. Figure 3.5 shows the momentum dependence of the reconstruction efficiency and ghost fraction. One can see that for lower momentum tracks (below 10 GeV), the efficiency drops while the plateau is at around 96%.

Figure 3.6 shows ghost fraction and reconstruction efficiency versus Velo occupancy (the occupancy in the other detectors correlate with the Velo occupancy, so only plots against Velo occupancy are shown). The efficiency

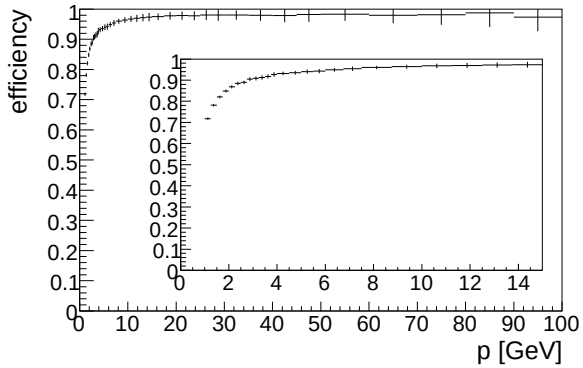
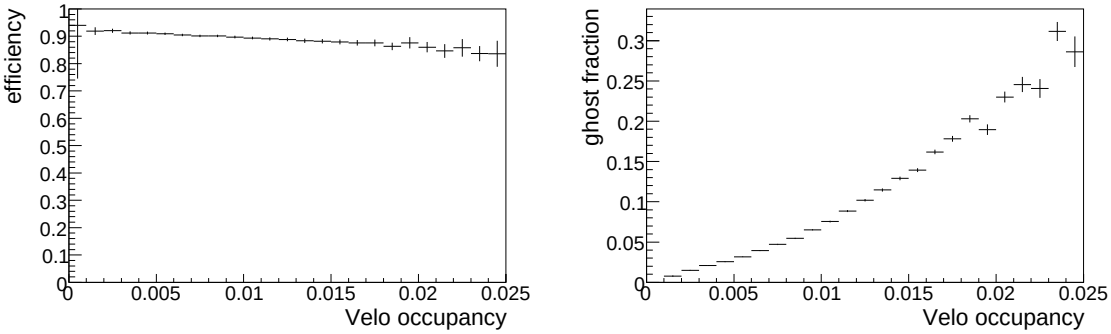
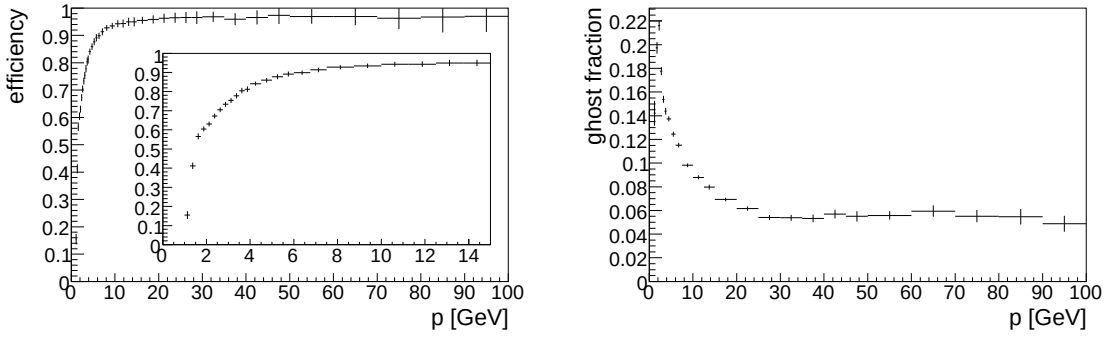


Figure 3.3: *Reconstruction efficiency in the Velo against momentum. The inlay plot shows the low momentum end in greater detail. Reconstructed particles with momenta below 1 GeV are not shown in this and subsequent plots.*



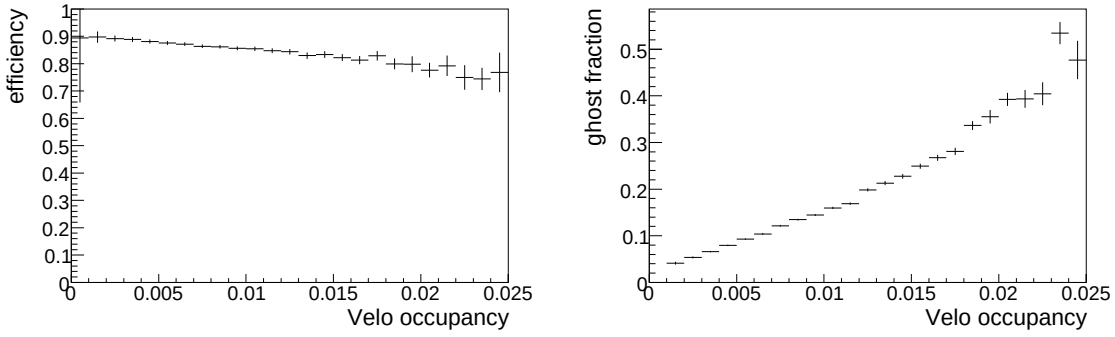
(a) **(b)**

Figure 3.4: *Reconstruction efficiency (a) and ghost fraction (b) in the Velo against Velo occupancy.*



(a) **(b)**

Figure 3.5: *Reconstruction efficiency (a) and ghost fraction (b) for Long tracks found by the Forward Tracking against momentum.*



(a) **(b)**

Figure 3.6: *Reconstruction efficiency (a) and ghost fraction (b) for Long tracks found by the Forward Tracking against Velo occupancy.*

degrades only a little for the higher occupancies while the ghost fraction will probably cause problems for high occupancies — 50% or even more is a sizeable fraction. Note that a large part of the ghost fraction seen at high occupancies seems to be inherited from the Velo ghost fraction. Also, the effects of higher measurement density on the ability to find the correct cluster in Hough space can be seen.

3.2.3 Performance of T station seeding

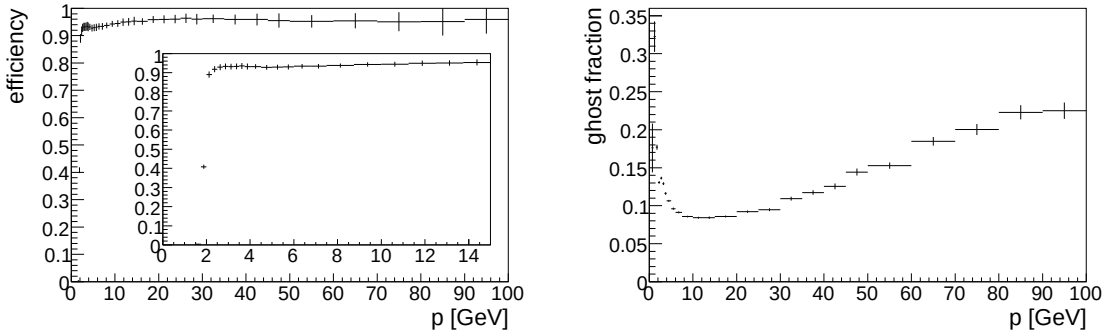
The efficiency for finding track seeds has been evaluated using the T station seeding algorithm described earlier. The event-averaged reconstruction efficiency is $(91.1 \pm 0.1)\%$ while the ghost fraction is at $(11.2 \pm 0.1)\%$.

For this algorithm, the plots versus occupancy have been done twice, for both Inner and Outer Tracker occupancies, to highlight possible differences between the two detector systems. In Figure 3.7, one observes the expected behaviour for the reconstruction efficiency (it plateaus for high momenta), while there are curious features in the ghost fraction plot: The ghost fraction has a minimum at around 10 GeV and rises quite quickly for momenta lower than this. This is expected because only few ghosts have low curvature which corresponds to high reconstructed momenta. The ghost fraction also rises (more slowly) for higher momenta which is somewhat contrary to the naive expectation. The reason for this is that the number of ghost tracks dies out more slowly with rising momenta than the number of reconstructed particles, see Figure 3.10a.

Looking at the plots in Figures 3.8 and 3.9, it can be seen that both efficiency and ghost fraction drop dramatically at high occupancies. This is due to the algorithm ignoring detector modules which show per-module occupancies above a certain threshold. The fact that efficiency drop looks different for OT and IT (apart from scaling the occupancy) can be attributed to spillover events which cause a certain level of "background occupancy" in the Outer Tracker which is not present in the Inner Tracker. This can be seen in Figure 3.10b which shows IT versus OT occupancy.

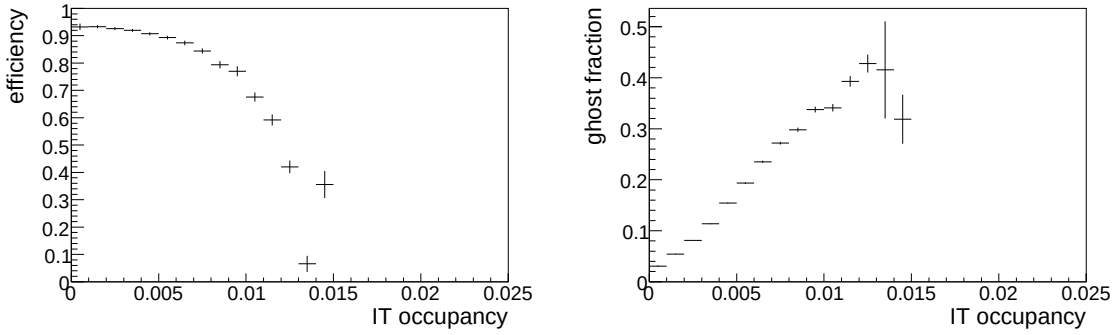
3.2.4 Performance of Track Matching

For Long tracks, Track Matching has an efficiency of $(79.2 \pm 0.2)\%$ with a ghost fraction of $(13.5 \pm 0.1)\%$. The efficiency is significantly worse than that of Forward Tracking. The ghost fraction matches roughly the 16% that one would expect by adding Velo and T seeding ghost fractions (about 5% and 11%, respectively). It is a little better because of the requirement to have matching tracks in the Velo and the T stations.



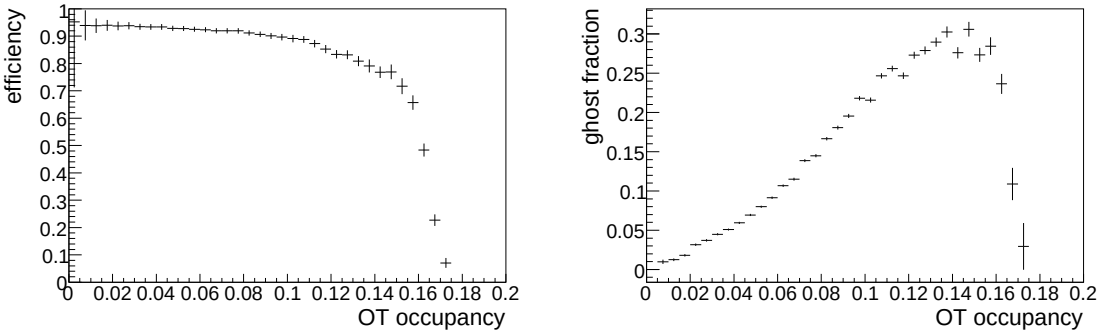
(a) (b)

Figure 3.7: *Reconstruction efficiency (a) and ghost fraction (b) for T tracks found by the T station seeding algorithm against momentum. The sharp rise in ghost fraction for low momenta can be explained by the fact that only few arbitrary combinations of measurements have low curvatures and do thus appear to have high momenta. The slower rise for high momenta is due to the number of reconstructed particles decreasing more rapidly with momentum than the number of ghosts (cf. Fig. 3.10a).*



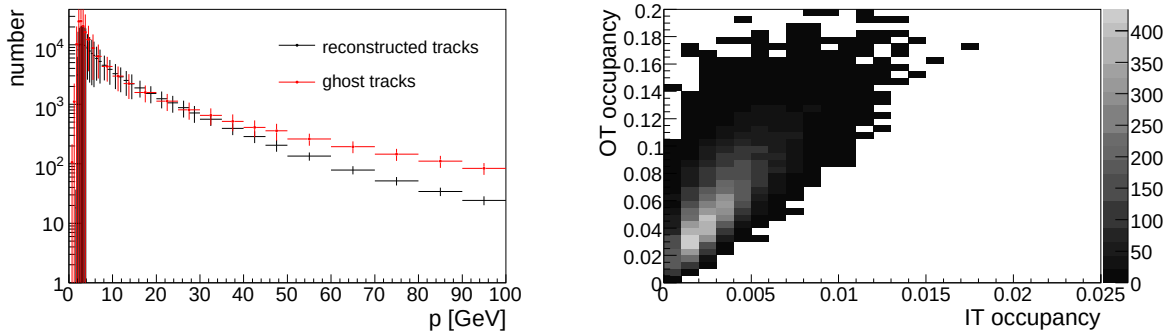
(a) (b)

Figure 3.8: *Reconstruction efficiency (a) and ghost fraction (b) for T tracks found by the T station seeding against IT occupancy. Both curves drop at high occupancy (cf. Fig. 3.2a) because of cuts applied in the algorithm (see text).*



(a) (b)

Figure 3.9: *Reconstruction efficiency (a) and ghost fraction (b) for T tracks found by the T station seeding against OT occupancy. Both curves drop at high occupancy because of cuts applied in the algorithm (see text).*



(a) *This figure shows the total number of reconstructed particles and ghosts per momentum bin. It can be seen that the number of ghosts diminishes more slowly with momentum than the number of particles. (Note the logarithmic scale.)*

(b) *Occupancy in the Inner Tracker versus occupancy in the Outer Tracker. The OT occupancy does not scale exactly with the IT occupancy due to spillover which gives rise to "background occupancy" in the OT.*

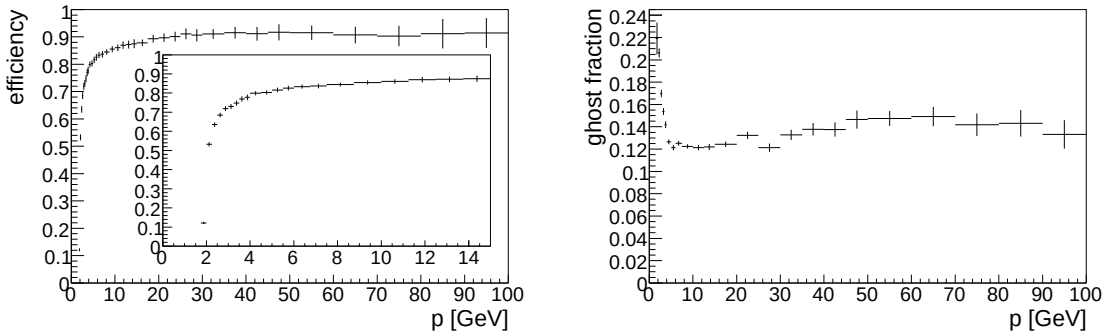
Figure 3.10:

The particles found by Forward Tracking and Track Matching are not identical: It can be seen in the next section that taking tracks from both algorithms gives better efficiency than either of the two can achieve alone.

Figures 3.11 and 3.12 show the behaviour of the algorithm as function of momentum and Velo occupancy. Apart from the fact that the efficiency is lower and the ghost fraction is a little higher than in the Forward Tracking case, one notes that the efficiency degrades more quickly for high Velo occupancies when compared with Forward Tracking (Figure 3.6).

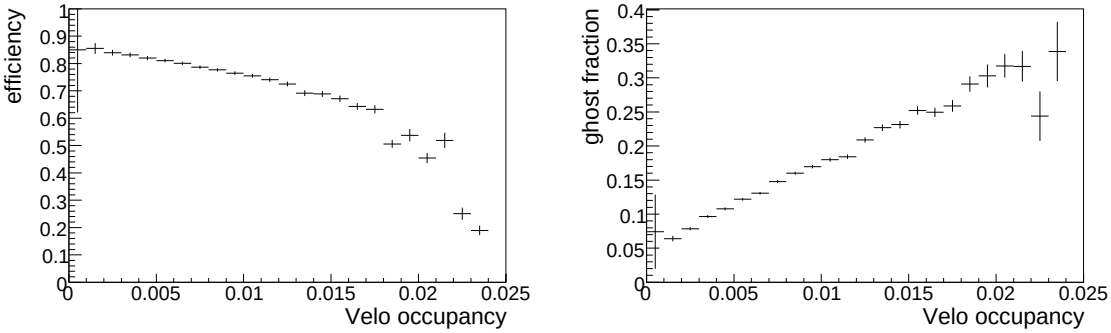
3.2.5 Overall performance for Long tracks

The overall performance for Long tracks after the clone killer has performed its task is characterised by a reconstruction efficiency of $(90.3 \pm 0.2)\%$ and a ghost fraction of $(16.7 \pm 0.1)\%$. The plots (see Figures 3.13 and 3.14) exhibit the expected properties, and it can be seen that one actually gains in efficiency by combining tracks from Forward Tracking and Track Matching. Interestingly, not only different good tracks are found by these two strategies, the ghost fraction is higher as well, suggesting that the ghost tracks found by the two strategies are different. This might be exploited to improve the discrimination between ghosts and good tracks: When a track has only been found by one of the two strategies, it is more likely to be a ghost than one that has been found by both. Of course, this can not be the only criterion on which such a decision must be based. Unless one is sure that the decision taken is correct in the vast majority of cases, it should probably be taken at the analysis level rather than at the reconstruction level, enabling each user to choose between high efficiency and a very pure track sample.



(a) **(b)**

Figure 3.11: *Reconstruction efficiency (a) and ghost fraction (b) for Long tracks found by the Matching strategy against momentum.*



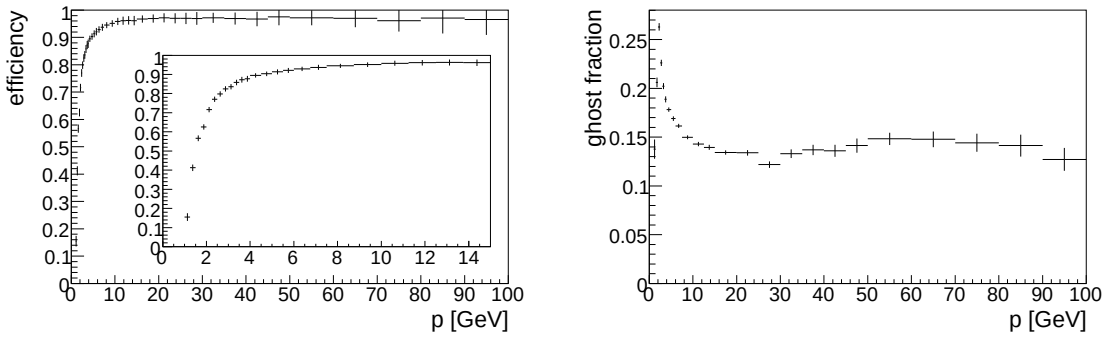
(a) (b)

Figure 3.12: *Reconstruction efficiency (a) and ghost fraction (b) for Long tracks found by the Matching strategy against Velo occupancy.*

Table 3.1 summarises the performance of the different algorithms.

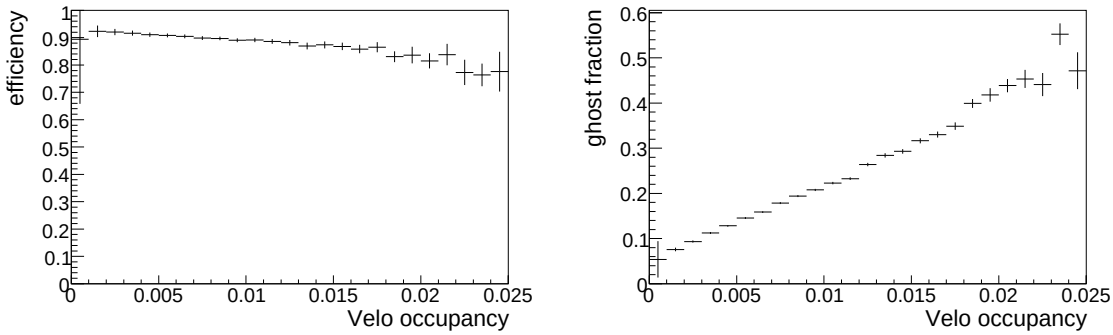
Algorithm	Efficiency [%]	Ghost fraction [%]
Velo	(90.4 ± 0.1)	(4.7 ± 0.1)
Forward	(87.0 ± 0.2)	(11.2 ± 0.1)
T Seeding	(91.1 ± 0.1)	(11.2 ± 0.1)
Matching	(79.1 ± 0.2)	(13.5 ± 0.1)
Overall performance for Long tracks	(90.3 ± 0.2)	(16.7 ± 0.1)

Table 3.1: *Summary of the performance of the algorithms investigated in this chapter.*



(a) **(b)**

Figure 3.13: *Reconstruction efficiency (a) and ghost fraction (b) for Long tracks against momentum.*



(a) **(b)**

Figure 3.14: *Reconstruction efficiency (a) and ghost fraction (b) for Long tracks against Velo occupancy.*

Chapter 4

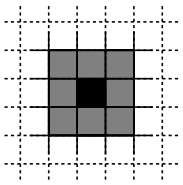
Cellular Automaton principles

In this chapter, an introduction to the concepts of cellular automata is given. Then, the use of cellular automata in pattern recognition and track finding algorithms is introduced with a simplified example. The next chapter will discuss the implementation of such an algorithm for track finding in the LHCb Outer Tracker.

4.1 Introduction to Cellular Automata

The concept of cellular automata has been around since the first days of computing (see e.g. [17]), and has been used for a variety of purposes. Conway's "Game of Life" (see [18]) is a simple example which became more widely known and acquired some popularity. Therefore, it will be used as an example to illustrate cellular automaton concepts.

Conway's "Game of Life" models some aspects of the behaviour of living cells: It consists of an infinitely extended rectangular two-dimensional grid of *cells*; each of them is either *alive* or *dead*. The *neighbourhood* of a cell are the eight cells surrounding it. Time evolves in discrete steps, and the *state* (i.e. dead or alive) of each cell in a time step depends only on the states of its neighbours in the previous time step:



Partial view of the grid of a cellular automaton. A cell (black) is shown together with its neighbourhood (grey).

- If a cell has less than two neighbours, it dies of "loneliness" (it can't survive alone).
- If there are more than three neighbours, the area is overpopulated and the cell dies of "starvation" (e.g. due to a limited supply of nutrients or similar constraints).
- In addition, if a dead cell has three living neighbours, that cell will

become alive in the next time step. (This is a simple model for replication.)

Figure 4.1 shows the evolution of a simple pattern according to those rules. One can see that the pattern "glides" across the grid of the cellular automaton.

In general, cellular automata consist of so-called *cells* (of which each has a *state*), a definition of the *neighbourhood* of a cell and a rule describing the evolution of a cell depending on its neighbourhood. Time proceeds in discrete time steps, and the new state of each cell depends only on the previous state of its neighbours. All cells change their state simultaneously.

The use of cellular automata for track reconstruction was pioneered by I. Kisel and used successfully in several experiments (see e.g. [19], [20], [21], [22]). The idea is to define a cell, called *tracklet* in the context of cellular automata used for track finding, as smallest track segment imaginable, namely a connection between two successive measurements. The neighbourhood of these tracklets is defined in such a way that only tracklets become neighbours which might be successive parts of a track. To do this, a physically motivated *local* track model is used: The decision if a tracklet is a neighbour of another tracklet is taken according to that model (e.g. a straight line in the absence of scattering and magnetic fields). This needs to be done only once for each tracklet, and only tracklets which are spatially close to the tracklet currently being considered can influence the decision — that is why the model is called local.

There is a big advantage to using a local track model in this way: Since there are not many tracklets spatially close to a given tracklet, and since the neighbourhood of a tracklet has to be determined only once, the resulting algorithm will touch each tracklet only once, avoiding the excessive buildup of combinatorics that many algorithms using global track models show. In these algorithms, the approach is to take all reasonable combinations of measurements for a given starting and ending measurement and then pick the best combination. Cellular automaton based algorithms use the local track model to "distill" the data in such a way that the approach "take all reasonable combinations and chose the best" usually boils down to "there is only one alternative to form a combination, so take it".

4.2 Cellular automaton used in tracking

In this section, an overview of the algorithm is presented; the emphasis will be on conveying the idea. For illustration purposes, a heavily simplified two-

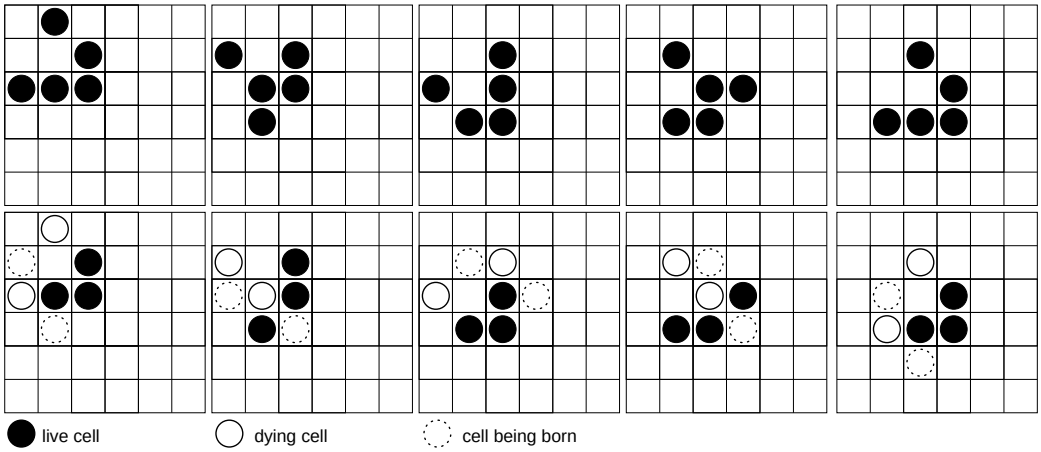


Figure 4.1: *The evolution of an initial pattern, a so-called "glider", in Conway's "Game of Life" is shown. The top row shows the evolution of the state of the cellular automaton with time increasing from left to right. The bottom row illustrates which cells die or will be born in the next time step — here, the rules can be seen at work. The pattern repeats after five time steps, but its position has changed — it "glides" across the grid.*

dimensional detector will be used. An implementation of such an algorithm for the LHCb Outer Tracker will be described in the next chapter.

For this overview, the simplified detector model consists of a detector which measures the x position of incident particles in layers perpendicular to the z direction, completely ignoring the additional degree of freedom in the y coordinate. Each layer is made of a line of sensors, one positioned after the other with no gaps in between. Each of these sensors will just measure if it has been hit by a particle or not. Figure 4.2 shows a sketch of such a simplified detector.

In principle, a pattern recognition algorithm using a cellular automaton proceeds through the following stages:

- Tracklet generation
- Neighbour finding
- Automaton evolution
- Forming and selecting of track candidates

These stages are discussed in the following sections.

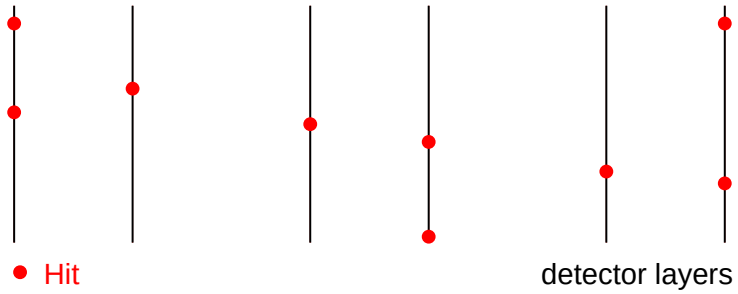


Figure 4.2: *Sketch of the measurements in the detector before the algorithm has started.*

4.2.1 Tracklet generation

During this stage, tracklets are generated between any two adjacent layers of the detector. Figure 4.2 shows a sketch of how the starting situation might look like.

To reduce combinatorics as early as possible in the algorithm, only those measurement combinations are used to form tracklets which are in fact possible. What "possible" means in a particular implementation can be quite different. One might have geometrical constraints which limit the slope of the line between the two measurements, or the requirement that the tracklet points to the primary vertex, for example.

For the tracklets to become part of a cellular automaton, the tracklets must also have a state. In our case, it is just a counter, initialised to zero, which will be used by subsequent stages of the algorithm.

Figure 4.3 illustrates how the result of this stage of the algorithm might look like.

To obtain a fast algorithm in the end, all knowledge about two measurement combinations produced by the same particle in adjacent layers should be used already at this stage to reduce combinatorics as much as possible. Forming only the tracklets which fit into the track model is a first step in that direction: For example, isolated noise measurements with nothing else nearby (like the one in the upper right corner of Figure 4.3) should not produce a tracklet, thus making such noise measurements completely invisible to subsequent stages. This leads to an algorithm which remains fast and efficient even in the presence of considerable noise.

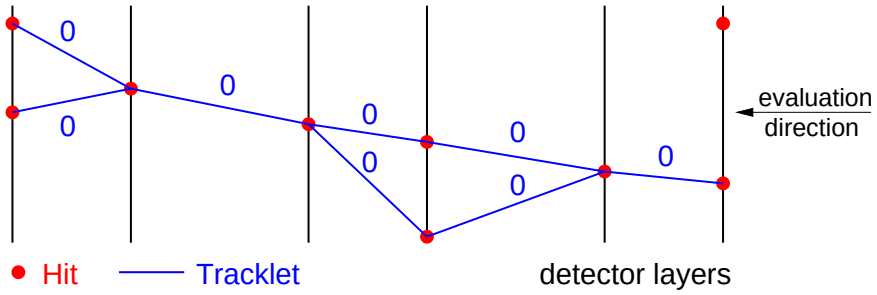
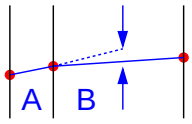


Figure 4.3: *Sketch of the detector at the end of the tracklet generation phase. Tracklets have been drawn between layers. The state of the tracklets is shown as well. Initially, all tracklets have their counter at zero. The order in which the algorithm processes the individual layers is indicated ("evaluation direction").*

4.2.2 Neighbour finding

While the previous stage provided tracklets, the basic working objects of the algorithm, there is no connection among them — in cellular automaton terms, the neighbourhood has not yet been defined. For the sake of this discussion, we will stick to one way to define the neighbourhood and provide a (necessarily non-exhaustive) list of alternatives for other implementations near the end.

The aim of this stage is to form a relationship between a combination of two tracklets which look like they might originate from a particle travelling through the detector, and not to form relationships among tracklets for which it is clear that no such connection exists. Again, the reason behind this is to reduce combinatorics as early as possible in the algorithm. Tracklet B is called a *neighbour* of tracklet A, if



Tracklet B is a neighbour of A.

- the starting measurement of tracklet A is the one in which tracklet B ends
- the kink angle (the angle at which they intersect) is small enough

The Figure in the margin depicts the situation. The first requirement simply comes from the assumption that each particle intersects with a detector layer at a single position. The second requirement is given by the local track model: In our case, locally, a track should look like a straight line, while we allow for small deviations (non-vanishing kink angles) which in the real experiment might come from the deflection in a magnetic field or multiple scattering.

Just like the algorithm works in evaluation direction (in the pictures from right to left), the neighbour relation is directed as well: It permits going from left to right, i.e. in the opposite direction. This is a convention, but one that will be useful.

The reason for this is that it is desirable to find tracks which go through the whole detector first. If particles can leave the detector early, the number of correct tracklets will be lower at one end of the detector. Starting with finding neighbours from this side will ensure that dead ends are encountered early in the process of forming track candidates (which starts from the opposite side, following the direction of the neighbour relation). Depending on the problem at hand, the decision at which end of the detector to start may affect timing properties, efficiency and ghost fraction of the algorithm — best is probably to do the experiment and the use the alternative which works better.

Other possibilities to define the neighbour relation might include

- a χ^2 distance between the parameters of a local track fit for each of the two tracklets; this has been done in [23], for example
- search windows around the projected position of the tracklet into the next layer(s)
- additional cuts, e.g. if the local curvature (obtained from the three measurements) is consistent with a momentum estimate obtained through other means

This list is clearly not exhaustive, and for each implementation, an implementor needs to look for information which can be used at this stage of the algorithm to keep the combinatorial burden as low as possible.

4.2.3 Automaton evolution

Now that the neighbourhood of a tracklet is defined, the rules of the evolution of the cellular automaton must be defined. What one would like to have in the end is a collection of chains of tracklets with each tracklet chain being a candidate for a track. Long chains should be favoured over short ones because a track going through the whole detector is more likely to come from a physical particle than a short one for which no continuation can be found. In principle, the chains themselves are already formed: The neighbourhood relation holds the tracklets together, so candidates can be found by starting with a tracklet and continuing with one of its neighbours. It is the task of the evolution of the cellular automaton to dig out the long chains before the shorter ones.

This is done using the counters which have been initialised to zero in the tracklet generation phase. During each step of the evolution of the automaton, the counter of each tracklet is incremented if one of its neighbours has a counter which is at least as high as its own present counter. For example, a tracklet might have at some step in the evolution a counter of 5 and neighbours with counters 2, 5, 4 and 3. Then, the new counter of the tracklet will be 6 for the next step. The evolution stops when all counters have stabilised. Figure 4.4 gives an impression how things may look like afterwards.

After the evolution phase, the tracklets with the highest counter are those which give the longest tracklet chains. But the algorithm gains more than that: By following the neighbour relation from a tracklet with the highest counter, branching to neighbours which lead to short chains can be avoided. For such branches, the sequence of counters along the chain suddenly "jumps" by more than one. This will become clearer when looking again at Figure 4.4. There is a "side path" consisting of two tracklets, both of which have a counter of zero. This means that both tracklets have no neighbours. Even if the left tracklet of the side path was a neighbour of the tracklet to its left (the one with counter 3 in the chain of successive counters), the side path would not be followed because the change in counter when going from one tracklet to the next does not equal -1.

One must mention here that there is an issue with the algorithm presented so far: If a measurement is missing in a layer, either due to detector inefficiency or because the particle passed through a dead region of the the detector, no tracklets can be generated. A way needs to be found to jump

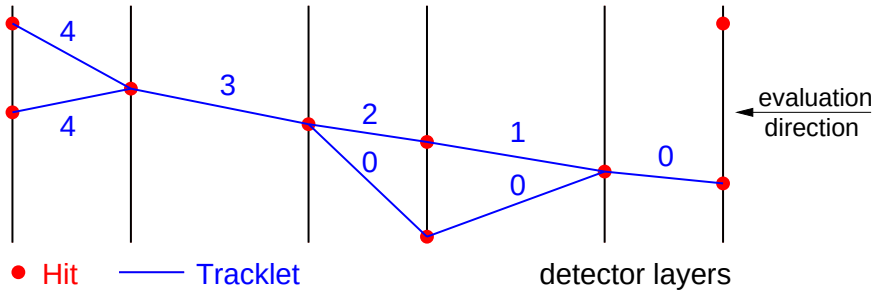


Figure 4.4: *Sketch of the state of the automaton after the evolution phase: tracklet counters are shown next to each tracklet; the neighbourhood relation itself is not shown because it would needlessly obscure the picture. The two tracklets which have a counter of zero do not have neighbours due to the large kink angles, therefore, their counter is never incremented.*

over such "holes" in a chain of tracklets. If the detector is reasonably efficient, this will not occur often, and when it does occur, shortened chains can still be found which the algorithm can merge afterwards into longer candidates.

4.2.4 Forming and selecting candidates

In principle, the essence of forming candidates has already been described in the last section. Sorting all tracklets by decreasing counter, candidates can be "read off":

Starting with a tracklet with the highest counter, the algorithm follows the neighbourhood relation while obeying the "counter must decrease by one"-constraint along the chain. It branches (i.e. makes several candidates out of a single one) where there is more than one neighbour which satisfies these criteria. Figure 4.5 shows the situation for our example: There are two candidates, each starts at one of the tracklets with counter 4, and they differ only in their first tracklet.

First, all candidates are formed which start with tracklets with highest counter value. Then, in a procedure similar to the one described in Section 2.4.4, an attempt is made to select the good candidates and reject the ghosts. In principle, this selection process uses standard pattern recognition tools and is not strictly part of the cellular automaton. Tracks are sorted by some quality measure and selected in order of decreasing quality. Their measurements are marked used. If they have a used measurement fraction greater than some cut, they are rejected. Also, tracklet reuse is not permitted because it is extremely unlikely that two different particles will travel through identical sensors in two different layers. Obviously, this last choice may have to be reconsidered in scenarios where the track density is very high.

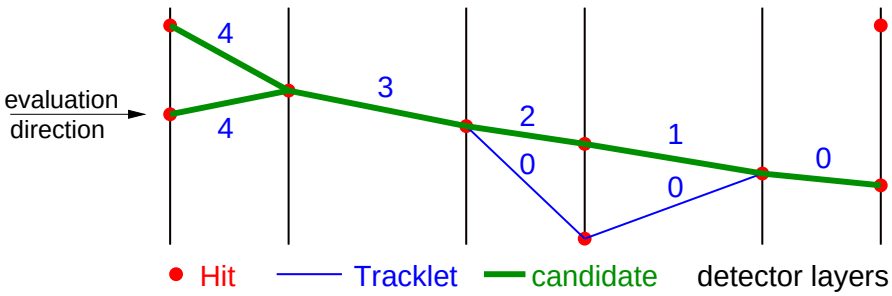
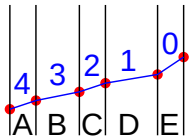


Figure 4.5: *Sketch of the state of the automaton after candidates have been formed: Two candidates can be seen which only differ in one tracklet (thick green lines).*

After dealing with the longest candidates, the algorithm continues with shorter ones. Time is saved by stopping at tracklets already used instead of following them. Otherwise, the algorithm would find shortened versions of the tracks already selected. For example, consider a selected track like the one starting with tracklet A in the figure in the margin. If the reuse of tracklets was permitted, the candidate consisting of tracklets B, C, D and E would be found when looking for four-tracklet candidates, duplicating work already done.



Reusing tracklets leads to shortened versions of the chain starting at tracklet A.

This procedure for following tracklets and selecting candidates continues until the candidates formed are too short to be plausible. It is up to the implementor to decide when this limit is reached. Once all candidates have been found, it is possible to "polish" the candidates: One can try to merge short candidates which are likely to belong to the same particle into a single longer candidate. It is also possible to add unused measurements to nearby tracks if they have not been picked up by the cellular automaton for some reason.

Chapter 5

Cellular automaton based seeding for the LHCb Outer Tracker

In this chapter, an implementation of a seeding algorithm based on the cellular automaton approach introduced in the last chapter is described. For clarity, an overview is given, highlighting additions to and changes from the simplified description given in the previous chapter. Then, the different stages of the algorithm will be discussed in more detail.

5.1 Overview

There are two important details which are relevant for the Outer Tracker but have been omitted for clarity in the first description of the algorithm. One is the third degree of freedom in the coordinates: The track parameters y and t_y must be inferred using the measurements in the stereo layers. The other detail is the inner structure of the Outer Tracker with each layer consisting of two monolayers of straw tubes, measuring drift times.

The first item is obviously dealt with by using the stereo layers in the process: In this specific implementation, there are two cellular automata running in parallel, one for each of the two projections. The one working in yz projection is tightly coupled to the evolution of the automaton working in xz direction, enabling the use of the former to confirm the choices made by the latter.

The second item is accounted for by forming clusters from neighbouring measurements in different monolayers of the same module.

Thus, the algorithm consists of the following stages:

- **Data preparation:** In this stage, measurements are retrieved, and clusters are formed. They are sorted by increasing coordinate value along the measurement direction.
- **Tracklet generation:** Tracklets are formed between all adjacent x layers.
- **Stereo enhancement:** Clusters from both u and v layers are combined to form a pseudo-x clusters which are then matched to the existing tracklets. This can be considered as the tracklet generation stage of the cellular automaton working in yz projection.
- **Neighbour finding:** The procedure is very similar to what was described in the last chapter, the main difference is that the neighbour relations for the cellular automaton working in yz projection are worked out as well. The evolution of both cellular automata can be done efficiently at the same time.
- **Forming and selecting candidates:** During this stage, track candidates are formed. For each candidate in xz projection, all possible candidates in yz projection are formed in parallel. A candidate selection phase follows which brings down ghost and clone fraction to acceptable levels.
- **Track conversion and cleanup:** The remaining candidates are converted to standard `LHCb::Track` objects which can be used by the rest of the reconstruction software. There are also some cleanup tasks which the algorithm needs to perform in this phase, for example memory management. These cleanup tasks are only mentioned here for completeness and to indicate that they exist, but they will not be discussed further.

These stages are performed once for the lower half and once for the upper half of the Outer Tracker with no coupling between the two. This is possible because the number of particles going from the lower half into the upper one or vice-versa is negligible; the magnet does not cause a noticeable deflection in the y direction. As has been mentioned in the last chapter, the first four stages work towards the primary vertex, starting with the layer with the largest z position in T3. Candidates are formed in the opposite direction, following tracklets with large counters towards the calorimeters and RICH2.

The chapter starts by stating how performance of the single steps of the algorithm will be evaluated, then the different stages are described in more detail, giving figures for the performance of the different stages.

The next chapter discusses the overall performance of the algorithm.

5.2 Evaluating performance

Performance of the algorithm is measured using the same sample of events that was used in Section 3.2. More specifically, a particle is considered to be reconstructible if it satisfies the following criteria:

- It is reconstructible in the OT, i.e. it must have in each of the three OT stations at least one x and one stereo measurement.
- Its momentum at the production vertex is greater than 1 GeV.
- It comes from the primary vertex region, i.e. the z coordinate of its production vertex must satisfy $|z| < 15$ cm.
- It is neither electron nor positron.

This is very similar to what is used in Section 3.2. In fact, only the requirement on the number of hits in the T stations has been adapted to suit an algorithm designed for the OT only.

To evaluate the performance of individual stages of the algorithm, there are additional criteria which must be met for a particle to be reconstructible in a given stage. For example, a particle will only allow to form a tracklet between two given x layers if it has measurements in both of these layers. The exact definitions are mentioned when the performance of the stage in question is discussed.

For simplicity in the evaluation of single stages of the algorithm, this thesis will focus on particles which leave measurements in all six x layers of the Outer Tracker. The corresponding tracks should be found as chains of five tracklets. Therefore, reconstruction of shorter chains will be skipped for the performance evaluation of individual stages. For the evaluation of the full algorithm in Chapter 6, tracklet chains of lengths five and four are used. There, figures are presented for both alternatives, reconstructibility in the OT and reconstructibility in the OT plus the additional requirement of measurements in all six x layers.

The reason to evaluate the algorithm on this particular subset is that it currently has no way to cope with interrupted tracklet chains: Whenever a particle leaves no measurement in one of the inner four x layers, the algorithm can not find it. As only about 8.8 % of all particles reconstructible in the OT (in the sense defined above) are affected by this and time was running out, it was decided that the remaining time for finishing this thesis would be better spent implementing code to use the stereo layers to bring down the

ghost fraction[1]. With some more work, the algorithm can be improved to handle interrupted tracklet chains.

While the evaluation of the overall performance can be done using the same Monte Carlo sample and the same tools as in Section 3.2, closer investigation of the individual stages needs more specialised code, associating the result of individual stages to Monte Carlo truth. This makes the code very slow; therefore, only a subsample of 2000 events will be used for this purpose. However, care has been taken to evaluate the performance of individual stages on a different subsample than the one used to derive cuts to avoid any bias.

5.3 Data preparation

In this section, the data preparation process is described. There is no major difference between x and stereo layers during this stage of the processing. The preparation phase consists of the following steps:

5.3.1 Selection of measurements

The first step is to retrieve an array with Outer Tracker measurements. If their drift radii are not within acceptable limits, the measurements are ignored. This is done because spillover tends to produce measurements corresponding to drift radii which are outside of the physical dimensions of the straws (see Section 1.4.1).

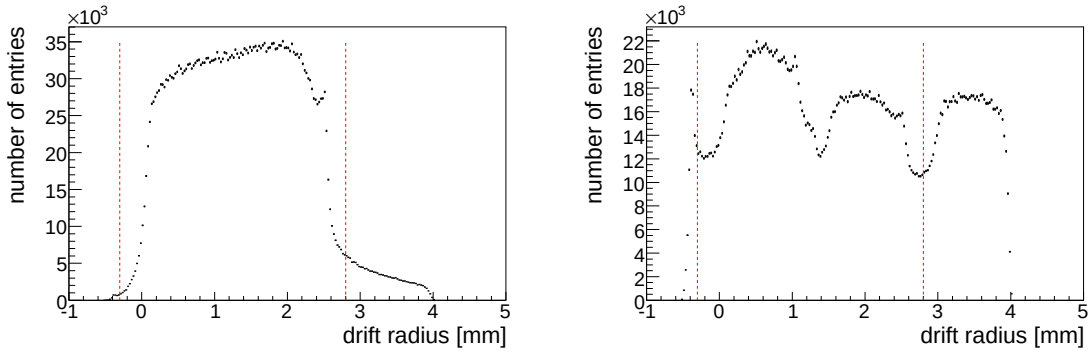
Figure 5.1 shows the distribution of drift radii for good measurements and for anything but good measurements (i.e. noise, spillover and crosstalk). One can see that a drastic reduction in the number of measurements can be achieved. It has been decided to cut so that only measurements satisfying $-0.3\ \text{mm} < r < 2.8\ \text{mm}$ survive.

5.3.2 Conversion to working objects

From the remaining measurements, working objects (of type `CASeedHit`) are created. Their function is to facilitate bookkeeping in the algorithm and to hide the complex detector geometry. This is achieved by using axes in the measurement directions x, u and v[2]. Sine and cosine of the stereo angles of

[1] Back then, the algorithm did not use the stereo layers at all, and ghost fractions of 50 % and more had to be tolerated if one wanted to keep efficiency on the order of 90 %.

[2] The term *measurement direction* will be used to denote the direction of the x, u or v axis.



(a) (b)

Figure 5.1: *Distribution of drift radii as returned by the reconstruction software at this stage of the processing; good measurements are in (a), (b) shows everything else. The structure in the right plot is a consequence of the spillover contribution from other bunch crossings. Dashed red lines will from now on indicate where cuts are performed.*

the individual layers and starting and ending points of the wires in question are also provided. Figure 5.2 sketches the axes used.

5.3.3 Sorting of the measurements

The working objects created during the last stage are sorted by increasing position in measurement direction. Sorting them makes searching by position very fast; binary searches[3] are used.

5.3.4 Forming clusters

Except for very low momentum particles which curl in the detector, one expects a particle to produce two measurements per layer in neighbouring straws from different monolayers. An example of such a situation is shown in Figure 5.3. There may of course be fewer measurements due to the particle passing through the insensitive area between two straws or between two modules, or in case of detector inefficiency.

Using pairs of measurements like the one depicted in Figure 5.3, clusters are formed starting from one end of each layer, moving in measurement direction towards the other end. The cluster position is just the arithmetic mean

[3]Binary searches are based on the same idea as finding a zero of a function by bisection. Arrays of length N can be searched using $\mathcal{O}(\log N)$ comparisons with this technique.

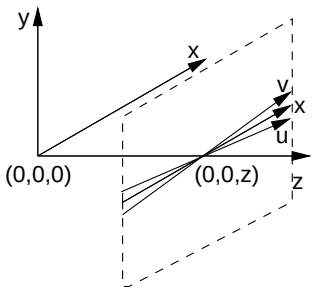


Figure 5.2: *Sketch depicting axes in the three possible measurement directions x, u and v, all attached to a single point on the z axis.*

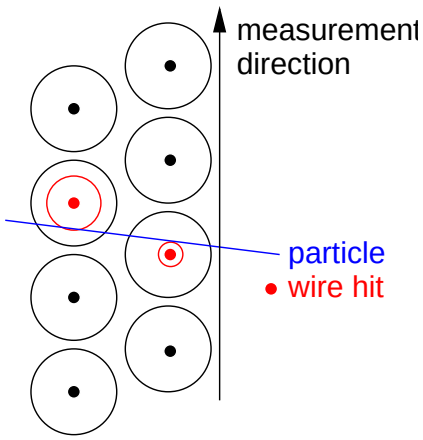


Figure 5.3: *Sketch of a typical cluster formed by a particle travelling through an OT module (only the straws themselves are drawn). Drift circles have been added for the straws giving a measurement.*

of the two wire positions. Clusters may overlap, the second measurement used to form a cluster may be used to start a new cluster if it has a neighbouring measurement like the first one in Figure 5.3. Pairs of measurements which were used to form clusters are ignored by the rest of the algorithm, the generated clusters take their place.

The reason for forming clusters is to group possibly related measurements — it will be shown in the next section that these clusters do not only reduce the number of possible combinations the algorithm has to cope with, they can actively be used to rule out that two clusters have their origin in the same particle.

For the rest of this thesis, the term *cluster* will refer to both a cluster in the sense of a combination of two measurements and a single measurement. For most aspects of the algorithm it does not matter if the algorithm uses a real cluster or a single measurement. In the few places where it does matter, the distinction between a single measurement and a cluster will of course be made, so it will become clear when the difference is important.

5.4 Tracklet generation

This section will describe the tracklet generation stage which forms tracklets from clusters in two adjacent[4] x layers of the detector.

For simplicity and clarity, it is convenient to introduce three terms which will be used in the following discussion. These terms are applied to the tracklets that are or could be generated between two given x layers.

- A *good tracklet* between two given x layers is a tracklet which has at both ends measurements from the same particle. There can also be measurements from other particles, noise and spillover which can contribute.

- Among all good tracklets for a particle between two given x layers, there is a *best tracklet* which is the one containing most measurements from that particle.

- A *bad tracklet* between two given x-layers is not a good tracklet. The terms *wrong tracklet* and *wrong combination* are also used interchangeably.

[4]Here, "adjacent" means that tacklet generation happens only between x layers which have no further x layers between them.

As the main interest lies in particles satisfying the criteria given in 5.2, the terms good tracklet and best tracklet will imply in the following text that the particles in question do indeed satisfy these criteria. Thus, plots and figures for best or good tracklets imply that the particles which produced the tracklets have more than 1 GeV momentum at their production vertex, are neither electrons nor positrons, etc. This is done to keep the wording simple.

In principle, there are three kinds of cuts applied to select the pairs of clusters which will be used to form tracklets. These are discussed in the subsections below. The order in which the cuts are discussed corresponds to the order in which they are applied, with fast and effective cuts first, more sophisticated ones requiring more computation time later.

The efficiency of this stage of the algorithm will be defined and discussed at the end of this section.

5.4.1 Geometrical cuts

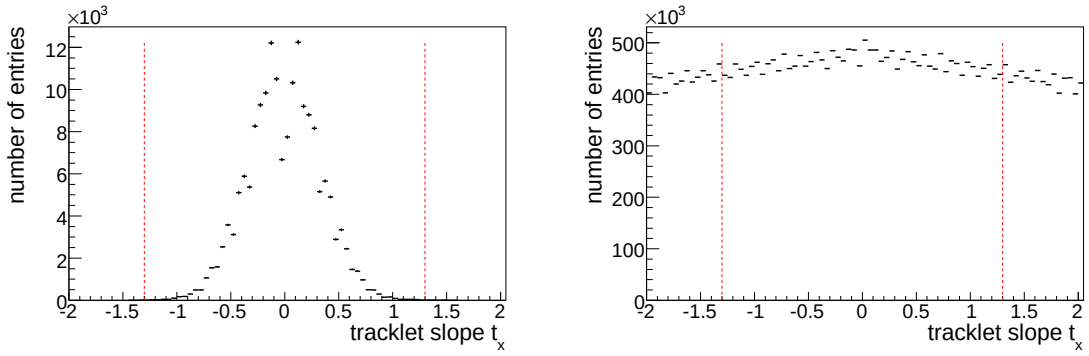
This subsection describes cuts for tracklet generation which involve geometric aspects. They are described below:

Slope cut

Particles coming from the primary vertex region and traversing all stations of the Outer Tracker give rise to a tracklet slope distribution like the one shown in Figure 5.4. Therefore, tracklets are generated only if their slope would fall into the range $|t_x| < 1.3$. On the sample used to produce these plots, the loss introduced by this cut is less than 0.01 %.

The structure which makes the plots look like being composed of two curves (most apparent in the right plot) is a binning effect: The slope is given by $t_x = dx/dz$ where dx and dz are the differences in wire or cluster positions in x and z direction. dx can only take a fixed set of values because the straw pitch remains constant, and dz is also quantised because the module-module distances in z do only take a finite number of values. Thus, t_x itself will only take discrete values, which are not equidistant in t_x. Filling such a distribution into a histogram will show binning effects.

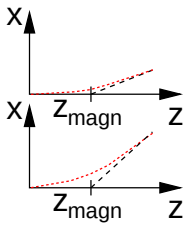
Due to the possibility to form wrong combinations with almost arbitrary slope within the physical dimensions of the detector, there is not much more to learn from the plot for wrong combinations, except for the large number of such possibilities.



(a) (b)

Figure 5.4: *Distribution of tracklet slopes for best tracklets (a) and wrong combinations (b). The dashed lines illustrate the cuts applied; this form of illustration will be used throughout the rest of the chapter. The small "jumps" in the two curves which are most apparent in the right plot are due to binning effects, see text.*

Correlation between slope and position

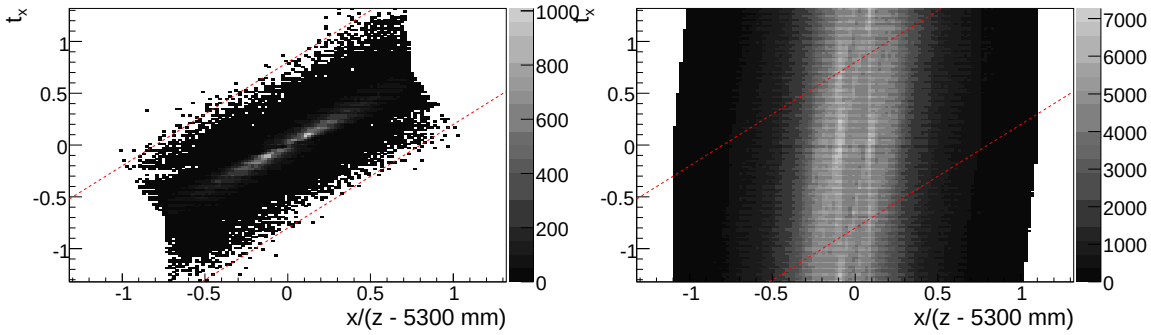


For particles from the primary vertex which traverse the magnet, there is a correlation between their x position in the T station and their slope t_x (the figure in the margin sketches two such cases, z_{magn} is the bending plane of the magnet at $z = 5300$mm).

Figure 5.5 is a scatter plot showing the slope of the line connecting the first measurement of a tracklet with the centre of the magnet at z_{magn} versus its slope t_x. The left plot is for best tracklets, the right one for wrong combinations. For best tracklets, one observes a diagonal band which is filled with measurements while the edges remain empty. In the right plot for wrong combinations, the edges are filled, indicating that only certain combinations of x position and slope t_x are produced by particles of interest. Note that a similar correlation has also been observed in [10].

This algorithm exploits this fact by requiring that $|x/(z - 5300\text{ mm}) - t_x| < 0.8$ where x and z are the coordinates of the first cluster of the tracklet. Less than 0.05 % of all best tracklets are lost by this cut.

Figure 5.6 sketches why the slope predictions from the x positions are not very accurate: For outbending particles, a line from the centre of the magnet through one of the measurements gives a reasonable idea of the slope of the particle trajectory. Unfortunately, this estimate is not too good for particles with very high momenta which do not bend in the magnet, and it becomes even worse for particles with trajectories bending in the other direction. Therefore, the window has to remain quite broad. It was tried



(a) **(b)**

Figure 5.5: *Scatter plot showing the slope of the line connecting the first measurement of a tracklet with the centre of the magnet versus its slope. Best tracklets are in (a), (b) contains only wrong combinations. One observes that best tracklets only occupy a small portion of the area shown in the plot.*

to improve this by using the curvature estimate obtained from the p_T-kick method (see Section 5.4.2), but there was little improvement. Therefore, it was decided to skip the additional computation and live with the simple version described above.

Custom search windows for trigger applications

It is possible to restrict tracklet generation to a narrow range in (x, t_x) parameter space. This range may depend on the z position. One possible application of such a cut is the confirmation of hardware trigger candidates in the software trigger: Usually, an estimate of the expected x position and slope t_x of a track can be obtained from the position of the energy deposits in the calorimeters or muon stations. To check if there is a corresponding track in the T stations, it is not necessary to reconstruct the whole detector. A narrow window around the expected track trajectory is sufficient to check for the presence of a track. This reduces combinatorics drastically and helps to meet the stringent constraints on algorithm execution time in the software trigger.

This has not yet been tested, but since the execution time for the offline case is moderate and the code is not yet optimised for speed, the application of the algorithm in the software trigger framework seems to be possible. However, some additional work would still be needed to restrict the data preparation step and the stereo enhancement to the range of interest to make

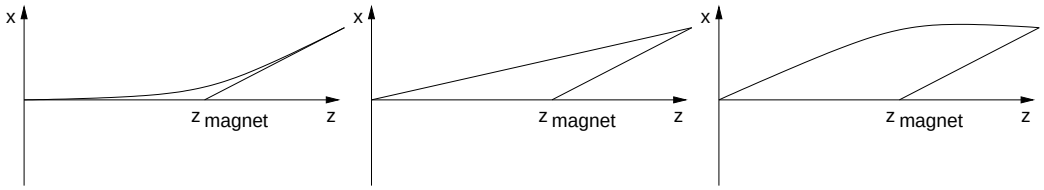


Figure 5.6: *Sketch illustrating why the correlation between x position and slope for tracks from the primary vertex region does not allow for tighter cuts: For outbending tracks with not very high momenta (left), the slope of the line joining the centre of the magnet with a measurement on the track is expected to agree quite well with the observed track slope in the T stations. If the track momentum is very high (middle), the agreement is not so good anymore, and it becomes even worse if the track curvature has opposite sign (right).*

the resulting algorithm fast enough for the trigger.

In the offline version of the algorithm, no cut is applied.

5.4.2 Momentum estimation and optional cut

For this cut, the momentum of the tracklet is estimated using the p_T-kick method described earlier. Since the coordinates and slopes of tracklets in yz projection are unknown at this stage, $y = t_y = 0$ is assumed. This will degrade the momentum resolution of the p_T-kick method because the field integral which is calculated will be slightly wrong. On the other hand, having an estimate of the tracklet momentum is certainly better than having none. To speed up the process, the result of applying the p_T-kick method to a pair (x, t_x) of tracklet coordinates and slopes is saved in a lookup table during initialisation. The binning is such that the x coordinate is sampled using bins 2.5 cm wide, and the slope is sampled with bins which are 0.005 wide.

In some applications, it could be convenient to cut away low momentum tracklets, thereby speeding up the algorithm. An example for such an application might be in the software trigger.

Figure 5.7 shows the momentum resolution obtained using this method for best tracklets.

The core has been fitted with a single Gaussian; its width corresponds to a momentum resolution of $\Delta p/p = \frac{p-p_{rec}}{p} \approx 5.3\ \%$. Here, p is the true momentum of the particle at its production vertex, and p_{rec} is the momentum reconstructed using the p_T-kick method as described above.

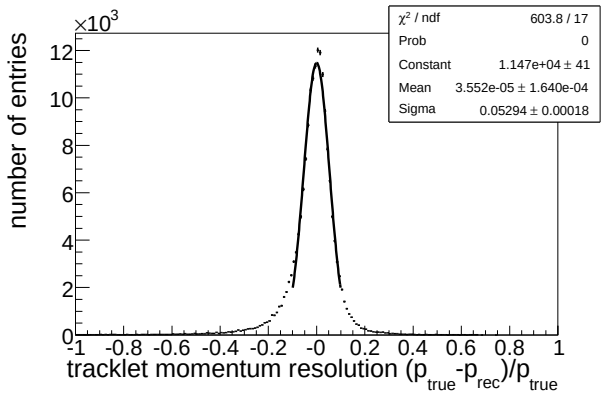


Figure 5.7: *Momentum resolution $\Delta p/p$ obtained during the tracklet generation stage using the p_T-kick method for best tracklets, assuming the tracklet is in the $y = 0$ plane. The fit to the core of the distribution was done with a single Gaussian.*

In the standard configuration, no cut is performed. The momentum estimate is used later in the algorithm.

5.4.3 Clustering continued: pitch residuals

Let us for the moment assume that all particles travelling through the detector were incident perpendicular to the detector layers (or, equivalently, parallel to the z axis, for this discussion we work only in xz projection). Then a two-measurement cluster would look like the one shown in Figure 5.8. The two drift radii will be called r_1 and r_2, and the amount of staggering between the two monolayers in the module, from now on called *pitch*, will be denoted by d. Then the following relation must hold (within the limits dictated by the detector resolution):

$$r_1 + r_2 = d$$

This means that the algorithm can tell if the two measurements forming the cluster do actually belong together.

The fact that most particles traverse the detector at an angle can be corrected for easily by calculating the effective pitch d_{eff} which will be seen by a particle travelling through the detector layer with a slope t_x[5]. The result is obtained without difficulty by projecting the vector joining the two wire

[5]In fact, the only reason why this method is not used earlier in the algorithm is that an estimate of the slope is needed to calculate the effective pitch.

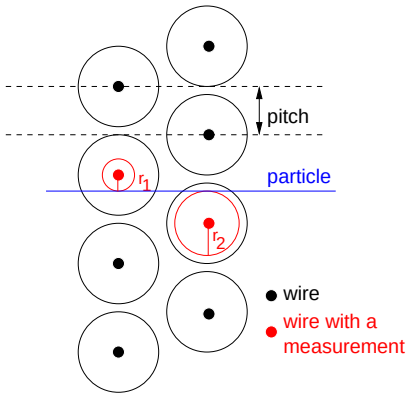


Figure 5.8: *The figure shows a cluster produced for a particle incident perpendicular to the detector layer. The amount of staggering between the two monolayers is called the* pitch.

positions of the straws involved, denoted $\vec{a}$, onto a plane perpendicular to the trajectory of the particle (cf. Figure 5.9):

$$d_{eff}(t_x) = |\vec{a} - (\vec{a} \cdot \vec{s}(t_x)) \cdot \vec{s}(t_x)|$$

Here, $\vec{s}(t_x)$ is the unit vector in the direction of the trajectory of the particle:

$$\vec{s}(t_x) = \frac{1}{\sqrt{1+t_x^2}} \begin{pmatrix} 1 \\ t_x \end{pmatrix}$$

Using the effective pitch d_{eff}, one distinguishes two cases: The particle either passes between the two wires , or it does not (see Figure 5.10). In the first case, the following relation must hold:

$$r_1 + r_2 = d_{eff}(t_x)$$

The second case is characterised by this equation:

$$|r_1 - r_2| = d_{eff}(t_x)$$

So, by checking if one of the two relations is is fulfilled (within the limits dictated by detector resolution), it can be decided if the slope t_x is consistent with the two measurements originating from the same particle. This is a powerful tool during tracklet generation because it can be used to veto wrong cluster combinations, both for recognising clusters which do not originate from a single particle, and for vetoing two (possibly perfectly) good

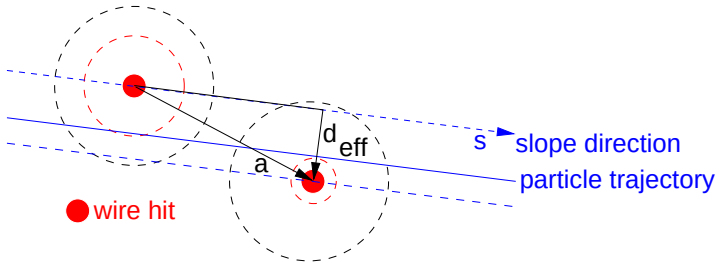


Figure 5.9: *Calculation of effective pitch. $\vec{a}$ is the vector joining the two wires, $\vec{s}$ is a unit vector in direction of the trajectory of the particle. Then, the effective pitch $\vec{d}_{eff}$ can be obtained by subtracting from $\vec{a}$ the projection of $\vec{a}$ onto $\vec{s}$.*

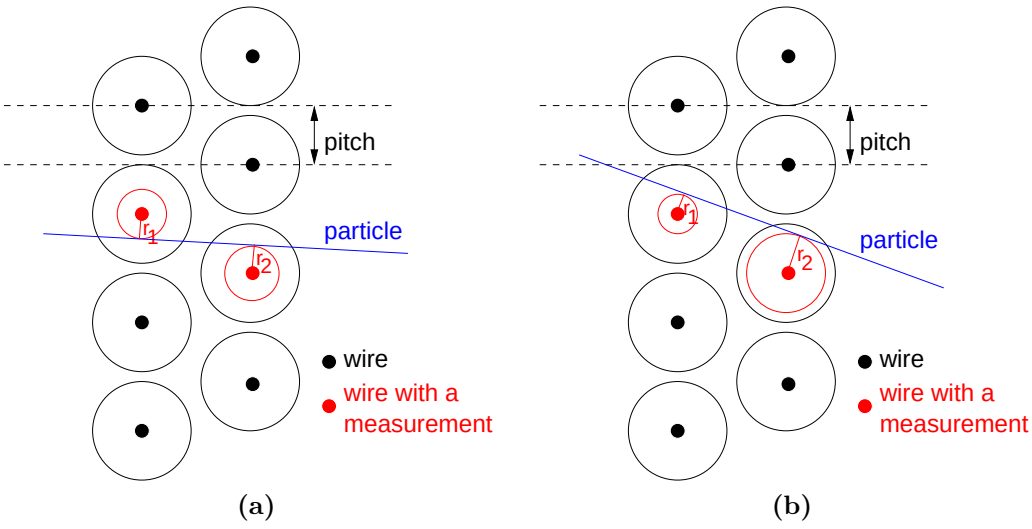


Figure 5.10: *For particles passing through the detector with a non-vanishing slope, there are two situations: the particle passes either between the two wires (a), or it does not (b).*

clusters which do not belong together (for these combinations, the slope will be wrong in most cases). Such a check will also be made in the next stage of the algorithm, therefore, a quantity needs to be defined which captures this behaviour — the *pitch residual*. It can be defined in terms of two quantities p_+ and p_-:

$$p_+ = r_1 + r_2 - d_{eff}(t_x)$$
$$p_- = |r_1 - r_2| - d_{eff}(t_x)$$

Using these quantities, the pitch residual can be written down:

$$\text{pitch residual} = \begin{cases} p_+ & \text{if } |p_+| < |p_-| \\ p_- & \text{otherwise} \end{cases}$$

Figure 5.11 shows clearly that the pitch residual distributions look different for best tracklets and wrong combinations. Note that the entries in the plots are weighted: Tracklets formed from two single measurements do not show up at all because pitch residuals are not defined for single measurements. For tracklets consisting of three or four single measurements, the contributions have been weighted with 1.0 and 0.5, respectively, to account for the fact that the tracklets with four single measurements contribute two pitch residuals.

For best tracklets, a Gaussian core is observed. The asymmetric shape of the distribution for wrong combinations is due the way the pitch residual is defined: $d_{eff}(t_x)$ is always subtracted.

Note that it is possible to estimate the detector resolution using the left plot of Figure 5.11. The core should have a width of $\sqrt{2}\sigma_{OT}$ where σ_{OT} is the resolution of the Outer Tracker. A simple gaussian fit to the core

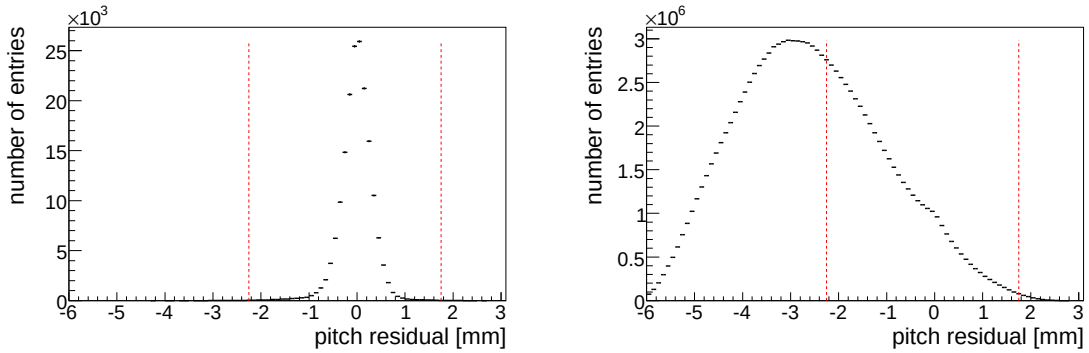


Figure 5.11: *Distribution of pitch residuals for best tracklets (a) and wrong combinations (b). The latter shows a different behaviour than (a).*

(|pitch residual| < 1.0) of the distribution gives $\sigma_{fit} = (0.300 \pm 0.003)$ mm; in the simulation, $\sqrt{2}\sigma_{OT} = 0.283$ mm (the simulation assumes 200 μm resolution). The remaining difference is due to the relatively broad interval over which the fit was performed.

This procedure might also be used later in the experiment to estimate the resolution of the Outer Tracker, preferably using tracks of high purity. However, one should probably not use an algorithm which cuts hard on the pitch residual to avoid bias.

In the standard configuration, the algorithm restricts pitch residuals to the range between −2.25 mm and 1.75 mm. This corresponds to a fraction of 99.87 % of all pitch residuals that went into the left histogram of Figure 5.11.

5.4.4 Efficiency of tracklet generation

Of course, the interesting question is if the procedure described above does indeed generate the desired tracklets. One defines the tracklet generation efficiency between two given x layers as

$$\varepsilon_{TL} = \frac{\#(F \cap R)}{\#R}$$

Here, R is the set of reconstructible particles, i.e. those for which a good tracklet can be formed in principle, and F is the set of particles for which a good tracklet was generated.

Using this efficiency definition, the average tracklet generation efficiency between two adjacent x layers turns out to be 99.60 % (tracklet-averaged, because cut inefficiencies are tracklet-averaged as well), just multiplying the cut inefficiencies given above (assuming they were independent) gives an expected efficiency of 99.68 %. The remaining discrepancy might be due to particles decaying between two layers.

The comparison above was done using tracklet-averaged quantities because the inefficiencies derived from the plots are tracklet-averaged quantities as well. The event-averaged efficiency is 98.65 %.

Thus, very few particles are lost at this stage, and the origin of the inefficiency is mostly understood. However, a loss at this stage does not mean a particle can not be reconstructed by the algorithm. It just means that one loses a tracklet between two layers, so the tracklet chain for that particle will be shortened or interrupted.

Table 5.1 again summarises the efficiency figures.

Cut on slope t_x	loss < 0.01 %
Cut on $x/(z - 5300\text{ mm}) - tx$	loss < 0.05 %
Cut on pitch residuals	loss < 0.26 %
Combined expected efficiency	99.68 %
Observed inefficiency (tracklet-average)	99.60 %
Observed inefficiency (event-average)	98.65 %

Table 5.1: *Origin of losses in the tracklet generation stage, observed efficiency. Expected and observed efficiency agree quite well.*

5.5 Stereo enhancement

During this phase of the algorithm, the tracklets in xz projection are enhanced by adding compatible stereo hits. This is a two-step process: First, pseudo-x clusters are formed from all geometrically possible two-cluster combinations in two adjacent stereo layers. These are added to compatible tracklets in a second step. Figure 5.12 contains an illustration of the principle. The two steps will now be described in more detail.

5.5.1 Forming pseudo-x clusters

Suppose that the trajectory of a particle passing between two stereo layers can be described by a straight line. Then, the coordinates of the two clusters that the particle should produce in the u and v planes are given by

$$u = \cos\alpha(x_{mid} + t_x(z_u - z_{mid})) + \sin\alpha(y_{mid} + t_y(z_u - z_{mid}))$$
$$v = \cos\alpha(x_{mid} + t_x(z_v - z_{mid})) - \sin\alpha(y_{mid} + t_y(z_v - z_{mid}))$$

Here, z_u and z_v are the z coordinates of the u and the v plane, α is the stereo angle, and x_{mid} and y_{mid} are the coordinates of the particle at $z_{mid} = \frac{1}{2}(z_u + z_v)$. Assuming that the parameters of the trajectory are initially unknown, these two relations can be used to get an initial estimate for x_{mid}:

$$x_{mid} = \frac{u + v}{2\cos\alpha} - \frac{\sin\alpha}{\cos\alpha} t_y(z_u - z_{mid})$$

The fact that z_{mid} is the midpoint between the two stereo planes has been used above. The last term is small: $|\sin\alpha|$ is on the order of 0.1, $|t_y|$ is smaller than 0.4, and any two neighbouring stereo layers are no further apart than 1 m; so the second term will be smaller than 2 cm in any case;

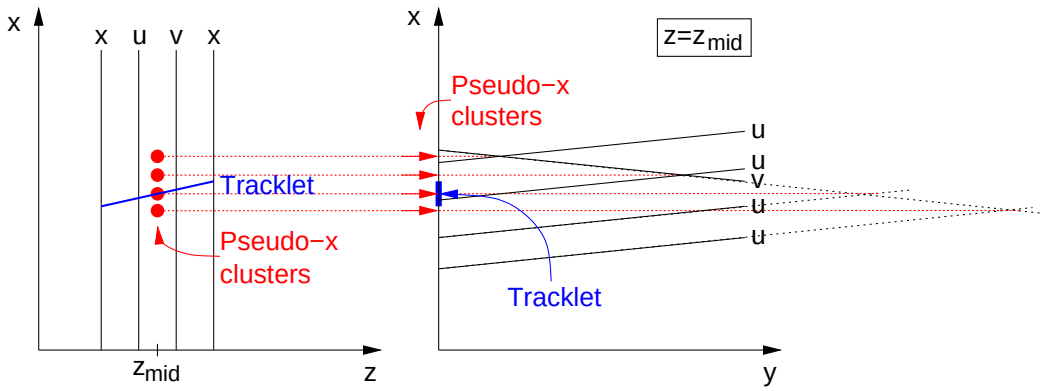


Figure 5.12: *Creation of pseudo-x clusters. On the right, some straws in u and v layers are combined in a yx projection view at $z = z_{mid}$ to form pseudo-x clusters. The x positions of the combinations indicated by the red arrows are at the intersection of the u and v straws (thought infinitely prolonged). The intersection can appear to have moved past the sensitive area of the straw in the plane at z_{mid} because the change in x position of the tracklet between the two stereo layers is not accounted for. As z_{mid} is in the middle between the two stereo layers, this effect cancels, and the x coordinate of the intersection point is relatively accurate. On the left, a view in xz projection is shown where the matching of a tracklet and the pseudo-x clusters takes place at $z = z_{mid}$.*

the exact value will of course depend on the slope t_y of the particle and the actual distance between stereo layers. These 2 cm are not much because they correspond to a window extending for less than four straws to each side.

Thus, neglecting the last term which can not be computed without more information, one can still obtain a reasonable estimate of x_{mid}. This makes it feasible to form pairs of stereo measurements from u and v layers and match them to compatible tracklets using x_{mid}.

To reduce combinatorics, only adjacent stereo layers are used (i.e. pairs of stereo layers with no further stereo layers in between), and only those pairs of clusters are considered for which it is geometrically possible that the corresponding straws have been hit by the same particle. From the cut on the slope t_x, the maximal x distance dx_{max} can be inferred that a particle can travel between the two stereo layers. The x distance between the wires of the stereo straws in question must be smaller than dx_{max} somewhere along the straws.

The resulting pseudo-x clusters are sorted by their x position to facilitate the matching to tracklets.

In this thesis, the term "long pseudo-x cluster" is used to refer to pseudo-x clusters constructed from stereo layers in different stations (the layers are separated by about 60 cm in z, hence their name), a "short pseudo-x cluster" consists of two stereo clusters from the same stations (separated by less than 10 cm in z).

5.5.2 Matching pseudo-x clusters and tracklets

A loop over all tracklets between the two x layers closest to z_{mid} is performed. If the pseudo cluster is consistent with the tracklet, it is added to the tracklet as stereo candidate: At the end of this stage, there are about 3.6 pseudo-x clusters consistent with a good tracklet on average, so it is fair to call them "stereo candidate". These stereo candidates will have the role of tracklets for the cellular automaton running in yz direction.

To check for consistency, a number of criteria are verified. The order discussed below again corresponds to what is done in the code, with computationally cheap checks first, the more expensive ones later. The efficiency of this stage of the algorithm is discussed last.

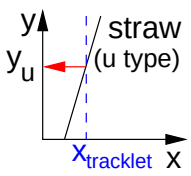
Matching tracklets and pseudo-x clusters

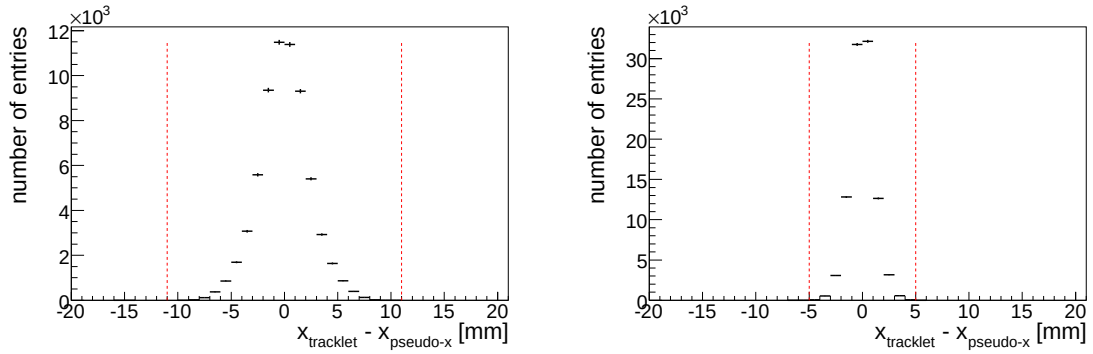
For an initial matching of tracklet and pseudo-x cluster, the algorithm compares how well the x positions of the two agree at $z = z_{mid}$ (cf. Figure 5.12). Figure 5.13 shows the difference in x position for best combinations of tracklets and pseudo-x clusters. The plot on the left contains "long" pseudo-x clusters, the plot on the right shows the same quantity for "short" pseudo-x clusters. One can see that the matching is better in the "short" case because the trajectory of a particle matches the straight line model better over short distances.

The algorithm accepts a combination of a pseudo-x cluster and a tracklet if the difference is less than 5 mm in the "short" and 11 mm in the "long" case. One loses less than 0.01 % of best combinations from this cut.

Restricting y coordinates to sensitive range of straws

Now that a parametrisation of the trajectory in xz projection is known, y coordinates y_u and y_v can be calculated for the stereo clusters forming the pseudo-x cluster. If they fall into the y range in which the corresponding



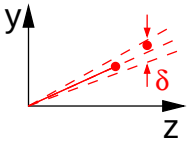


(a) (b)

Figure 5.13: *Difference in x position of pseudo-x clusters and tracklets for best combinations, (a) is for "long" pseudo-x clusters, (b) is for "short" ones formed from clusters in the same station.*

straws are sensitive, the combination of this tracklet with the pseudo-x cluster is investigated further.

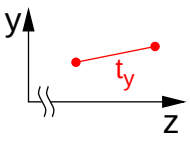
Vertexing cut on y coordinate



For all pseudo-x clusters, the y coordinate of the second measurement must be within a window δ around the line in yz projection which joins the first measurement of the pseudo-x cluster with the origin. Effectively, this imposes the restriction that the pseudo-x cluster points to the primary vertex in yz projection.

Figure 5.14 shows the difference between the y coordinate of a stereo cluster and its prediction derived from another stereo cluster under the vertex assumption. It can be seen that the accuracy of the prediction is slightly better in the "short" case. A combination of tracklet and pseudo-x cluster is accepted if this difference is smaller than 130 mm in the "short" or 150 mm in the "long" case. This cut introduces an inefficiency smaller than 0.09 % in the "long" case for the Monte Carlo sample studied, for the "short" case it is even less.

Cut on slope t_y (without vertex assumption)



For "long" pseudo-x clusters that were built from stereo layers in different OT stations, the lever arm is long enough to obtain an estimate of the slope t_y from y_u and y_v alone, without assuming anything about the primary vertex.

Figure 5.15 shows the slopes t_y derived without the vertex assumption for both "long" and "short" best combinations, In the first case, the lever

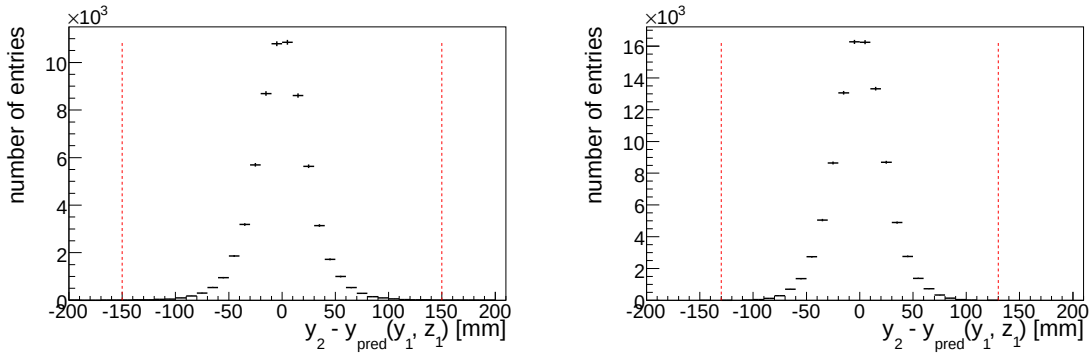
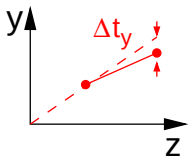


Figure 5.14: *Plot showing difference of prediction of y position of second stereo cluster and prediction of that position from first stereo cluster and a vertex assumption. (a) is for "long", (b) is for "short" best pseudo-x clusters.*

arm is long enough to obtain a reasonable estimate of the slope. For "short" pseudo-x clusters however, the distribution is broadened by the finite spatial resolution in y direction and the short lever arm. It is still possible to reduce background in both cases; cuts have been placed such that $|t_y| < 0.3$ or $|t_y| < 1.5$, respectively, to achieve this goal.

This cut will lose less than 0.07 % of best pseudo-x clusters.

Vertexing cut on slope t_y



An additional vertexing cut is applied: The difference between the slope calculated above (t_y^{clus}) and the slope of the line in yz projection which joins the first cluster with the origin (t_y^{vtx}) should be small.

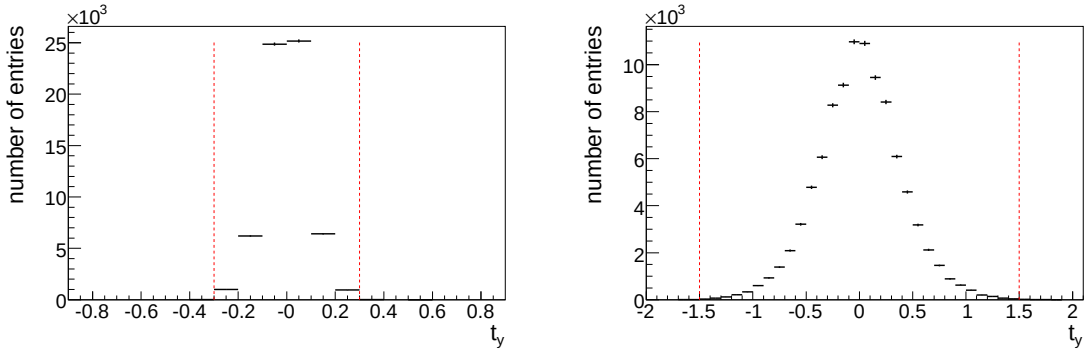
Figure 5.16 shows the results for "long" and "short" best pseudo-x clusters. Cuts have been placed such that $|\Delta t_y| = |t_y^{vtx} - t_y^{clus}| < 0.2$ and $|\Delta t_y| < 1.5$, respectively.

The loss in best tracklets introduced by this cut is less than 0.14 %.

Cut on the pitch residuals

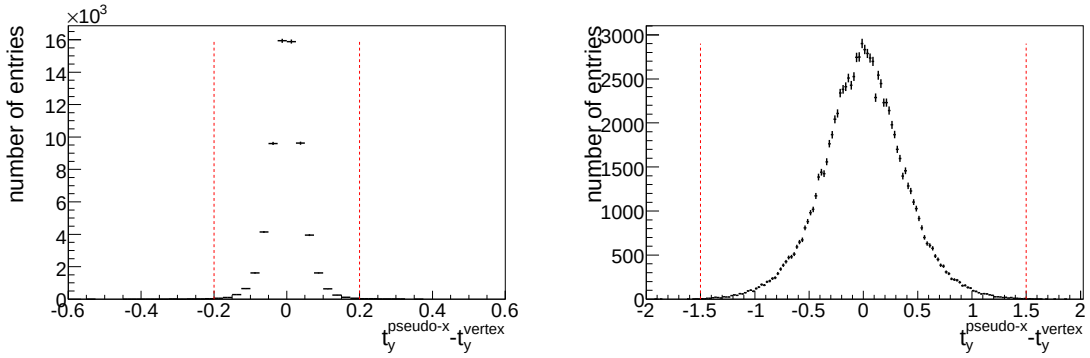
If all of the previous conditions were fulfilled, the pitch residuals of the stereo clusters are checked as discussed in Section 5.4.3, using the slope of the line connecting the cluster with the primary vertex in yz projection and the tracklet slope in xz to estimate the slope in measurement direction.[6]

[6]The distributions are not shown because they looks exactly like those in Figure 5.11. In principle, the tracklet slope alone would be sufficient because the admixture of t_y to the



(a) **(b)**

Figure 5.15: *Slope t_y for best combinations, calculated using stereo cluster y positions obtained from stereo coordinate and tracklet in xz projection. (a) is for "long" pseudo-x clusters. (b) for "short" ones shows the effect of worse spatial resolution in y and a short lever arm, widening the distribution.*



(a) **(b)**

Figure 5.16: *Plots showing the difference in slopes t_y derived with and without vertex assumption. (a) is for "long" best pseudo-x clusters, (b) is for "short" ones.*

Given the shape of the pitch residual distribution for best combinations, it is unlikely that more than one or two of the up to four available pitch residuals in a combination of tracklet and pseudo-x cluster are far from 0 mm. To exploit this, the algorithm calculates the product of all pitch residuals in such a combination.

Figure 5.17 shows the distribution of this product for best and wrong combinations of tracklets and pseudo-x clusters. Single measurements in a layer contribute a factor of 1.0 mm to the product; this has been chosen because multiplying with unity (when calculating in millimetres) does not affect the value of the product. A combination of a tracklet and a pseudo-x cluster is accepted if this product is smaller than 0.5 in magnitude.

Less than 0.18 % of best pseudo-x clusters are lost on the sample used to produce these plots.

5.5.3 Stereo enhancement efficiency

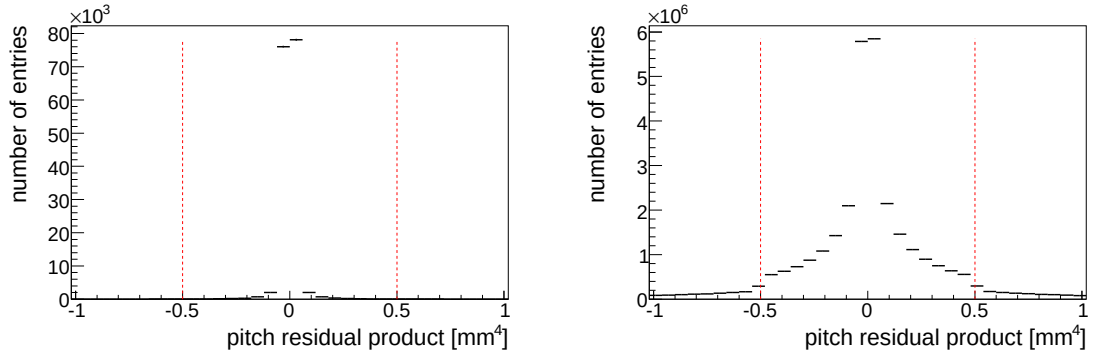
Reconstructible particles are all particles considered reconstructible during tracklet formation in the corresponding x layers. Additionally, these particles must have measurements in both stereo layers.

A particle is considered to be reconstructed if a pseudo-x cluster has been generated with measurements from that particle in both stereo layers, and the pseudo-x cluster must have been matched to a good tracklet coming from that same particle. Thus, the stereo enhancement efficiency must be smaller than the tracklet generation efficiency because the latter is required.

Using the estimates for the cut inefficiencies derived from the plots above and the observed tracklet generation efficiency from the last section, one would estimate the combined inefficiency for the stereo enhancement stage to be 99.10 %. Using the definition given above, one observes 99.08 % efficiency on the sample that was used to make the plots. In fact, the two numbers agree quite well. Table 5.2 lists the contributions again.

The loss of a pseudo-x cluster does not imply that a particle can not be reconstructed, so the algorithm can afford to suppress background more aggressively than during the tracklet generation stage.

effective slope is small due to the small stereo angle. The algorithm uses the additional information about the slope in yz only because it has been calculated earlier on.



(a) **(b)**

Figure 5.17: *Product of pitch residuals from a tracklet and a pseudo-x cluster. Single measurements contribute a factor of 1 mm. (a) is for best combinations, (b) for wrong ones. In (b), the peak comes from the core of the pitch residual distribution which has been isolated by cuts, but it is much wider than in (a) and also has tails.*

loss inherited from tracklet generation	0.41 %
matching of tracklets and pseudo-x clusters	< 0.01 %
vertexing cut on y	< 0.09 %
cut on t_y	< 0.07 %
vertexing cut on t_y	< 0.14 %
cut on product of pitch residuals	< 0.18 %
expected efficiency	99.10 %
observed efficiency	99.08 %

Table 5.2: *Origin of losses during the stereo enhancement phase, expected and observed efficiency. Expected and observed efficiency agree quite well.*

5.6 Finding neighbours and automaton evolution

This section describes how the algorithm proceeds to find neighbours for each tracklet. The evolution of the cellular automaton can be done at the same time.

To decide if a tracklet is a neighbour of another tracklet, the following checks are made:

- for two tracklets to be neighbours in xz projection, they must share the cluster in the middle[7]
- the kink angle (in xz projection) between the two tracklets must be small enough
- if there is only one single measurement in each of the three x layers involved, at least one of the two tracklets must have a pseudo-x cluster confirming it[8]
- the estimates for q/p of both tracklets obtained during the tracklet generation phase using the p_T-kick method must agree within a certain tolerance
- if both tracklets have stereo information from the last stage of the algorithm, there must be at least one combination of pseudo-x hits for which the two tracklets are compatible in yz projection

The cuts are discussed in more detail in the subsections below. Then, a description will be given of how the evolution phase of the automaton can be speeded up compared to the method outlined in the last chapter. The discussion will focus on tracklets in xz projection first; the pseudo-x hits in yz direction will be treated in a slightly different way which will be described last.

[7]This is due to the fact that a particle trajectory intersects a plane in exactly one point, as discussed in the last chapter.

[8]This cut is based on the fact that most particles produce two measurements per layer, therefore, the fraction of particles having only three x measurements and no pseudo-x clusters in a total of six layers is negligible. A lot of combinatorics is suppressed, though.

5.6.1 Cuts on kink angle and q/p

Figure 5.18 shows the kink angle between two tracklets in xz projection. The left plot is for the best combination[9], the right one is for all tracklet combinations with a shared middle cluster and either at least one bad tracklet or two good tracklets from different particles.

One can see that the category of best combinations is dominated by small kink angles; therefore a cut has been placed admitting combinations with a kink angle smaller than 50 mrad. About 99.43 % of all best combinations are kept by the cut.

Figure 5.19 shows the difference between the estimates for q/p of two tracklet combinations. A cut has been placed such that $|q_1/p_1 - q_2/p_2| < 10^{-4}$ MeV^{-1}. This cut will keep 99.80 % of best combinations.

5.6.2 Cuts used to check for stereo compatibility

To decide if two pseudo-x clusters are neighbours in yz projection, three checks are made:

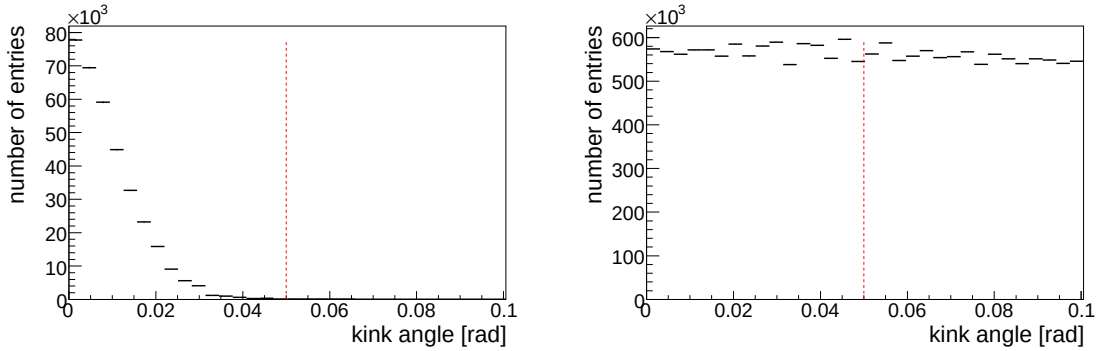
- The line in yz projection joining the two pseudo-x cluster midpoints must have a slope of less than 0.4 in magnitude. The midpoint of a pseudo-x cluster is at $y = (y_u + y_v)/2$ and $z = (z_u + z_v)/2$.
- The difference in slope between the line just mentioned and the slope of the "longer" pseudo-x cluster must be less than 0.3 in magnitude. Here, the slope of a pseudo-x cluster can be defined as $(y_u - y_v)/(z_u - z_v)$.
- Finally, the y position of the second midpoint must fall into a window around the line joining the first midpoint with the primary vertex. This window extends for 110 mm on both sides of this line.

Figures 5.20a, 5.20b and 5.20c illustrate the distributions and the cuts made. Less than 0.01 % of best tracklet combinations are lost in each of the three cases.

5.6.3 Automaton evolution

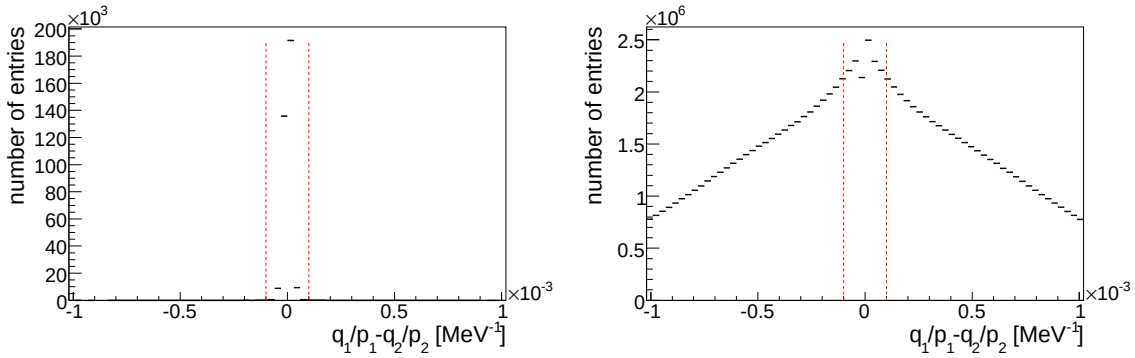
In the last chapter, the evolution of cellular automata was described in terms of discrete automaton time steps or generations. While this is adequate for Conway's "Game of Life" (where one is interested in the "dynamics") and well suited to run on specialised hardware (a dedicated microchip could

[9] combination of two best tracklets from the same particle



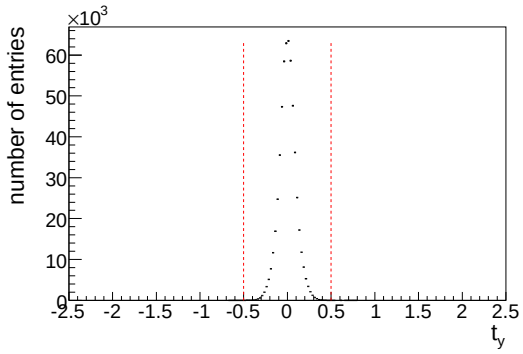
(a) (b)

Figure 5.18: *Kink angle distributions for two tracklet combinations in adjacent layers. Best combinations are shown (two best tracklets) in (a), in (b) shows wrong combinations (at least one bad tracklet or two good tracklets from different particles). One demands the kink angle to be less than 50 mrad for a combination to survive. The fact that the curves are not smooth (most visible for wrong combinations) is again due to a binning effect caused by a discrete set of tracklet slopes t_x (the effect is explained in 5.4.1).*

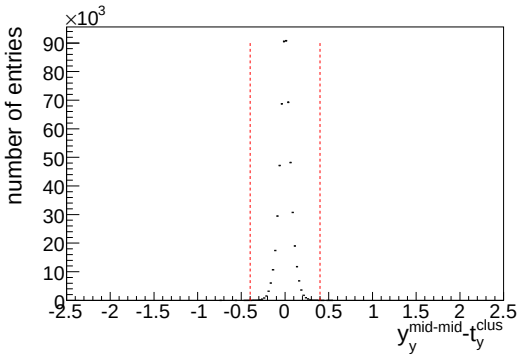


(a) (b)

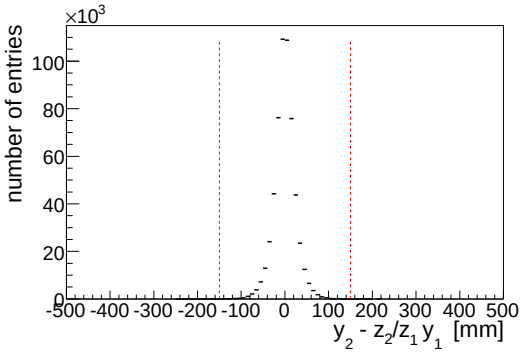
Figure 5.19: *Distributions of difference in q/p for two tracklet combinations in adjacent layers. Best combinations (two best tracklets) are in (a), in (b), wrong combinations are shown (at least one bad tracklet or two good tracklets from different particles). One demands that this difference is smaller than 10^{-4} MeV^{-1}.*



(a) *Slope distribution in yz projection derived from the coordinates of two pseudo-x cluster midpoints for best combinations.*



(b) *Distribution of the difference in slope t_y between the line joining the two pseudo-x cluster midpoints and the "long" pseudo-x cluster.*



(c) *Difference between the y position calculated for the midpoint of the second pseudo-x cluster and its prediction from the first pseudo-x cluster and the vertex assumption.*

Figure 5.20:

perform one step of the evolution of a cellular automaton per clock cycle in parallel for all cells), it is not the optimal solution for a software tracking algorithm. In fact, the rules of the cellular automaton used in track finding permit the result of the evolution to be anticipated in a single step layer after layer, without actually determining the states of the tracklets in each generation, waiting for them to become stable. The key to this shortcut are two observations:

1. Due to the fact that the neighbourhood relation is asymmetric (i.e. it has a direction), the state of a tracklet can only depend on tracklet states in the previous layer.

2. The counter of each tracklet is incremented during the evolution in such a way that it is always larger by exactly one than the highest counter in its neighbourhood.

So the counters in a layer can be set to their final values by looking at the counters of the neighbours as one finds them (provided that the same has been done for the neighbours, too).

There is one additional optimisation that can be performed: As has been described in the last chapter, the algorithm will enforce the "counter must decrease by one"-constraint along a tracklet chain when looking for track candidates, so the algorithm can just discard any neighbours for which this this constraint does not hold. The remaining stages of the algorithm do not need to check that this constraint is fulfilled, thus saving time.

5.6.4 Automaton evolution in stereo layers

Basically, one would like to have a separate cellular automaton searching for a track in yz projection for each track candidate that can be produced in xz projection. To achieve this, each tracklet stores counters and neighbour relations for each pseudo-x cluster that has been added in the stereo enhancement step.

Pseudo-x clusters found to be compatible in the sense of 5.6.2 are made neighbours; the evolution works just like in the tracklet case.

5.6.5 Performance evaluation

To evaluate the performance of this stage of the algorithm, particle reconstructibility and reconstructedness must be defined. A particle is considered reconstructible inside a layer for the neighbour finding stage if it has produced two good tracklets, one which starts and one which ends in the layer

in question. The particle is considered to be reconstructed (correctly) if there is at least one neighbourhood relation between two good tracklets from that particle which survived the "counter-must-decrease-by-one" constraint imposed by the automaton evolution.

Please note that no explicit constraints are imposed on the stereo part. Such a constraint is imposed implicitly by only considering those tracklets neighbours which are compatible in stereo, see above.

From the cuts imposed on the xz portion of the two tracklet combinations, one would expect an efficiency of 99.22 %. Using the efficiency definition above, 96.41 % are observed. The reason for this discrepancy is due to the requirement of stereo compatibility: Most tracklets do have stereo information, but in about 3 % of all cases, the correct pseudo-x cluster is missing while a wrong pseudo-x cluster is present. This can be caused by detector inefficiencies, measurements that have been lost by the cut on the drift radius or inefficiencies of the stereo enhancement phase. Not requiring compatibility in stereo will bring the efficiency of the neighbour finding phase to the expected value, but the increase in combinatorics caused by this roughly triples the overall execution time of the algorithm. Still, the ghost fraction stays moderate and overall efficiency improves when not requiring stereo compatibility at this stage (one finds about 5 % more particles).

At this point, it is interesting to observe that each tracklet has less than one neighbour on average — this will make the phase forming candidates quite fast, because there are not many alternatives to follow. Figure 5.21 shows the distribution of the number of neighbours per tracklet. The first "layer" of tracklets (the one with all counters zero) has not been included in this plot because these tracklets can not have neighbours by definition.

5.7 Forming and selecting track candidates

In this section, a description of the process used to form and select candidates will be given. Forming candidates will be described first, then a subsection on selecting the good ones is to be followed by a short discussion of the overall performance of this step.

To simplify wording in the following text, it is convenient to state a few conventions:

A candidate is said to be *good* if it has been reconstructed, i.e. it can be associated to a particle and satisfies the requirements stated in 5.2. The *best* candidate for a particle is the good candidate with most correct measurements. A *bad*, *wrong* or *ghost* candidate is not a good candidate.

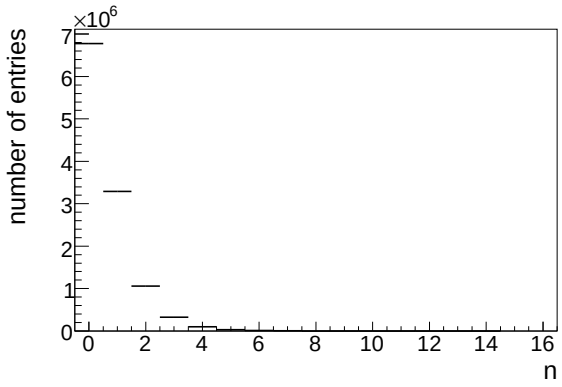


Figure 5.21: *Number of neighbours per tracklet. Note that the first bin shown is for no neighbours at all.*

5.7.1 Forming track candidates

Track candidates, or candidates for short, are formed by following the neighbourhood relation as described in the last chapter. For each tracklet in xz projection that is added, there may be several track candidates, depending on the number of pseudo-x clusters that the tracklet in question has. The clusters that are added to a track candidate in the process are also used to obtain an estimate of the track parameters using a least squares fit. In the process, some cuts are applied to make sure that no time is wasted on track candidates which are of very low quality. The surviving track candidates are passed on to the selection phase.

The remainder of this section will discuss the stages of following the neighbourhood relation to form candidates, the fit and the cuts applied in the process.

Following the neighbourhood relation

The essence of the tracklet following procedure has been described in the last chapter, what is still missing is a description of the way in which the stereo measurements are treated. This is done by building a "tree" of candidates: Starting from one tracklet as the root, each possibility to continue the track candidate results in a branch of that tree. The paths from the "leaves" to the root of the tree are the track candidates which are passed on to the selection phase.

This is realised easily by using a recursive procedure which gives a rule how to treat a tracklet which is to be added to a candidate.

Given a candidate and a tracklet to be added to it, the tracklet following procedure works like this:

1. add the clusters in xz projection to the candidate — this means adding them to the fit and keeping pointers to the measurements involved
2. for each pseudo-x cluster that the tracklet holds, make a new branch and add the individual stereo clusters to the resulting candidates
3. check that the resulting candidates still satisfy the quality cuts to be described in the next section
4. upon arrival at a leaf (i.e. a tracklet without neighbours), do some final quality cuts, attempt to remove outliers, and pass on the resulting candidate to the selection phase, breaking the recursion
5. for each of the surviving candidates and each of the neighbours of the tracklet added in the first step, repeat this procedure

Candidates are formed starting with tracklets with the highest counter to make sure long tracks are favoured.

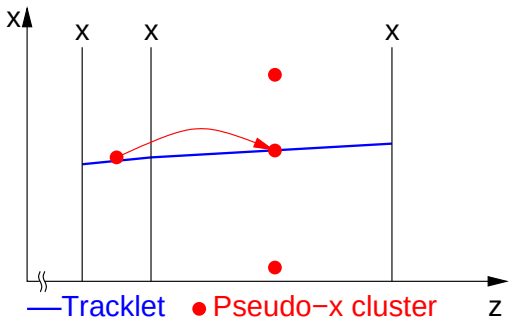
The fitting procedure and the cuts applied will be discussed later, but there is one remark about adding pseudo-x clusters that should be made at this point.

In section 5.6.4, a description of the neighbourhood relation of pseudo-x clusters was given. This information can be used to speed up the tracklet following process by remembering the pseudo-x cluster added previously:

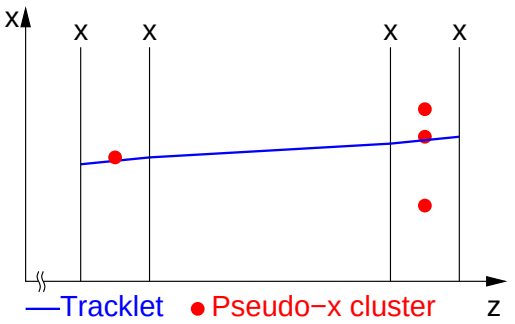
When adding the first pseudo-x cluster, all pseudo-x clusters associated to the current tracklet are used, as described above. If a pseudo-x cluster was added during the last step, its neighbours are known, so all others can be skipped (Figure 5.22a). The advantage of doing this is that time is better spent elsewhere instead of calculating a χ^2 for a candidate only to discard it immediately because the fit is bad.

As one does not want to suffer from interrupted chains of pseudo-x clusters, the algorithm permits starting over using all pseudo-x clusters in a tracklet if there was no pseudo-x cluster added during the last step (Figure 5.22b).

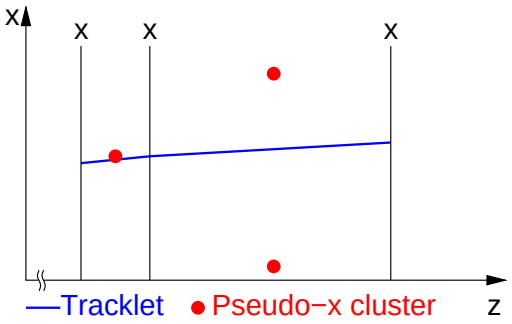
However, such a "restart" is only allowed if the algorithm did not run into a "dead end" (Figure 5.22c). Such "dead ends" arise when a pseudo-x cluster has no neighbours but the next tracklet does have pseudo-x clusters in principle. The algorithm discards track candidates which cause it to run into "dead ends".



(a) *When following tracklets, pseudo-x cluster neighbour relations are obeyed where available. In the picture, only the two pseudo-x clusters connected with the arrow are neighbours, the other two are skipped.*



(b) *When encountering missing pseudo-x clusters in a layer, the algorithm restarts with whatever is available in the next layer.*



(c) *A dead end: The tracklet in the middle of the picture has pseudo-x clusters, but none is a neighbour of the one on the left. Candidates which lead into dead ends are dropped.*

Figure 5.22: *Following the neighbour relation of pseudo-x clusters.*

This design choice cuts down on execution time drastically (by a factor on the order of three to four, depending on the event), while not decreasing performance.

This "dead-end"-recognition may have to be reconsidered in case of much higher occupancies and/or lower detection efficiencies: Coincidence of one or more wrong pseudo-x cluster and a missing good one will cause the candidate for the particle in question to be discarded, so it may become necessary to find a different strategy there.

Fitting track candidates

Each candidate is fitted to a simple track model: In xz projection, a parabola is used to account for the track curvature in the fringe field of the magnet, in yz projection, a straight line is a reasonable approximation.

A simple χ^2 fit is used. To obtain the fit parameters, a system of linear equations needs to be solved. This system is updated incrementally as more measurements are added. Only x measurements are used for the parabolic fit, and stereo measurements are used together with x information from the parabolic fit for the straight line fit in yz projection.

The fit itself is done in several stages:

Initial fit using clusters For the initial fit, clusters are used to obtain fit parameters rather than using the individual measurements. The advantage of this approach is that the clusters resolve the ambiguity in many cases. In section 5.4.3, the pitch residual was introduced as a way to tell if a cluster is consistent with a slope estimate. In particular, by looking at the quantities p_+ and p_- defined there, the algorithm can tell if a track passes between two wires ($|p_+| < |p_-|$) or if it does not (cf. Figure 5.10). In the first case, it is clear how to resolve ambiguities. In the second case, two pairs of ambiguity resolved positions on the drift circles are calculated, one for each of the two remaining possibilities. Then, the algorithm chooses the pair which agrees better with the slope observed during tracklet generation. Such a pair of points is shown in Figure 5.23, where the points are labelled P_1 and P_2. The midpoint P_{fit} of the line segment joining P_1 and P_2 enters the fit. Its measurement uncertainty is estimated according to the following formula:

$$\sigma = \frac{1}{2}\sqrt{2\sigma_{OT}^2 + d_1^2 + d_2^2}$$

where σ_{OT} is the Outer Tracker cell resolution[10] and the distances d_1 and d_2 are also defined in Figure 5.23.

For single measurements, a measurement uncertainty of

$$\sigma = \sqrt{(2r)^2 + \sigma_{OT}^2}$$

is assumed where r is the drift radius observed in that cell[11]. For single measurements, the wire position enters the fit.

Refit The refit works on the level of individual single measurements rather than the cluster level. Using the old fit parameters as starting points, a new fit is performed. Ambiguities are resolved towards the trajectory defined by the set of old fit parameters. The old parameters are also used to correct for the tilt of the Outer Tracker with respect to the LHCb coordinate frame[12].

[10]The cell resolution is taken to be 300 μm. This is a little worse than the design goal of 200 μm because wire propagation time is not taken into account.

[11] Note that no division by $\sqrt{12}$ is performed — the true position will not be distributed uniformly over the interval from $x_{wire} - r$ to $x_{wire} + r$. In most cases, the true position will be concentrated in a small area around these two points because most particles traverse the detector more or less perpendicular to the detector layers.

[12]The LHC tunnel is tilted by 3.6 mrad with respect to the horizontal plane defined by gravity. The Outer Tracker has been mounted onto its support structure in such a way that the modules hang without bending to avoid excessive stress — unfortunately, this means that for the Outer Tracker being in the frame defined by gravity and the LHCb x axis, the y and z axes are tilted with respect to the ones defined in the LHCb coordinate system.

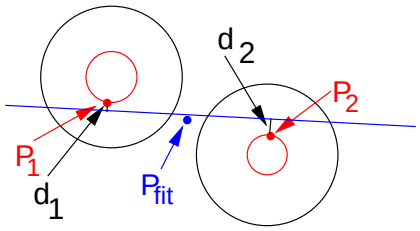


Figure 5.23: *Sketch illustrating the position of the points P_1, P_2, P_{fit} and the distances d_1 and d_2. P1 and P2 are the best guess for the ambiguity resolved positions on the drift circles obtained from a cluster (see text). d_1 (or d_2) is the distance of P_1 (or P_2, respectively) to the line through the cluster position with the slope of the tracklet that is considered (blue line). P_{fit} it the midpoint of the line segment joining P_1 and P_2.*

For the refit, points and distances of closest approach are calculated using the old fit, linearised around the wire position. Again, an Outer Tracker resolution of 300μm is assumed.

Outlier removal Outlier removal is rather simple: If a certain threshold in χ^2 contribution is exceeded, the (single) measurement with the largest χ^2 contribution is removed from the fit. Here, the χ^2 contribution of a measurement, denoted χ^2_+, is defined as

$$\chi^2_+ = \left(\frac{m - x_{fit}(z)\cos\alpha - y_{fit}(z)\sin\alpha}{\sigma_{OT}} \right)^2$$

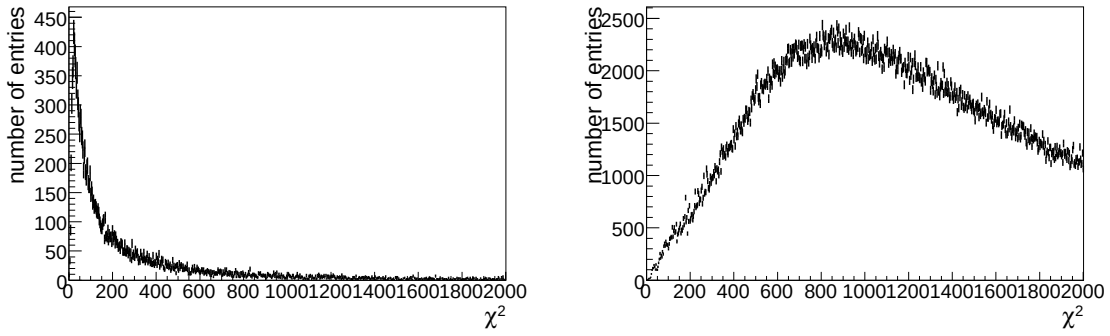
where the point (m, z) is the point on the drift circle that is closest to the trajectory defined by the fit parameters. m stands for x, u or v, depending on the type of measurement considered, and α is the corresponding stereo angle. $x_{fit}(z)$ and $y_{fit}(z)$ are the coordinates of the parabolic and straight line fits at the given z.

The resulting set of updated fit parameters is used to re-resolve the ambiguities of the remaining measurements, and the positions thus obtained are used for the subsequent refit. This continues until either all measurements are below the threshold or more than a given maximum number of measurements have been removed.

Candidate quality cuts

The algorithm uses several cuts to suppress bad candidates. There are two categories: The first one makes sure that combinatorics remains at a manageable level during the tracklet following procedure, the second category looks at the candidates before they are passed on to the selection phase, discarding bad ones.

Cuts used during tracklet following During the phase in which candidates are formed by following the neighbourhood relation, some very loose cuts are applied to reject candidates which have little resemblance to the tracks typically produced by a reconstructible particle. Figure 5.24 shows the distribution of the variable χ^2 for best and ghost candidates. This plot has been made with no cuts on χ^2 applied. Due to the large increase in combinatorics, only 500 events were used. It can be seen that ghost candidates are the main contribution for high χ^2 values, a trend that continues also for higher χ^2 than those shown in the plot. To suppress these ghosts, the following cuts are applied:



(a) **(b)**

Figure 5.24: *χ^2 distribution before cuts for best (a) and wrong (b) candidates. Only 500 events were used to produce this plot due to the increased combinatorics.*

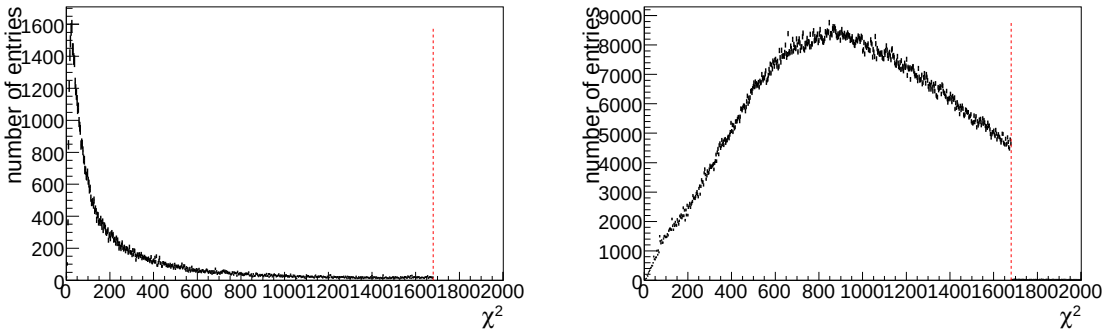
- One does not want measurements to be more than 1 cm away from the fit, therefore the overall increase in χ^2 per added combination of tracklet and corresponding pseudo-x cluster must not exceed $(1\text{ cm}/0.3\text{ mm})^2 = 1111$ times the number of measurements added. One centimetre was chosen to allow for outliers and fit parameter fluctuations when adding the first few measurements.

- As the initial fit takes ambiguities into account, the measurements should appear on the correct side of the wire, i.e. one does not want them to be more than a straw radius away from the fit on average. The algorithm takes advantage of that by enforcing a χ^2 budget for each candidate: The total χ^2 must not exceed $(2.5\text{ mm}/0.3\text{ mm})^2 = 70$ times the maximal number of measurements that a candidate can have. One assumes that a candidate starting with a tracklet with counter c can have at most $4(c+2)$ measurements. For chains of five tracklets, this means cutting at $\chi^2 = 1680$.

- Not more than 10^5 decisions per candidate are allowed. This is counted by multiplying up the number of neighbours and pseudo-x clusters that are followed at each step along the path taken to form the candidate.

Figure 5.25 shows the χ^2 distribution after the cuts. The tail for ghost candidates is truncated by the cuts.

The windows defined by the first two cuts are still huge compared to the detector resolution because outliers have not been removed and ambiguities may still be wrong before the refit.



(a) **(b)**

Figure 5.25: *χ^2 distribution after cuts for best (a) and wrong (b) candidates. The large tail for ghosts is truncated.*

The last cut protects the algorithm from high occupancy regions in the detector which can cause long algorithm execution times and exhaustion of available virtual memory in very rare cases. This last cut has been found not to affect the efficiency of the algorithm in any significant way. This can be understood easily enough: Picking the correct alternative among 10^5 possibilities is unlikely, even when using sophisticated quality indicators.

From the inefficiency of the tracklet generation stage alone, less than 93.4 % (event-averaged) of all particles with hits in all 6 x layers would be expected at this stage[13]. This can be estimated using ε_{TL}^5 (the event-averaged ε_{TL}), assuming the losses during tracklet generation in different layers are independent.

One observes 90.6% on that sub-sample after the cuts. The ghost fraction is 27.8%.

Thus, only a small fraction of the stereo inefficiency that affects neighbour finding is seen at this stage. This is not implausible: A neighbouring best tracklet that was missed does not mean the track is not reconstructed. Any nearby tracklet that fills such a gap can be used to form a tracklet chain.

Cuts used before the selection phase Before a track candidate is passed on to the track selection phase, there are a number of additional cuts it has to pass:

- A candidate with n tracklets must have at least $2n$ measurements. The

[13]It makes sense to do the comparison on this subsample because, ultimately, one is interested in how much is lost by these cuts. The overall performance of the algorithm is not a very good indicator because the algorithm has no provisions for coping with empty layers. Thus, the losses seen would be caused only in part by these cuts.

distribution of the number of measurements for best and for wrong candidates is shown in Figure 5.26 for the five tracklet chain case. Scaling the cut with the number of tracklets makes sure that the algorithm does not become much stricter for shorter chains.

- The candidate must have more than four stereo measurements. This is just a little more than is required to define a straight line in yz projection. Again, for chains of five tracklets, the distribution of the number of stereo measurements is shown in Figure 5.27.

- There must be at least two Outer Tracker stations with stereo measurements to make sure that there is sufficient lever arm to obtain an accurate description of the track in yz projection.

- The number of missing measurements must be less than $2\,(n-1)$ for chains of n tracklets. The number of missing measurements is calculated as $4\,(n+1)$ minus the number of measurements that candidate has, so one assumes that the candidate can have up to two measurements per layer. Figure 5.28 shows the distribution of the number of missing measurements for best and wrong candidates, both for tracklet chain lengths of five tracklets.

- After a refit, the outlier removal procedure may not remove more measurements than there are tracklets on the track. A measurement is considered to be an outlier if the χ^2 contribution is greater than 15. Figure 5.29 shows the number of measurements removed for best and wrong candidates, again for chains of five tracklets. Figure 5.30 shows the effect of the outlier removal on the χ^2 distribution.

- After the outlier removal, the algorithm verifies that the conditions imposed before the outlier removal still hold.

The surviving track candidates are passed on to the track selection phase. 90.2 % of all particles with all 6 x layers hit are still present after these cuts, while the ghost fraction has dropped to 13.9 %.

5.7.2 Track selection

After track candidates of a given length have been found, a track selection phase is needed to bring down the ghost fraction and suppress clones. This is done by assigning a quality variable to each candidate. The candidates are sorted by quality, then the algorithm can pick them in decreasing order, checking that the fraction of measurements used by candidates that have

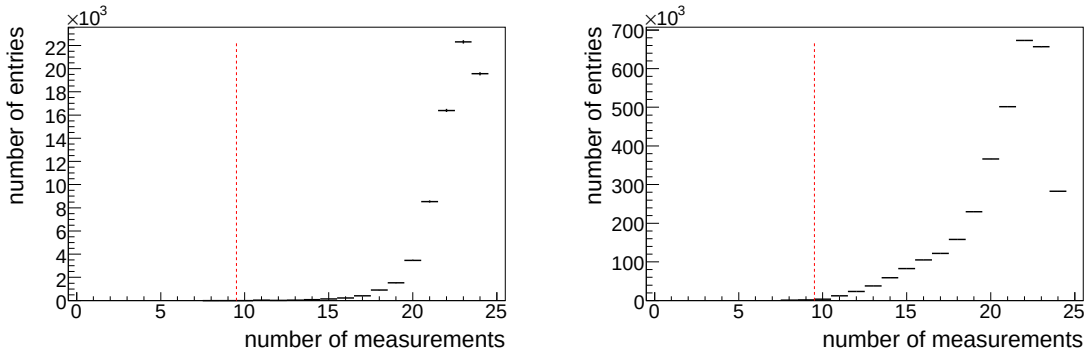


Figure 5.26: *Number of measurements per candidate. (a) is for best candidates, (b) for wrong ones. While the cut does little for the five tracklet chains considered here, its effects become stronger for shorter chains.*

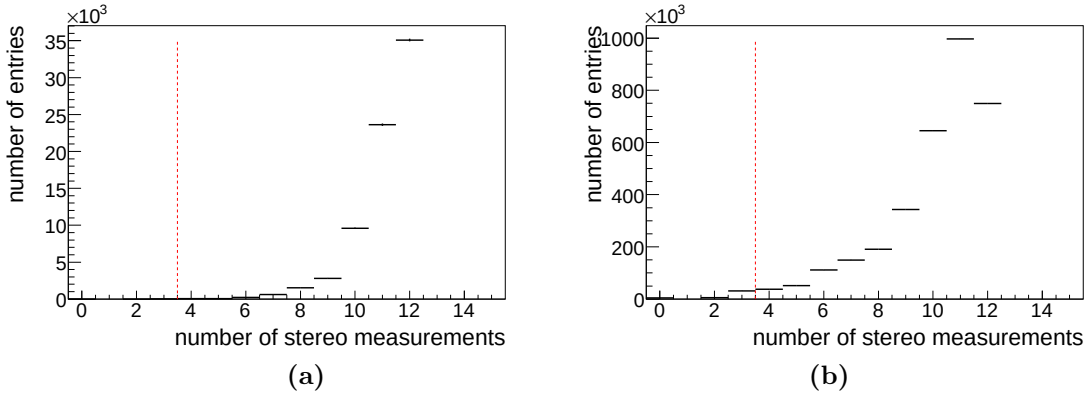


Figure 5.27: *Number of stereo measurements per candidate. (a) is for best candidates, (b) for wrong ones.*

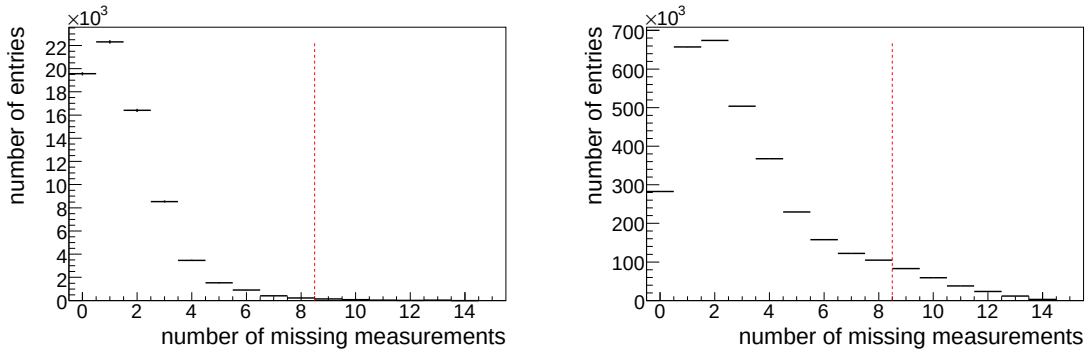


Figure 5.28: *Number of missing measurements per candidate. (a) is for best candidates, (b) for wrong ones.*

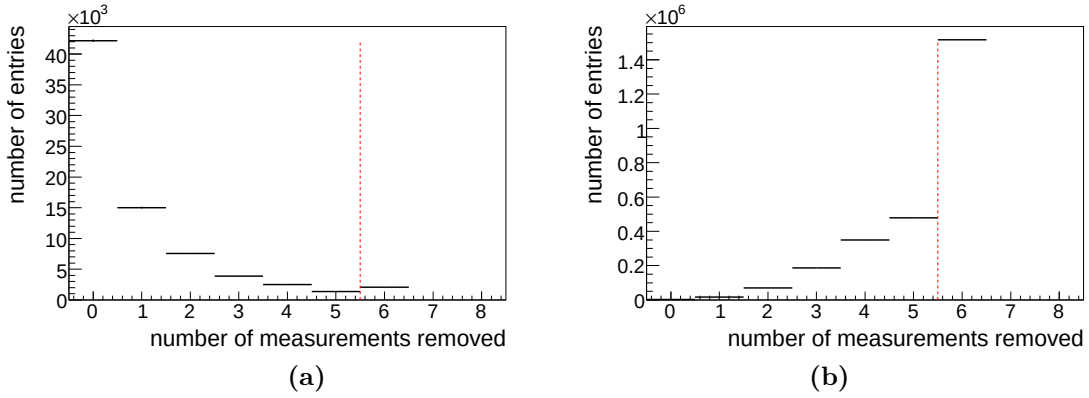
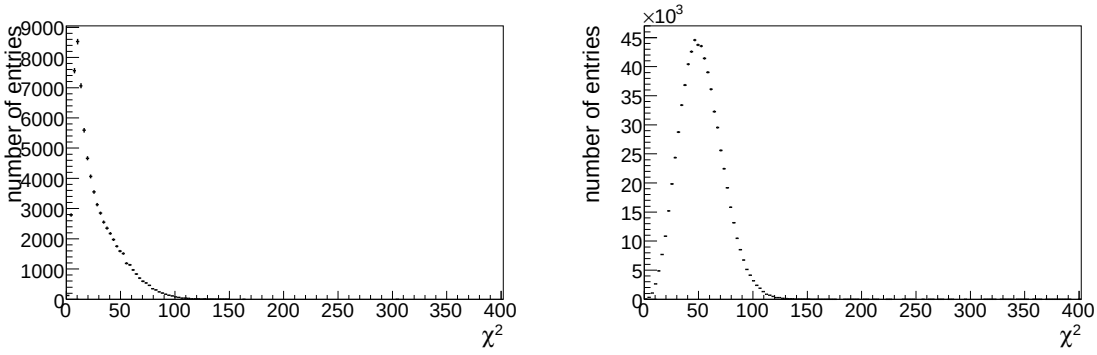


Figure 5.29: *Number of measurements per candidate removed by the outlier removal. (a) is for best candidates, (b) for wrong ones. The code stops after removing six measurements from a candidate to save time.*



(a) **(b)**

Figure 5.30: *χ^2 distribution after the outlier removal took place. The long tails have disappeared for both best (a) and ghost (b) candidates. χ^2 of most ghosts is still higher than that of a typical best candidate.*

already been selected is not greater than some threshold. If the candidate is good enough, its measurements are marked used, and the candidate is converted to a standard `LHCb::Track` object which can be used by the rest of the reconstruction software.

The next two subsections will describe the quality variable and the cut on the used hit fraction in more detail. Then the performance of the track selection will be evaluated on the sample of particles with 6 x layers hit.

Candidate quality estimate

A candidate quality estimate is introduced which combines three variables:

- the χ^2 probability for the fit in xz direction
- the χ^2 probability for the fit in yz direction
- the probability for the observed number of measurements on the candidate (see below for details)

The quality of a candidate is defined as the logarithm of the product of the three probabilities.

Figure 5.31 shows the distribution of the quality variable for the best alternative of each reconstructed particle and ghost candidates is shown. Figures 5.32, 5.33 and 5.34 show the different contributions separately.

The first two items are obvious choices for a quality indicator, and the idea for the third one has already been introduced in 2.4.4.

In this implementation, the third item is handled as follows: A candidate with n tracklets has measurements in $n + 1$ x layers. Assuming that a hypothetical particle travelling along the trajectory defined by the candidate traverses as many x as stereo layers, one expects the candidate to have $N = 4(n + 1)$ measurements, one in each monolayer. In a simple binomial model, the probability to observe only M measurements is given by

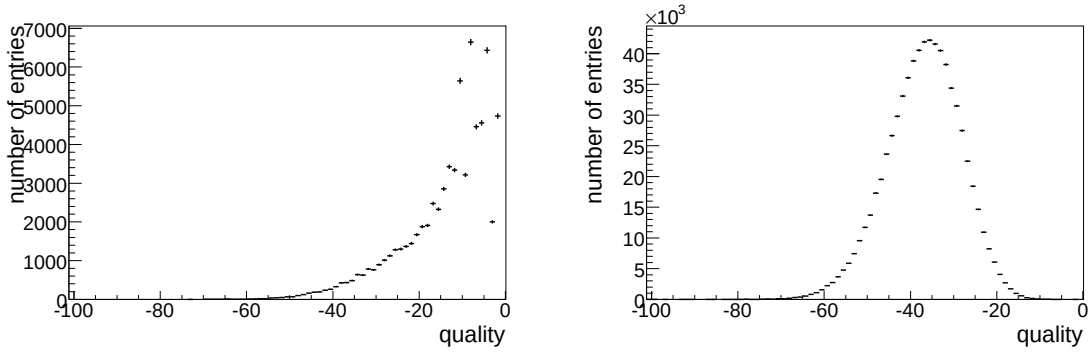
$$\binom{N}{M} p^M (1-p)^{N-M}$$

One needs to use an estimate for the effective monolayer detection efficiency p. For this implementation, this is taken to be 0.95, a little lower than the single cell efficiency of about 0.98 quoted in [24] to account for the insensitive area between straws or modules. The performance of the algorithm has not been found to depend critically on the precise value of 0.95, only when approaching 1.0 does the efficiency go down. This is understandable because, effectively, one prohibits a track to have less than two measurements per layer.

Cut on used measurement fraction for clone and ghost suppression

Candidates are selected in order of decreasing quality. When the fraction of measurements that a candidate shares with candidates selected earlier stays below 0.25, the candidate is kept, it is dropped otherwise. Selected candidates have their measurements flagged as used.

Figure 5.35 shows the distribution of used measurements for best alternatives of reconstructed particles, clones and ghost candidates. Clones and



(a) **(b)**

Figure 5.31: *Track candidate quality variable for best reconstructed alternative for particles (a) and ghost candidates (b).*

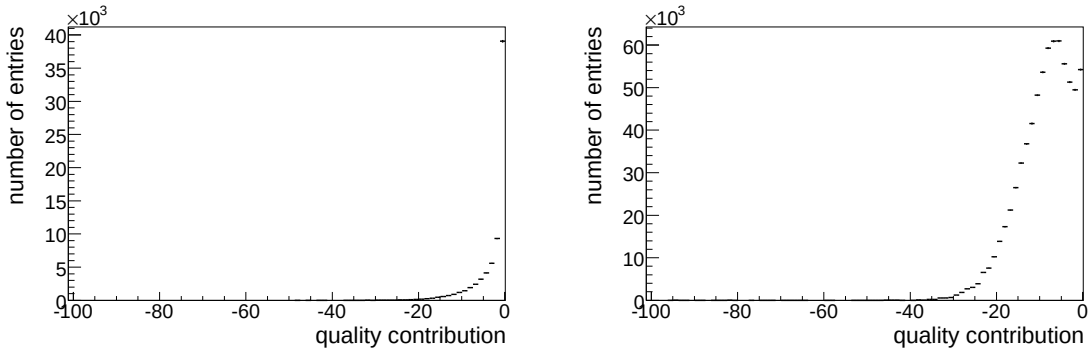


Figure 5.32: *Contribution from χ^2-probability of fit in xz projection to track candidate quality variable for best reconstructed alternative for particles (a) and ghost candidates (b).*

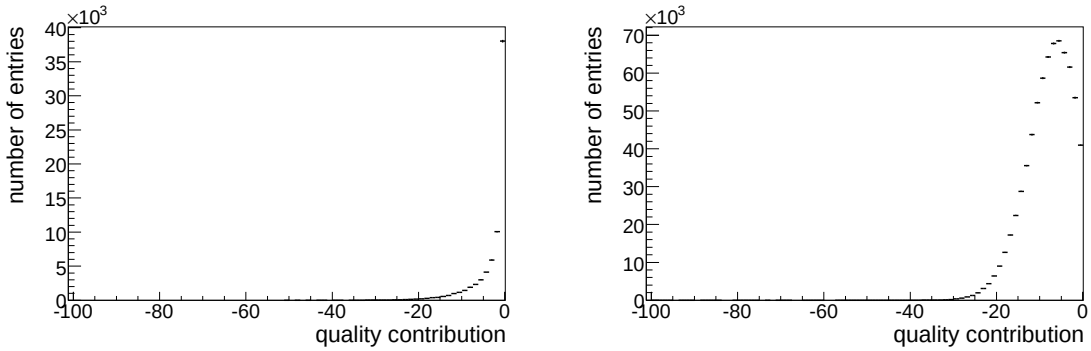


Figure 5.33: *Contribution from χ^2-probability of fit in yz projection to track candidate quality variable for best reconstructed alternative for particles (a) and ghost candidates (b).*

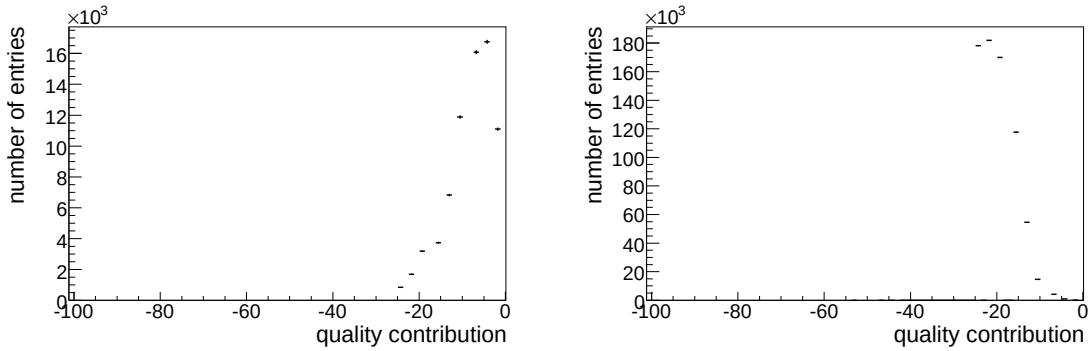


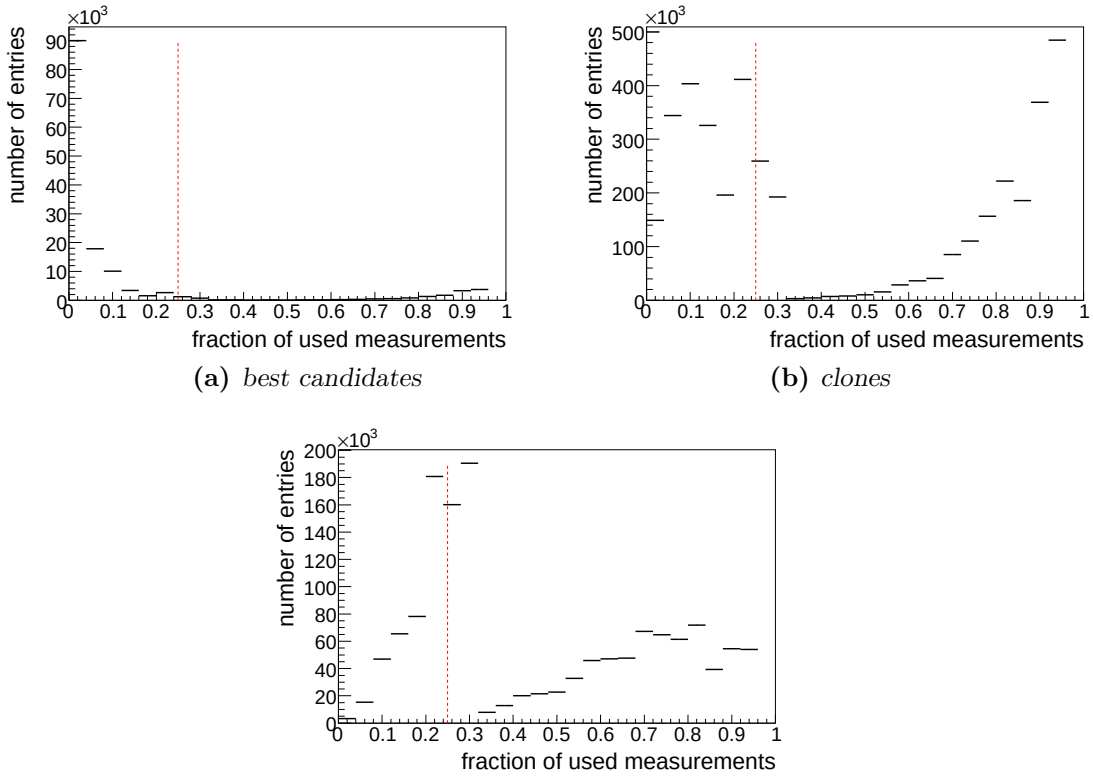
Figure 5.34: *Contribution from missing measurements to track candidate quality variable for best reconstructed alternative for particles (a) and ghost candidates (b). One can see the effect of the cut on the number of missing measurements — there are no entries below quality contributions of about* -25.

ghosts have a many used measurements while the best alternative for a given particle has typically only few.

Performance of the candidate selection stage

On the sample of particles with measurements in all six x layers that was used to evaluate performance of individual stages of the algorithm, the observed efficiency of the algorithm is 88.8 % after the candidate selection phase. The ghost fraction has dropped to 7.2 %, although this figure will go up once shorter candidates are also considered.

The fraction of (MC-)Clones in the sample was reduced from 66.0 % before the candidate selection phase to none. Again, this number will go up slightly once shorter candidates are treated as well.



(a) *best candidates*

(b) *clones*

(c) *ghosts*

Figure 5.35: *Fraction of used measurements of track candidates immediately before their selection. Best reconstructed alternative for particles is shown in (a), clones in (b) and ghost candidates in (c). During the selection process, the used measurement fraction of unselected candidates grows as more candidates are selected. As best tracks have two single measurements per detector layer in most cases, measurements are usually flagged used in pairs, so the discontinuity introduced by the cut is spread out over several bins.*

Chapter 6

Overall performance of the algorithm

The performance of the different stages of the algorithm has been evaluated in the last chapter on a subsample of all particles to facilitate understanding. In this chapter, the overall performance of the algorithm is evaluated using the same data sample that was used in Section 3.2. The algorithm uses tracklet chains of lengths five and four.

The algorithm will be compared to the one discussed in Section 2.4.4 which had its code for IT reconstruction disabled to make sure that both algorithms use OT measurements only. In this chapter, the OT disabled version of this algorithm is referred to as "standard algorithm" for brevity.

As the cellular automaton based algorithm in its current form does not reconstruct particles which give interrupted tracklet chains, the efficiency of both algorithms is compared on two samples. One of them is the subsample used in the last chapter which contains only particles that allow uninterrupted chains to be formed. This sample is referred to as "6X" because the contained particles have of measurements in all six x layers of the OT. The other is the sample of particles which are reconstructible in the OT, defined near the beginning of Section 5.2. This sample is referred to as "ALL" in the text. Figures and plots will be shown for both samples.

6.1 Efficiencies and ghost fractions

On the 6X sample, the event-averaged efficiency of the cellular automaton based algorithm is (93.4 ± 0.2) %. For the ALL sample, (89.8 ± 0.2) % efficiency are obtained. The ghost fraction is at (12.2 ± 0.1) % (for both samples). Figures 6.1 and 6.2 show the behaviour of the algorithm as a function

of momentum and Outer Tracker occupancy.

The efficiency curve is essentially flat with momentum, only for very low momenta in the range below 1.5 GeV does the efficiency drop significantly below 90 %. The ghost fraction initially drops for increasing momenta into a minimum at 8 % around momenta of about 9 GeV, rising to a plateau around 15 % for higher momenta. This rise for high momenta is due to the same effect as the one observed in Section 3.2.3. As can be expected, the ghost fraction rises with increasing detector occupancy from near-zero at very low occupancies to around 60 % at 20 % occupancy.

The standard algorithm shows (88.7 ± 0.2) % efficiency on the ALL sample and (89.9 ± 0.2) % efficiency on the 6X sample. The ghost fraction is at (11.4 ± 0.1) %. Figures 6.3 and 6.4 show the same quantities for the standard algorithm. On the 6X sample, both algorithms have about the same efficiency down to momenta of about 2 GeV. Below, the standard algorithm performs significantly worse. On the ALL sample, the cellular automaton based algorithm is slightly worse for high momenta while it keeps its advantage over the other algorithm in the low momentum region. The effects of the module occupancy cut on the efficiency described in Section 3.2.3 can be seen in the plots of efficiency against OT occupancy.

Table 6.1 summarises the efficiency and ghost fraction figures for both algorithms.

6.2 Purity and collection efficiency

Purity and measurement collection efficiency have been investigated for both algorithms. The tracks of both algorithms were fitted with the Kalman fitter in the LHCb software. The outlier removal procedure in this fitter may remove up to two measurements from a track for both algorithms. Table 6.2 summarises the results, Figures 6.5 and 6.6 show the behaviour of these two quantities with momentum and OT occupancy. As the numbers in Table 6.2 are very similar for both samples and the plots look almost alike as well, only the plots for the ALL sample are shown.

The cellular automaton based algorithm performs consistently a little

algorithm	ε (6X sample)	ε (ALL sample)	ghost fraction
cellular automaton	(93.4 ± 0.2) %	(89.8 ± 0.2) %	(12.2 ± 0.1) %
standard algorithm	(89.9 ± 0.2) %	(88.7 ± 0.2) %	(11.4 ± 0.1) %

Table 6.1: *Summary of efficiency ε and ghost fraction for both algorithms.*

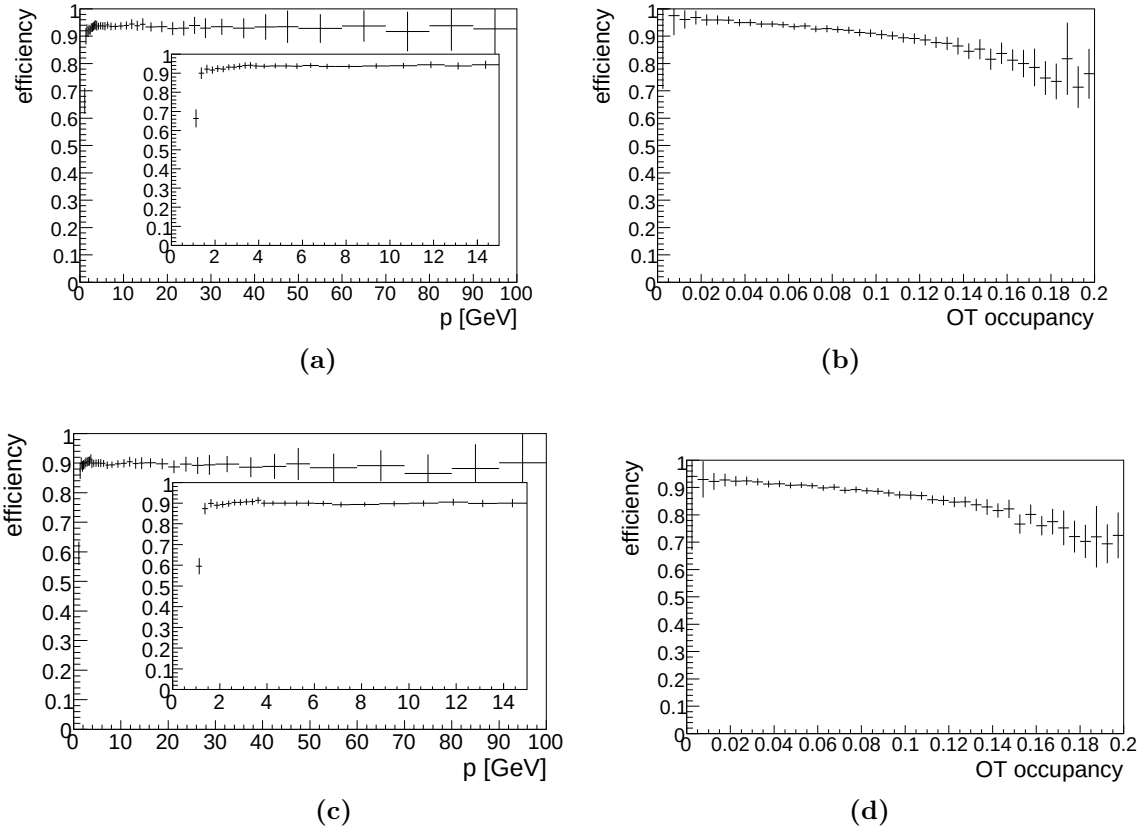


Figure 6.1: *Event averaged efficiency of the cellular automaton based algorithm, versus momentum (left) and versus Outer Tracker Occupancy (right). Top row is for the 6X sample, bottom row is for the ALL sample. On the 6X sample, the algorithm plateaus around 94 % efficiencies for higher momenta, and performance degrades gracefully for high detector occupancies.*

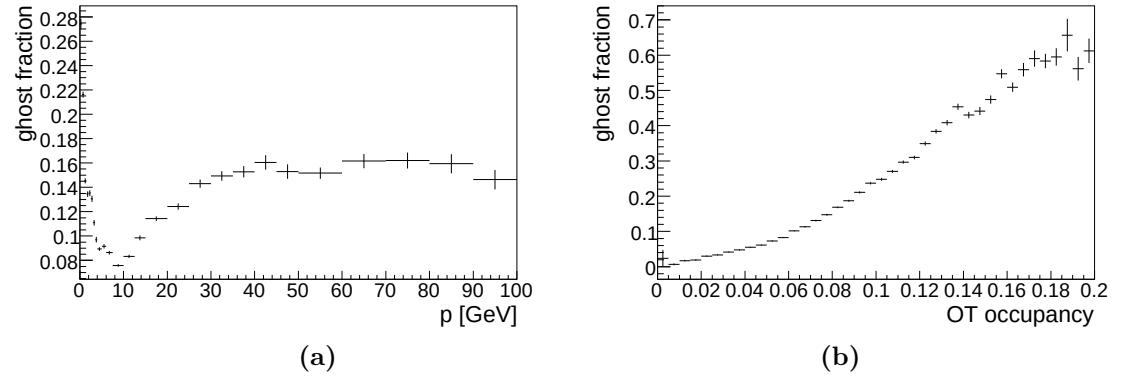


Figure 6.2: *Event averaged ghost fraction for the cellular automaton based reconstruction algorithm versus momentum (a) and versus Outer Tracker occupancy (b). As can be expected, the ghost fraction rises with occupancy.*

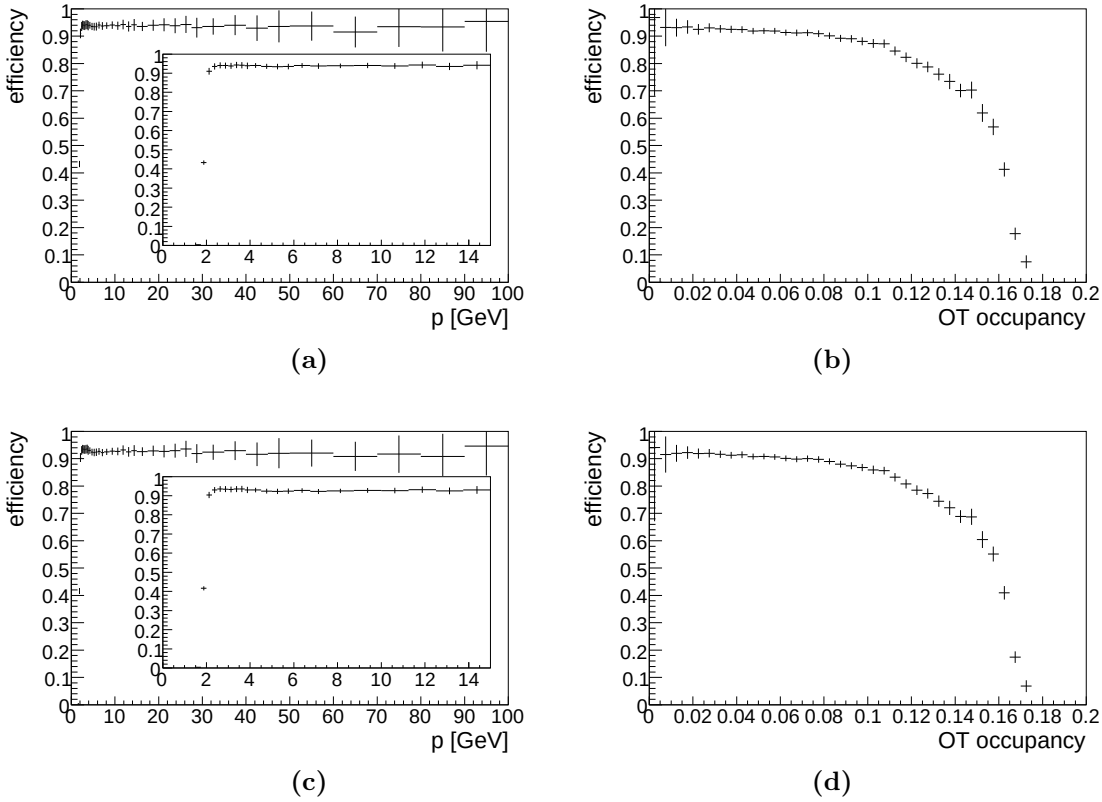


Figure 6.3: *Event averaged efficiency of the standard algorithm reconstructing only the OT. The top row is for the 6X sample, bottom row for the ALL sample. On the ALL sample, efficiency is a little better for higher momenta than for the cellular automaton based algorithm, however, for momenta below 2 GeV this algorithm performs worse on both samples.*

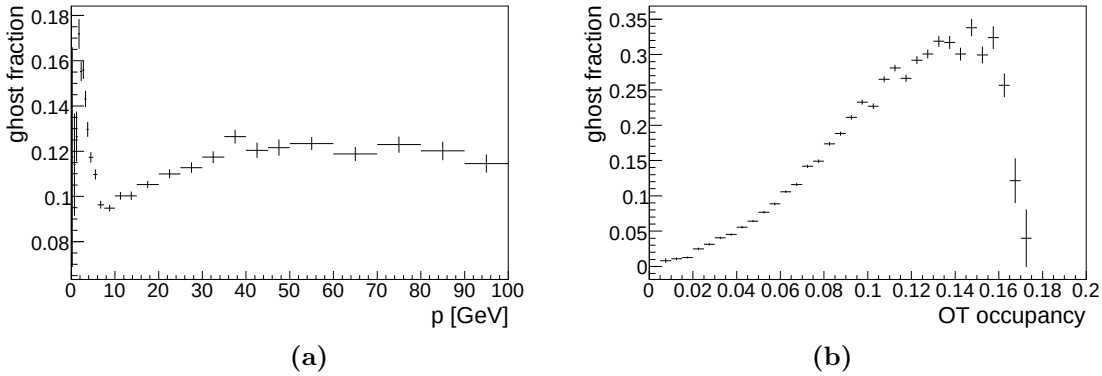


Figure 6.4: *Event averaged ghost fraction for the standard algorithm. The ghost fraction is shown versus momentum in (a), and versus OT occupancy in (b). The drop at high occupancies is due to the cuts described in 3.2.3.*

better than the standard one. The purity is relatively stable with respect to momentum and detector occupancy for both algorithms. Collection efficiencies for both algorithms go down a little for low momentum particles, but remain stable for high momentum ones. For the standard algorithm, the collection efficiency goes down faster with occupancy than for the cellular automaton based ones. This is probably due to the cut on the module occupancies mentioned in Section 3.2.3.

6.3 Execution time behaviour

In this section, the execution time of the algorithm is investigated as a function of the number of measurements in the detector. This number includes everything: noise, spillover, crosstalk and the measurements from the particles one wants to observe. The machine used to obtain these timing measurements is running Scientific Linux SLC3 and uses an AMD Opteron CPU with a core clock of 2.4 GHz.

To reconstruct the 15000 events in the sample examined, the cellular automaton based algorithm took 3941 seconds, while the standard algorithm only needed 2014 seconds. This is equivalent to 259 ms and, respectively, 134 ms per event on average. The cellular automaton based algorithm is about a factor of two slower, but the code has not been optimised for speed yet.

Figure 6.7 shows the behaviour of the total execution time with the number of measurements in the detector. For the cellular automaton based algorithm, one can see that its execution time goes up sharply above 10000 measurements in the detector[1]. The standard algorithm is more well behaved there because of the cuts on module occupancy. It also does not reconstruct events if IT and OT together contain more than 10000 measurements. This happened in 83 of 15000 events.

[1]The Outer Tracker has 53760 channels.

	standard algorithm	cellular automaton
purity (6X sample)	98.0 %	98.5 %
purity (ALL sample)	97.9 %	98.4 %
ε_{coll} (6X sample)	81.2 %	88.7 %
ε_{coll} (ALL sample)	81.0 %	88.3 %

Table 6.2: *Comparison of standard algorithm and cellular automaton-based approach in terms of purity and measurement collection efficiency ε_{coll}.*

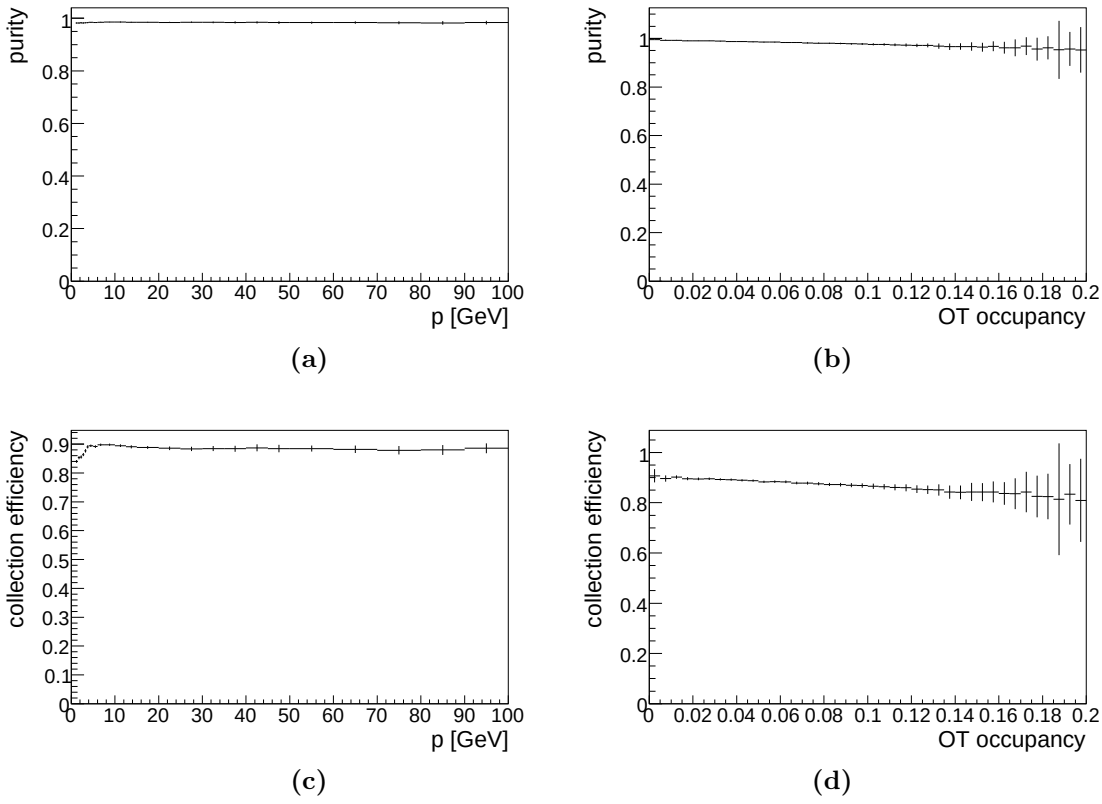
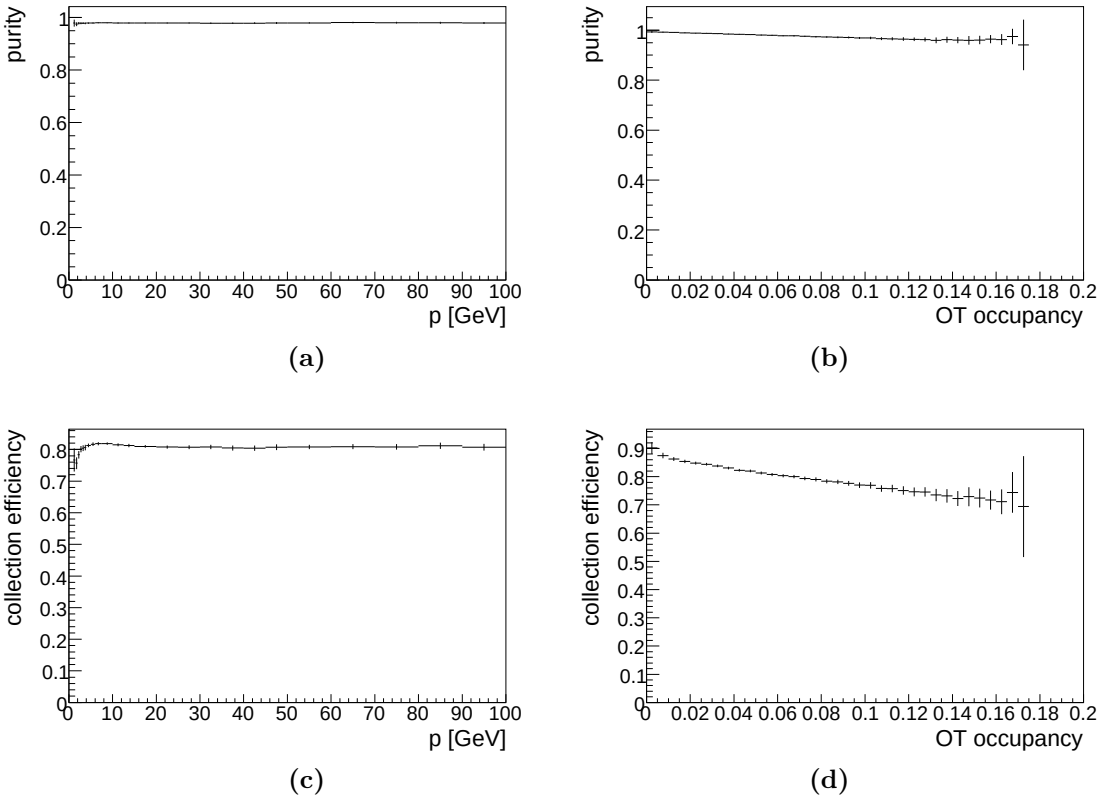


Figure 6.5: *Purity (top row) and measurement collection efficiency (bottom row) for the cellular automaton based algorithm on the ALL sample. On the left, the plots versus momentum are shown, those on the right are versus OT occupancy. Purity is stable with respect to momentum and occupancy. The collection efficiency worsens slightly for low momenta and high occupancies.*



(a) (b)

(c) (d)

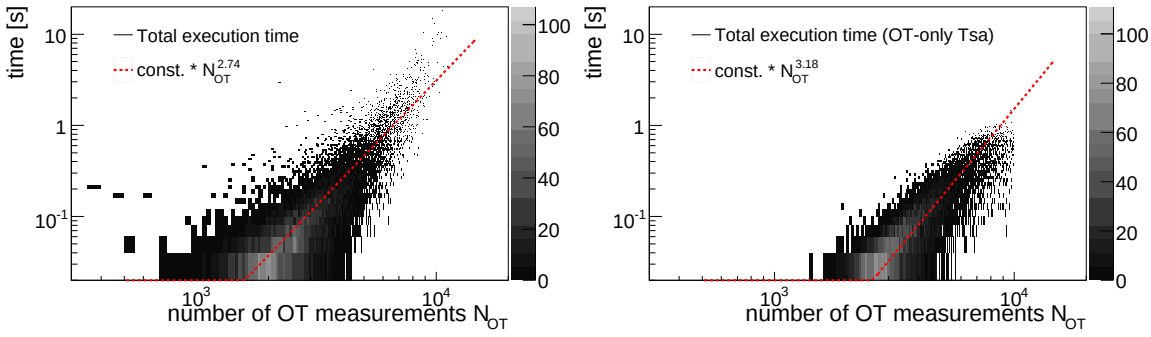
Figure 6.6: *Purity (top row) and measurement collection efficiency (bottom row) for the standard algorithm on the ALL sample. On the left, the plots versus momentum are shown, those on the right are versus OT occupancy. Purity is stable with respect to momentum and occupancy. The collection efficiency worsens slightly for low momenta and high occupancies, but in a more pronounced way than for the cellular automaton.*

The function $const. \cdot N_{OT}^{\alpha}$ was used to fit the data; N_{OT} denotes the number of measurements, and the constant factor in front of the expression and the exponent α were determined to get an estimate of the asymptotic behaviour of the algorithm. However, the spread of the data is large for high occupancies, and there is not much statistics in that area. Therefore, the results of the fit have to be interpreted with care and should not be relied on too heavily. The plots have logarithm scales on both axes because the exponent α of N_{OT} then shows up as the slope of a line approximating the distribution.

The cellular automaton based algorithm spends most of its time in the tracklet following procedure. Table 6.3 gives the average time per event spent in the individual stages.

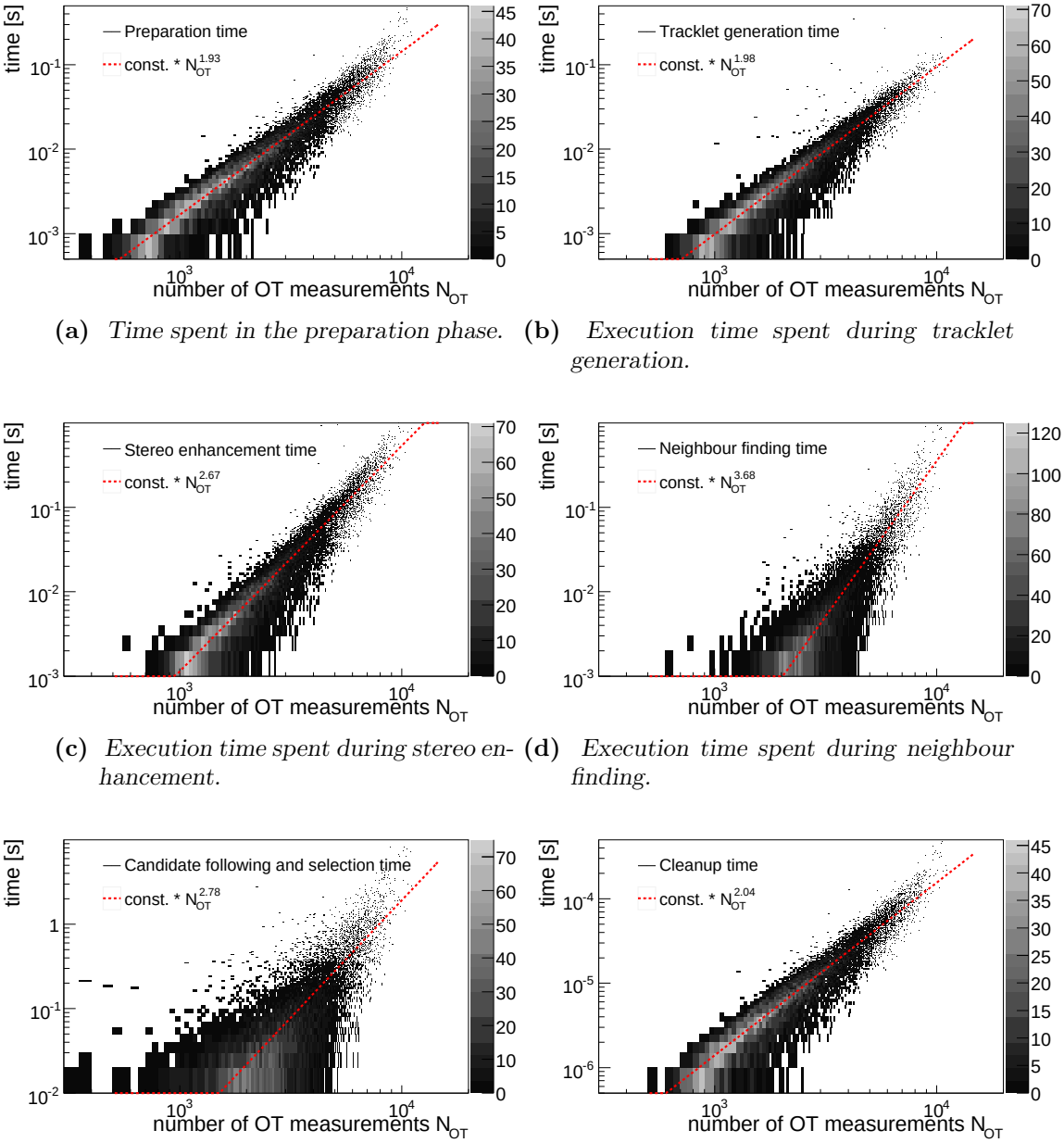
For completeness, Figures 6.8a, 6.8b, 6.8c, 6.8d, 6.8e, 6.8f show execution time spent in the individual stages versus the number of OT measurements.

While the behaviour of the tracklet generation and stereo enhancement stages is roughly what is to be expected from the loops that the algorithm performs, especially the timing behaviour of the tracklet following stage should be improved. This could be done by trying to exclude wrong possibilities earlier in the algorithm. For example, it may be possible to rule out more wrong combinations of tracklets during the neighbour finding stage by performing a fit early in the process. Also, some improvements seem to be possible in the stereo enhancement stage: At the moment, the algorithm will create pseudo-x clusters over the whole detector, even if there is no tracklet nearby to match it to.



(a) **(b)**

Figure 6.7: *Total execution time of both algorithm versus number of OT measurements. The cellular automaton based algorithm is shown in (a), (b) shows the standard algorithm. The algorithm shown in (b) does not process events with $N_{OT} > 10000$.*



(a) *Time spent in the preparation phase.* **(b)** *Execution time spent during tracklet generation.*

(c) *Execution time spent during stereo enhancement.* **(d)** *Execution time spent during neighbour finding.*

(e) *Execution time spent forming and selecting candidates.* **(f)** *Execution time spent during cleanup.*

Figure 6.8: *Execution time spent in the different phases of the algorithm versus number of OT measurements.*

stage	average time [ms]
preparation	6.9
tracklet generation	12.8
stereo enhancement	43.3
neighbour finding, automaton evolution	15.3
tracklet following, candidate selection	160.2
cleanup	20.6

Table 6.3: *Average time spent in the individual stages of the cellular automaton based algorithm.*

One should also think about creating proper three-dimensional tracklets instead of adding the information to the tracklets in xz projection. That way, quite a bit of complexity in the neighbour finding and tracklet following stages could be eliminated, resulting in potentially faster code.

Unfortunately, time was running out when the stereo path of the algorithm was implemented, therefore it was kept simple.

It may also be possible to perform tighter cuts once the algorithm can handle interrupted tracklet chains[2]. This approach seems to have great potential with respect to timing behaviour, especially in the light of the loose pitch residual cuts applied. Once the algorithm can afford to lose a tracklet occasionally, the cuts can be tightened. This would reduce background significantly and, more importantly, early in the algorithm instead of deferring the issue, leaving its solution to the tracklet following stage which is combinatorically expensive in presence of a lot of surviving background.

[2]These could be dealt with by extrapolating a tracklet to the next layer if no neighbour could be found, thus creating an "anchor" for tracklet generation in a layer lacking the expected starting measurement.

Chapter 7

Summary

In the first part of this thesis, an overview of the existing tracking algorithms of the LHCb experiment has been given. A validation tool has been implemented which was used to study their performance. In the second part, a new standalone reconstruction algorithm for the LHCb Outer Tracker was implemented which is based on a cellular automaton. The algorithm and its performance have been discussed in detail.

This algorithm reconstructs (93.4 ± 0.2) % of particles with measurements in all six x layers of the Outer Tracker and (89.9 ± 0.2) % of particles which are reconstructible in the OT. (12.2 ± 0.1) % of all reconstructed tracks can not be associated with any particle in the event and are thus considered as ghost tracks. The algorithm is comparable to existing algorithms in terms of efficiency, ghost fraction, purity and measurement collection efficiency. In the momentum region below 2 GeV, the algorithm yields higher efficiency than existing implementations. Additionally, the measurements available in the detector are collected more efficiently over the whole momentum range. The algorithm based on cellular automaton principles has proven to be more robust with respect to high occupancies which will be crucial both for higher luminosity and potentially higher noise levels.

As the code is not yet optimised for speed, it is still about a factor of two slower than the standard LHCb reconstruction algorithm for the Outer Tracker.

Summarising, the cellular automaton has proven to be a versatile tool which can be used to write an efficient and fast implementation of a pattern recognition algorithm in a short time. While the performance of the algorithm presented in this thesis is already good, there is potential to improve efficiency and speed, and to reduce the ghost fraction.

Bibliography

[1] *LHCb: Technical Proposal*. Tech. Proposal. CERN, Geneva, 1998.

[2] LHCb Collaboration, R. Antunes-Nobrega et al. *LHCb reoptimized detector design and performance Technical Design Report*. Technical Design Report LHCb. CERN, Geneva, 2003.

[3] LHCb Collaboration, P. R. Barbosa-Marinho et al. *LHCb VELO (VErtex LOcator) Technical Design Report*. Technical Design Report LHCb. CERN, Geneva, 2001.

[4] LHCb Collaboration, P. R. Barbosa-Marinho et al. *LHCb Inner Tracker Technical Design Report*. Technical Design Report LHCb. CERN, Geneva, 2002. revised version number 1 submitted on 2002-11-13 14:14:34.

[5] LHCb Collaboration, P. R. Barbosa-Marinho et al. *LHCb Outer Tracker Technical Design Report*. Technical Design Report LHCb. CERN, Geneva, 2001.

[6] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, D82:35–45, 1960.

[7] Pierre Billoir. Track fitting with multiple scattering: A new method. *Nucl. Instr. Meth.*, A225:352, 1984.

[8] R. Fruhwirth. Application of kalman filtering to track and vertex fitting. *Nucl. Instrum. Meth.*, A262:444–450, 1987.

[9] O. Callot and S. Hansmann-Menzemer. The forward tracking algorithm and performance studies. Technical Report LHCb-2007-015. CERN-LHCb-2007-015, CERN, Geneva, May 2007.

[10] R. W. Forty and M. Needham. Standalone track reconstruction in the t-stations. Technical Report LHCb-2007-022. CERN-LHCb-2007-022, CERN, Geneva, Mar 2007.

[11] E. Rodrigues. Dealing with clones in the tracking. Technical Report LHCb-2006-057. CERN-LHCb-2006-057, CERN, Geneva, Nov 2006.

[12] D. Hutchcroft. Velo pattern recognition. Technical Report LHCb-2007-013. CERN-LHCb-2007-013, CERN, Geneva, Mar 2007.

[13] J. Van Tilburg and M. Merk. *Track simulation and reconstruction in LHCb. oai:cds.cern.ch:CERN-THESIS-2005-040*. PhD thesis, Vrije Univ. Amsterdam, Amsterdam, 2005. Presented on 01 Sep 2005.

[14] M. Needham and J. Van Tilburg. Performance of the track matching. Technical Report LHCb-2007-020. CERN-LHCb-2007-020, CERN, Geneva, Mar 2007.

[15] E. Rodrigues. Tracking definitions. Technical Report LHCb-2007-006. CERN-LHCb-2007-006, CERN, Geneva, Feb 2007. revised version submitted on 2007-03-28 09:34:37.

[16] R. Brun and F. Rademakers. Root — an object oriented data analysis framework. *Nucl. Instr. and Meth.*, A389:81–86, 1997. See also http://root.cern.ch/.

[17] J. v. Neumann. *The Theory of Self-reproducing Automata*. Univ. of Illinois Press, Urbana, IL, 1966. based on work in the late 1940s, but published posthumously.

[18] M. Gardner. The fantastic combinations of john conway's new solitaire game "life". *Scientific American*, 223: 120–123, 1970.

[19] A. Glazov, I. Kisel, E. Konotopskaya, and G. Ososkov. Filtering tracks in discrete detectors using a cellular automaton. *Nucl. Instr. and Meth.*, A329:262–268, 1993.

[20] I. Kisel, V. Kovalenko, F. Laplanche, and others (NEMO Collaboration). Cellular automaton and elastic net for event reconstruction in the nemo-2 experiment. Technical report, 1997.

[21] I. Abt, D. Emeliyanov, I. Kisel, and S. Masciocchi. Cats: a cellular automaton for tracking in silicon for the hera-b vertex detector. *Nucl. Instr. and Meth.*, A489:389–405, 2002.

[22] I. Abt, D. Emeliyanov, I. Gorbounov, and I. Kisel. Cellular automaton and kalman filter based track search in the hera-b pattern tracker. Technical report, 2002.

[23] D. Emeliyanov and I. Kisel. Cats track fitting algorithm based on the discrete kalman filter. Technical Report HERA-B note 00-032, 2000.

[24] G. W. van Apeldoorn, S. Bachmann, T. H. Bauer, E. Bos, Yu Guz, T. Haas, J. Knopf, J. Nardulli, T. Ketel, A. Pellegrino, T. Sluijk, N. Tuning, U. Uwer, P. Vankov, and D. Wiedner. Beam tests of final modules and electronics of the lhcb outer tracker in 2005. Technical Report LHCb-2005-076. CERN-LHCb-2005-076, CERN, Geneva, Oct 2005.

Danksagung

Viele Menschen haben zum Gelingen dieser Arbeit in vielfältiger Weise mit beigetragen, denen ich an dieser Stelle danken möchte.

Mein besonderer Dank gilt Herrn Professor Ulrich Uwer, der mir dir Arbeit an diesem Thema ermöglicht und mir stets mit Rat und Tat zur Seite stand. In gleicher Weise möchte ich mich bei Stephanie Hansmann-Menzemer bedanken, die in sich in genauso aufopfernder Weise um mich gekümmert hat und der ich vermutlich noch öfter auf die Nerven gefallen bin.

Ich bin Johannes Albrecht, Johan Blouw, Stephanie Hansmann-Menzemer, Jens Kessler, Markus Moch, Gunther Schiller, Rainer Schwemmer und Ulrich Uwer dankber für ihre kritischen Anmerkungen.

Ich möchte mich ebenfalls bei der gesamten HE-Gruppe für die schöne Zeit und nette Atmosphäre bedanken, in der ich arbeiten durfte.

Mein besonderer Dank geht an Tanja Haas, die mich während meiner Miniforschung aushalten musste und später dann wieder. In gleicher Weise danke ich meinen (teilweise kurzzeitigen) Zimmernachbarn Johnnes Albrecht, Iuri Bagaturia, Marc Deissenroth, Jens Kessler, Tanja Haas, Matthias Mozer, Rainer Schwemmer, Christoph Werner und Roger Wolf, denen ich sicher gelegentlich auf die Nerven gefallen bin.

Ich danke Ivan Kisel für seine inspirierende Vorlesung “Pattern Recognition in High Energy Physics” im Sommersemester 2005 und für die hilfreichen Diskussionen und Vorschläge später.

Ich danke der gesamten LHCb-Kollaboration für die freundlich Aufnahme, die ein Neuankömmling dort erfährt.

Ich entschuldige mich bei allen, denen ich zu danken vergessen habe, mein Dank sei ihnen hiermit “nachgericht”.

Zu guter Letzt möchte ich mich bei meiner Familie bedanken, die mir nicht nur das Studium und diese Diplomarbeit ermöglicht hat.

Erklärung

Ich versichere, dass ich diese Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 4. Juli 2007 _____