Development of a Hough Transformation Track Finder for Time Projection Chambers

Dissertation

zur Erlangung des Doktorgrades
des Fachbereichs Physik
der Universität Hamburg

vorgelegt von

Isa Heinze

aus Rahden

Hamburg
2013

Gutachter/in der Dissertation:	Prof. Dr. Erika Garutti Dr. Philip Bechtle
Gutachter/in der Disputation:	Prof. Dr. Erika Garutti Prof. Dr. Johannes Haller
Datum der Disputation:	05.12.2013
Vorsitzender des Prüfungsausschusses:	Dr. Georg Steinbrück
Vorsitzender des Promotionsausschusses:	Prof. Dr. Peter Hauschildt
Dekan der MIN-Fakultät:	Prof. Dr. Heinrich Graener
Leiterin des Department Physik:	Prof. Dr. Daniela Pfannkuche

Zusammenfassung

Der International Linear Collider (ILC) ist ein geplante Teilchenphysikexperiment. Eines der beiden Detektorkonzepte ist das International Large Detector (ILD) Konzept für welches eine Zeitprojektionskammer als Hauptspurdetektor vorgesehen ist. Im ILD wird das Particle Flow Konzept verfolgt, was zu besonderen Anforderungen an den Detektor führt. Insbesondere ist für das Spursystem eine sehr gute Impulsauflösung nötig.
Verschiedene Prototypen wurden gebaut um zu zeigen, das es möglich ist eine Zeitprojektionskammer zu bauen, die die Anforderungen für eine Kammer im ILD erfüllt. Einer der Prototype ist der Large Prototype mit dem Auslesetechnologien getestet werden, die derzeit entwickelt werden. Parallel dazu wird Rekonstruktionssoftware entwickelt mit der die gemessenen Daten rekonstruiert werden können.
In dieser Arbeit ist die Entwicklung eines Spurfindealgorithmus, basierend auf der Hough Transformation, beschrieben. Es kann sowohl gekrümmte Spuren (mit Magnetfeld) als auch gerade Spuren (ohne Magnetfeld) finden. Dieses Paket wurde in erster Linie für Large Prototype Teststrahldaten geschrieben. Es wurde aber auch auf einer Monte Carlo Simulation von Spuren in der ILD Zeitprojektionskammer getested.
Desweiteren wird die Analyse von Teststrahldaten in Hinblick auf die erreichtbare Einzelpunktauflösung präsentiert. Die Daten wurden mit dem Large Prototype und einem Auslesemodul mit Gas Elektron Multiplier Verstärkung genommen wurden. Für die Rekonstruktion dieser Daten wurde das oben erwähnte Softwarepacket verwendet. Die Einzelpunktauflösung steht direkt in Verbindung mit der Impulsauflösung des Detektors. Daher ist eine gute Punktauflösung notwendig um eine gute Impulsauflösung zu erzielen.

Abstract

The International Linear Collider (ILC) is a planned particle physics experiment. One of the two detector concepts is the International Large Detector (ILD) concept for which a time projection chamber is foreseen as the main tracking device. In the ILD the particle flow concept is followed which leads to special requirements for the detector. Especially for the tracking system a very good momentum resolution is required.
Several prototypes were build to prove that it is possible to build a TPC which fulfills the requirements for a TPC in the ILD. One is the Large Prototype with which different readout technologies currently under development are tested. In parallel reconstruction software is developed for the reconstruction of Large Prototype data.
In this thesis the development of a track finding algorithm based on the Hough transformation is described. It can find curved tracks (with magnetic field) as well as straight tracks (without magnetic field). This package was mainly developed for Large Prototype testbeam data but was also tested on Monte Carlo simulation of tracks in the ILD TPC.
Furthermore the analysis of testbeam data regarding the single point resolution is presented. The data were taken with the Large Prototype and a readout module with GEM (gas electron multiplier) amplification. For the reconstruction of these data the software package mentioned above was used. The single point resolution is directly related to the momentum resolution of the detector, thus a good single point resolution is needed to achieve a good momentum resolution.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The International Linear Collider (ILC) is a planned accelerator which will collide electrons and positrons at a center of mass energy of up to 1 TeV. With this machine precise measurements of the Higgs boson, which was discovered at the LHC recently, and physics beyond the standard model of elementary particle physics are possible. One of the most important measurements planned is the measurement of the Higgs recoil mass.
Two detectors are foreseen for the ILC. One of them is the International Large Detector (ILD) in which the particle flow concept is followed. This introduces special requirements for the tracking system. Efficient tracking and precise measurement of the momenta of charged particles is needed. A time projection chamber (TPC) is foreseen as main tracking device which provides many measurement points (continuous tracking) and thus delivers a good momentum resolution which is essential for measurements such as the Higgs recoil mass. Another advantage of a time projection chamber is its low material budget.
In order to develop a time projection chamber for the ILD several prototypes were build. One of them is the Large Prototype which is located at one of the testbeam areas at DESY. It can hold up to seven readout modules. With this prototype readout technologies can be tested which are developed for the future ILD TPC. Additionally software to reconstruct the data taken with the Large Prototype is developed.
For this thesis a track finding algorithm for Large Prototype data based on a Hough transformation was developed. Tracks need to be found in testbeam data reliably because they are needed for important measurements such as the single point resolution and the momentum resolution. Compared to other track finding algorithms, like the Kalman Filter, the Hough transformation has some advantages particularly important for testbeam data.

- In testbeam data the readout technology is still under development and measurements might not look as expected and need to be cut out from further reconstruction and analysis. In the worst case parts of the readout modules break during data taking. Both cases lead to missing measurements, even several neighboring measurements in a row. Local methods like the Kalman Filter add measurements to the track step by step. If several measurements are missing this method does not work reliably, especially if several neighboring measurements are missing. The Hough transformation is a global method. All measurements are used at the same time and in the same way. For this reason the Hough transformation can cope with missing measurements.

- Local track finding algorithms need to predict where the next measurement is expected. Therefore they need information on the readout geometry. The Hough transformation as a global methods needs the measurements only. This is particularly interesting for testbeam data. Data taken with readout structure of any shape can be used without any extra work. The Hough transformation as it was implemented can even be used for different prototypes without applying any changes.
- The Hough transformation can find any track shape describable by one set of parameters. It thus works for straight tracks and curved tracks. This feature is an advantage because testbeam data might be taken with and without magnetic field.

All these features make the Hough transformation particularly suitable for track finding in testbeam data.
This thesis starts with a general overview over the standard model of elementary particle physics in Chapter 2 and a description of the International Linear Collider and the International Large Detector concept in Chapter 3. In Chapter 4 an introduction to time projection chambers is given and in Chapter 5 the reconstruction algorithms and the reconstruction software are presented. Chapter 6 gives an overview over the most important track finding methods available. The Hough transformation was implemented in a software package called Pathfinder. This software package is presented in Chapter 7. In this chapter also first systematic tests on the performance of the Hough transformation are presented. For more detailed tests Pathfinder was used to find tracks in simulated data in the ILD detector and the results were compared to those obtained by another track finding algorithm based on a clustering algorithm and a Kalman Filter. This algorithm was developed specifically for the reconstruction of tracks in the ILD. These studies are presented in Chapter 8. A measurement of the single point resolution in testbeam data taken with the Large Prototype is finally presented in Chapter 9. For the track reconstruction the Hough transformation was used. As summary and an outlook are given in Chapter 10.

Chapter 2

Particle Physics

2.1 The Standard Model of Elementary Particle Physics

The standard model of elementary particle physics [1, 2, 3] describes all known elementary particles and the interactions (excluding gravitation) between them. There are two kinds of particles within this model: fermions (half-integer spin particles) and bosons (integer spin particles).
The most important properties of the fermions are summarized in Table 2.1. The fermions can be divided into two groups: Quarks having an electric charge of either -1/3 or 2/3 and leptons having an electric charge of -1 or 0. The difference between quarks and leptons is that quarks carry color charge while the leptons are color neutral. The fermions can be divided into three generations. Each generation contains two quarks (one with an electric charge -1/3 and one with an electric charge of 2/3) one charged and one neutral lepton. The first generation contains the lightest, the third generation the heaviest particles. Furthermore each particle has a partner with opposite charges called antiparticle.
Particles carrying color charge cannot exist freely, thus quarks build hadrons (either mesons consisting of quark and antiquark of baryons consisting of three quarks or three antiquarks). The only exception is the top quark which decays before it can hadronize.

	1st Generation	**2nd Generation**	**3rd Generation**	**Charge**
Quarks	up 2.3 MeV	charm 1.275 GeV	top 173.5 GeV	2/3
	down 4.8 MeV	strange 95 MeV	bottom 4.65 GeV	-1/3
Leptons	electron 0.511 MeV	muon 105.658 MeV	tau 1776.82 MeV	-1
	e-Neutrino < 2 eV	μ-Neutrino < 0.19 MeV	τ-Neutrino < 18.2 MeV	0

Table 2.1: Fermions in the standard model of elementary particle physics and some of their properties [4].

Boson	**Mass**	**Charge**	**Interaction**
photon	0	0	electromagnetic
Z	91.1876 GeV	0	weak
$W^{\pm}$	80.385 GeV	±1	weak
gluon	0	0	strong
Higgs	126.0 GeV [5]	0	
	125.3 GeV [6]	0	

Table 2.2: Bosons in the standard model of elementary particle physics and some of their properties [4].

The world around us is built from first generation particles only. Protons and neutrons consist of two up quarks and one down quark and two down quarks and one up quark, respectively. Protons and neutrons build nuclei which then, together with electrons build atoms. To create particles from other generations a collider is needed. A summary of some properties of the bosons, the second big group of elementary particles, is given in Table 2.2. The photon, the $W^{\pm}$ and Z boson and the gluon mediate the interactions between the fermions. The electromagnetic interaction acts on electrically charged particles only and is mediated by the photon. The W and Z bosons mediate the weak interaction. Every particle participates in this interaction. The electromagnetic and the weak interaction are unified to the electroweak interaction at energies of about 100 GeV. Lastly the strong interaction is mediated by gluons. It only acts on particles carrying color charge (quarks and gluons). The fourth fundamental interaction is gravitation, which is not included in the standard model of elementary particle physics. A boson mediating gravitation (graviton) has not been discovered up to now.
The standard model of elementary particle physics predicts massless bosons, however for the W and Z bosons a mass was observed. In the theory a mass can be introduced by an electroweak symmetry breaking which is described by the Higgs mechanism [7,8,9]. This introduces a Higgs field and the interaction of the fermions with the field generate their masses. The Higgs mechanism introduces an additional boson: the so called Higgs boson. A boson being very similar to the Higgs boson was discovered recently [5, 6] at the Large Hadron Collider (LHC [10]) at CERN. Evidence for this particle was also found at the Tevatron [11] at Fermilab.

2.2 Unanswered Questions

The standard model of elementary particle physics describes the results of experiments very well. However, there are observations which cannot be explained by this model.

- Only a small fraction of the matter in the universe consists of standard model particles. The rest consists of unknown constituents called dark matter [12] and dark energy [13, 14].

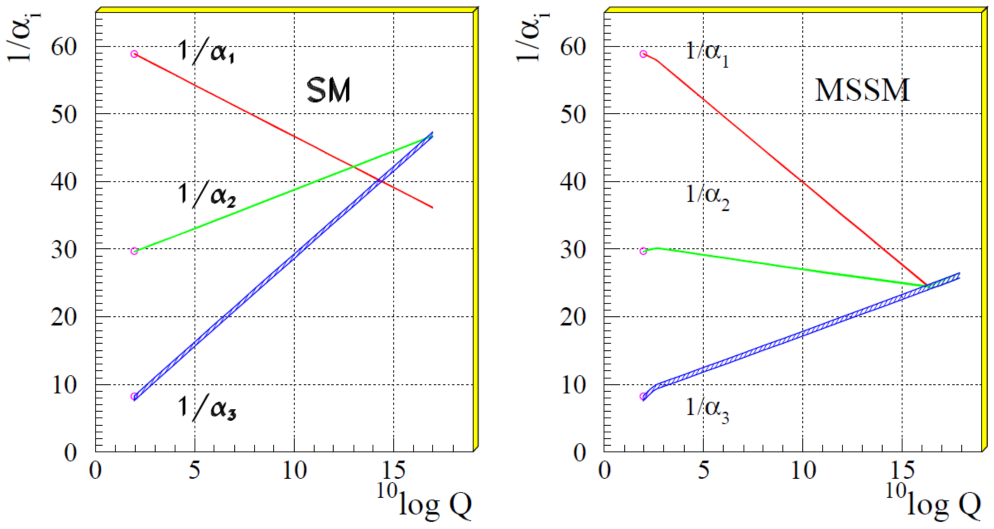


Figure 2.1: The two graphs show the development of the coupling constants for the electromagnetic, weak and strong interaction with energy for the standard model (left) and the MSSM (right). In the MSSM the coupling constants do unify at large energies whiled they do not in the standard model. Picture taken from [21].

- The Grand Unification Theories (GUT [15, 16]) predict that the coupling constants have the same values at very high energies. The coupling constants have been measured for low energies [17]. An extrapolation to large energies does not lead to a unification (see Figure 2.1).
- The universe consists of more matter than antimatter. This asymmetry is not predicted by the standard model of elementary particle physics.

To explain the observations the standard model of elementary particle physics needs to be extended. One of the most popular theories is supersymmetry [18,19,20] which can explain some of the observations listed above. In this theory a new symmetry between bosons and fermions is introduced. Each boson gets a fermionic partner and each fermion gets a bosonic partner. Supersymmetry would lead to a unification of the couplings at large energies as shown in Figure 2.1 for the minimal supersymmetric standard model (MSSM [4]). Up to now no evidence of supersymmetry was found. The symmetry (if realized in nature) must be broken which means that the masses of the supersymmetric partners must be larger than those of the particles.
The LHC is a suitable machine to discover new physics explaining the open questions mentioned above. It can reach mass regions which were never reached before. However, since hadrons are brought to collision precision measurements are difficult. Hadrons are composite particles and thus the QCD background is high and the energy of the interacting particles (quarks and gluons) is not known exactly. In order to perform precise measurements a lepton collider is needed. In such a collider elementary particles are brought to collision which leads to much cleaner events. Furthermore the initial state is well known. In Chapter 3 a possible future lepton collider will be presented.

Chapter 3

The International Linear Collider

It is planned that the next big high energy physics experiment will be a lepton collider. Leptons are elementary particles. This fact leads to cleaner events than at hadron colliders. Furthermore the initial state is known precisely in lepton colliders because the energy of the interacting particles is known precisely. Electron-positron colliders were built before, the largest one was the Large Electron-Positron Collider (LEP [22]). Building a new LEP with the same radius but higher center of mass energy is not feasible. The energy losses due to synchrotron radiation are too large. The energy a particle loses during one turn is given by

$$\Delta E = \frac{(Ze)^2\beta^3\gamma^4}{\epsilon_0 3R} \propto \frac{E^4}{m_0^4 R} \tag{3.1}$$

where $\gamma = E/(m_0c^2)$ and $\beta = v/c \approx 1$. Ze represents the particles charge, for electrons and protons Z is 1, e is the elementary charge and m_0 is the mass of the particle at rest. R is the radius of the storage ring. The amount of energy a particle loses per turn is proportional to the fourth power of the center of mass energy. To reduce the losses of energy one can either use heavier particles (for this reason LHC is colliding protons in the former LEP tunnel) or go to larger radii. In Table 3.1 a few possible accelerator setups are listed. In the first two lines setups similar to LEP and LEP2 are given, already at LEP2 an electron lost about 3% of its initial energy in each turn. The third line shows the same values for LHC. Since the used particles are heavier the losses of energy are significantly smaller. The last two lines represent hypothetical storage rings using electrons at energies higher than those used at LEP (energies which are foreseen for the ILC). In this case a particle would lose a significant amount of its initial energy in one turn. One would have to build a storage ring with radii of several tens (or even hundreds) of kilometers to reduce the energy losses to the level of those at LEP2.
A storage ring for the next lepton collider is thus not a reasonable option. For this reason the next electron-positron collider will be a linear collider where synchrotron radiation does not play any role. The disadvantage, however, is that each beam can be used only once.

E_{cm} in GeV	Particle	Radius in m	ΔE_{turn} in GeV
91 (LEP)	electron	3096	0.122
208 (LEP2)	electron	3096	3.3
14000 (LHC)	proton	3096	$6 \cdot 10^{-6}$
500	electron	3096	111.6
1000	electron	3096	1785.8

Table 3.1: Energy losses due to synchrotron radiation for different accelerator setups. The first two lines are similar to LEP and LEP2 respectively. The third line corresponds to an LHC like ring. The last two lines correspond to storage rings with center of mass energies as they are foreseen for the ILC.

3.1 The Accelerator

The International Linear Collider (ILC, a detailed description can be found in [23] and [24]) is a future linear collider in which electrons and positrons will be brought to collision with a center of mass energy of 200-500 GeV and a possible upgrade to 1 TeV. The peak luminosity foreseen is about $2 \cdot 10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$. The basic design parameters are summarized in Table 3.2. An overview of the layout of the ILC can

Parameter	Value
Center-of-mass energy	$200 - 500$ GeV
Peak Luminosity	$2 \cdot 10^{34}$ cm^{-2}s^{-1}
Average beam current in pulse	9 mA
Pulse rate	5 Hz
Pulse length	~ 1 ms
Number of bunches per pulse	1000-5400
Charge per bunch	1.6-3.2 nC
Accelerating gradient	31.5 MV/m
RF pulse length	1.6 ms
Beam power	10.8 MW
Beams size at IP	640×5.7 nm
Total AC power consumption	230 MW

Table 3.2: Basic design parameters of the International Linear Collider [23].

be seen in Figure 3.1. The main parts are the electron and positron sources, the two damping rings (one for the electrons, one for the positrons), the two main linacs (one for the electrons, one for the positrons) and the beam delivery system.
The electrons are created by illuminating a photo cathode with a laser and are accelerated to 5 GeV. Before injecting them into the electron damping ring their spin vectors are rotated so that all of them show into the same direction. The electron beam is then used to create positrons. It is passed through the main linac and a helical undulator where photons are produced via photoproduction. The photons hit a Ti-alloy target where electron-positron pairs are created. The positrons are separated from the electrons and accelerated to 5 GeV. Before injecting the positrons into the positron damping ring their spins are rotated such that they show into the same direction.

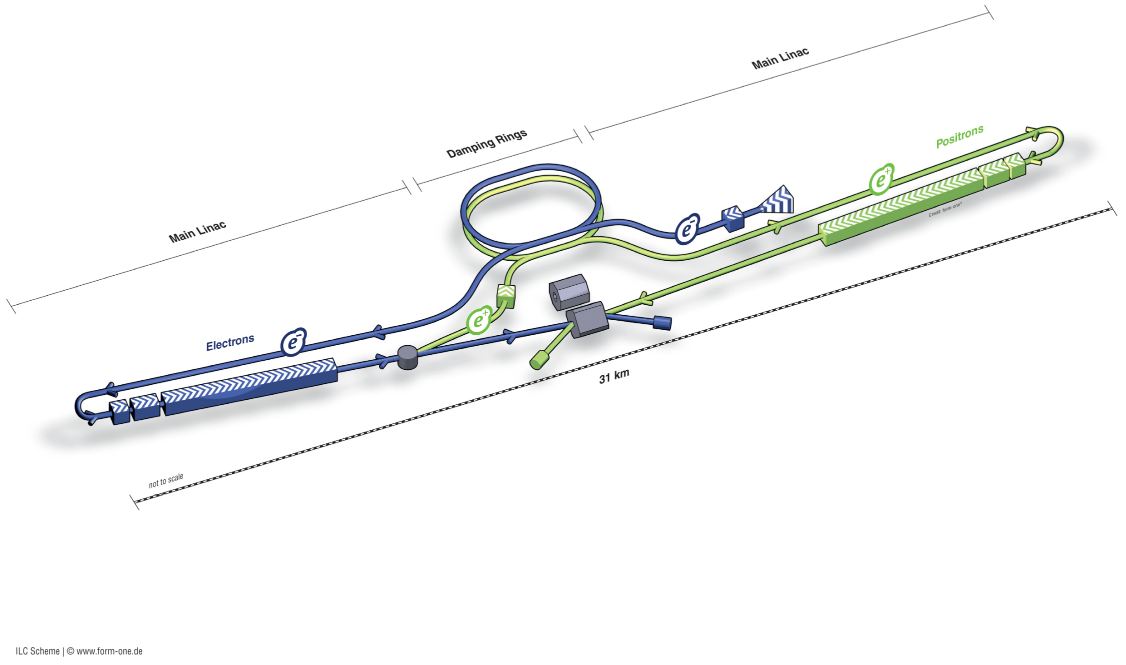


Figure 3.1: The International Linear Collider [25].

The damping rings operate at a beam energy of 5 GeV. Their task is to stabilize the beams which are then transported to the main linacs. On the way to the main linacs the bunches are compressed and the spins are rotated to the orientation desired at the point of interaction.
The two main linacs accelerate the electron and positron beam to the final beam energy over a length of about 23 km. The accelerating gradient is achieved by superconducting radio-frequency accelerating cavities. The beam delivery system finally transports the beam from the main linacs to the point of collision and then to the beam dump. Before bringing the beams to collision they are focused, the energy and the polarization of the beams is monitored. The beams collide in the interaction region with an angle of 14 mrad. The small angle is necessary to avoid the outgoing beam to run into the incoming beam. The interaction region is shared by two detectors which can be moved into and out of the beam line (push-pull configuration).
The advantage of a linear collider is that it can be run at any center-of-mass energy in the given range and it can be upgraded to higher energies by extending the main linacs to larger length. Thus the machine can be adapted to the needs depending on the processes that should be measured. Another advantage is that the polarization of the beams is preserved during the acceleration process in a linear collider. The polarization is important in electroweak processes at high energies because the couplings to the gauge bosons are different. By choosing the polarization of the beams the event rates can be enhanced for certain processes under investigation or certain processes can be suppressed to reduce background.
Another advantage is that leptons will be brought to collision. This leads to much cleaner events, there are less pileup and less underlying events. The biggest back-

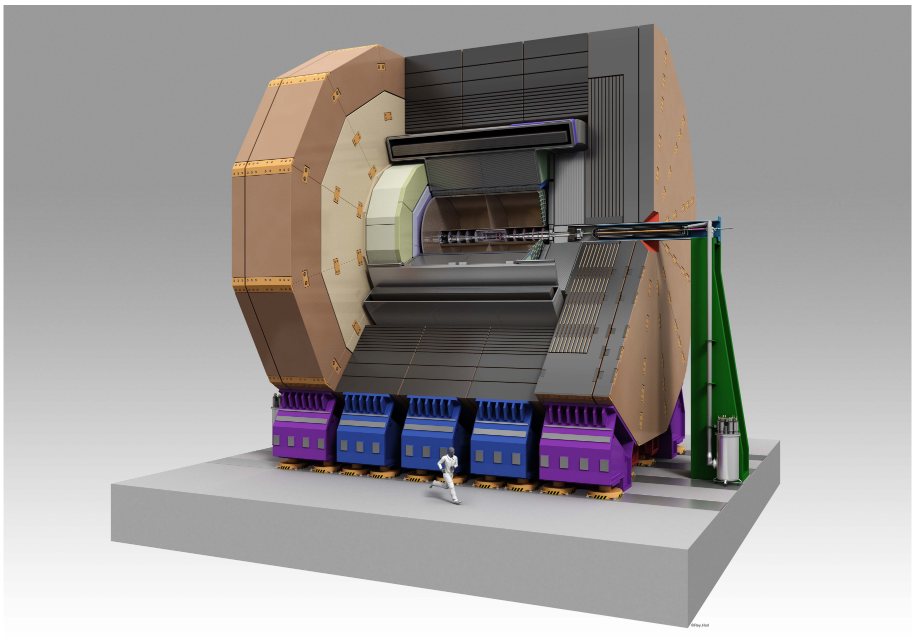


Figure 3.2: The International Large Detector [28].

ground will come from secondary electron-positron pairs, but they only appear in a very narrow region around the beam.
Furthermore the theoretical predictions for cross-sections are known very precisely for the ILC. By searching for deviations of the cross-sections one could find hints for new particles.

3.2 The International Large Detector

One of the detector concepts foreseen for the International Linear Collider is the International Large Detector (ILD) concept [26, 27]. An overview of the detector can be seen in Figure 3.2. It is a multi-purpose detector following the particle flow concept which means that every measurement in the detector is assigned to one particle. This leads to special requirements the detector has to fulfill, such as a good tracking performance (in particular a good momentum resolution) and a high granular calorimeter. The material budget of the tracking system must be as low as possible so that the particles interact as little with the detector material as possible before entering the calorimeter. The calorimeter on the other hand must have a high material budget so that the whole energy of the particle can be measured. A short overview of all detector components is given in the following.
The tracking system of the ILD comprises several detectors. The innermost detector is the vertex detector (VTX) with very good spatial resolution (better than 3 µm). This is needed for the detection of secondary vertices which is important to reconstruct final states including heavy flavor particles. The vertex detector is followed by the silicon strip detector (SIT) in the barrel and the forward tracking detector (FTD) in the end cap region. The latter one provides coverage close to the beam

pipe. The largest of the tracking detectors is a time projection chamber (TPC) with the order of 200 measurement points. Since it is a gaseous detector the material budget added by the TPC is low. More details on the ILD TPC will be discussed in Chapter 4.3. To improve the tracking performance additional silicon detectors surround the TPC: the ETD in the end cap region and the SET in the barrel region.
The calorimeter system consists of a highly segmented electromagnetic calorimeter (ECAL) followed by a highly segmented hadronic calorimeter (HCAL). The gaps in the very forward region are filled with high precision calorimeters. They need to be radiation hard because they are located very close to the beam pipe. These detectors are the LumiCAL, which will measure the luminosity with very high precision, the BeamCAL, which estimates the luminosity bunch by bunch, and the LHCAL, which extends the coverage of the HCAL.
The whole detector is surrounded by a superconducting magnet providing an axial magnetic field of 3.5T which is needed for momentum measurements. The return yoke is instrumented with detectors which serve as a tail catcher for the calorimeter and as muon detectors.
Overall the ILD detector will be about fifteen meters high and located on a platform so that it can be moved in and out off the beam line as a whole.

3.3 Physics at the ILC

Since the center-of-mass energy can be changed relatively easily at the ILC and the cross-sections of the physics processes depend on the center-of-mass energy a large variety of physics processes can be studied. The cross-sections of some of these processes are shown in Figure 3.3. It is planned to run at different energies. At 250 GeV the cross-section of the Higgs-strahlung process (which will be discussed later) is almost at a maximum and one of the most important processes to be studied at the ILC. At energies higher than 500 GeV processes including supersymmetry are expected to start playing a role. At the ILC polarized beams are available. By choosing different polarizations signal processes can be enhanced or background processes can be suppressed [30]. This feature is important for searches for new physics where certain standard model processes can be reduced by choosing the proper polarization.

3.3.1 Higgs

As mentioned before, one of the most important processes to be studied at the ILC is the Higgs-strahlung process. The Feynman diagram for this process is shown in Figure 3.4. At the LHC it was observed that the Higgs boson couples to vector bosons [5, 6]. Thus the production processes will be available at the ILC.
In this process the Higgs mass can be calculated by using the mass recoiling against the Z boson. The Higgs mass is given by

$$M_{\mathrm{H}}^2 \;=\; s + M_{\mathrm{Z}}^2 - 2E_{\mathrm{Z}}\sqrt{s}, \tag{3.2}$$

where $\sqrt{s}$ is the center-of-mass energy, M_{Z} is the mass of the Z boson and E_{Z} is the energy of the Z boson [31]. The center-of-mass-energy is given, the mass of the Z

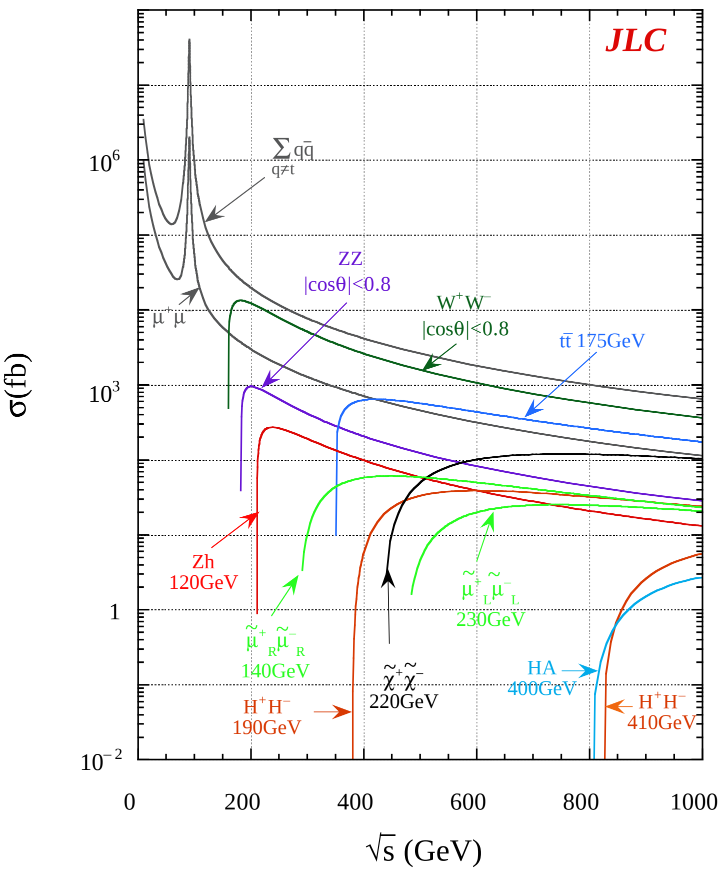


Figure 3.3: Cross sections of different physics processes available at the ILC [29].

boson is known well. The energy of the Z boson is calculated from the momenta of its decay products. Thus a detector is required which provides high momentum resolution. In case the Z boson decays to electrons or muons the momentum of the particles will be measured in the tracking system of the detector. For the ILD a time projection chamber was chosen as the main tracking device. It provides many measurement points and a long lever arm. Both is essential for a good momentum resolution (see Chapter 4.1). Figure 3.5 shows how the recoil mass distribution

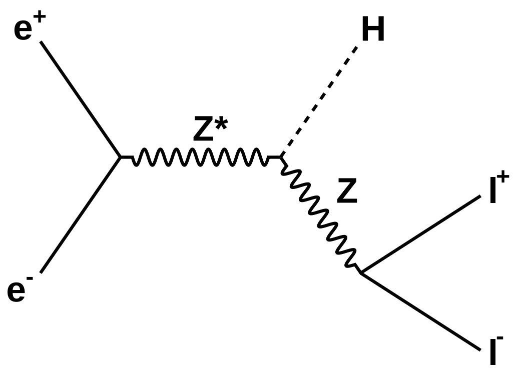


Figure 3.4: Feynman diagram of the Higgs-strahlung process where the Z boson decays leptonically.

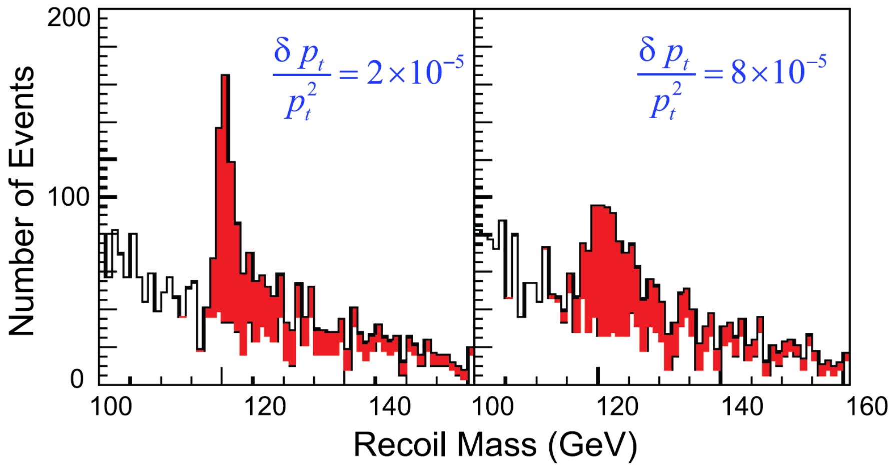


Figure 3.5: Mass recoiling from Z decaying into two muons at $\sqrt{s}$ = 500 GeV for two different momentum resolutions and a simulated Higgs mass of 120 GeV [32].

changes for different momentum resolutions. Only small changes broaden the signal peak significantly.
The advantage of using the recoil mass is that the Higgs mass can be calculated without using the decay products of the Higgs boson. This method is thus model independent and also invisible decay modes are taken into account.

3.3.2 SUSY

Supersymmetry has not yet been discovered and large region of the parameter space are excluded by the LHC results. However, there are still SUSY models that could be realized in nature (for example natural SUSY [33,34]). Most of the models involve small charge differences between the two lightest supersymmetric particles (LSP and NLSP) (a few GeV) and include light charginos and neutralinos (masses below 200 GeV) [30]. These particles, if they exist, can thus be produced at the ILC.
Particularly challenging is the small mass splitting of the LSP and the NLSP. The NLSP will decay into the LSP (which is expected to be a neutralino ($\tilde{\chi}_1^0$)) and standard model particles. Only the standard model particles can be measured in the detector, the chargino will add missing energy. Since the mass splitting between LSP and NLSP is small, the standard model particles will have small momenta which need to be measured precisely. This requires efficient tracking and good momentum resolution of the main tracking device, especially for tracks with large curvature.
A study on how the momentum resolution influences results on SUSY analysis is for example shown in [35]. The SPS1a' model [36] (where the mass difference between $\tilde{\tau}_1$ and $\tilde{\chi}_1^0$ is small (about 10 GeV)) was used to measure the τ polarization in $\tilde{\tau}_1$ decays. Figure 3.6 shows the analysis result for different energy resolutions (the energy is measured via the momentum in the tracker). The worse the resolution gets, the more the result differs from the nominal value.

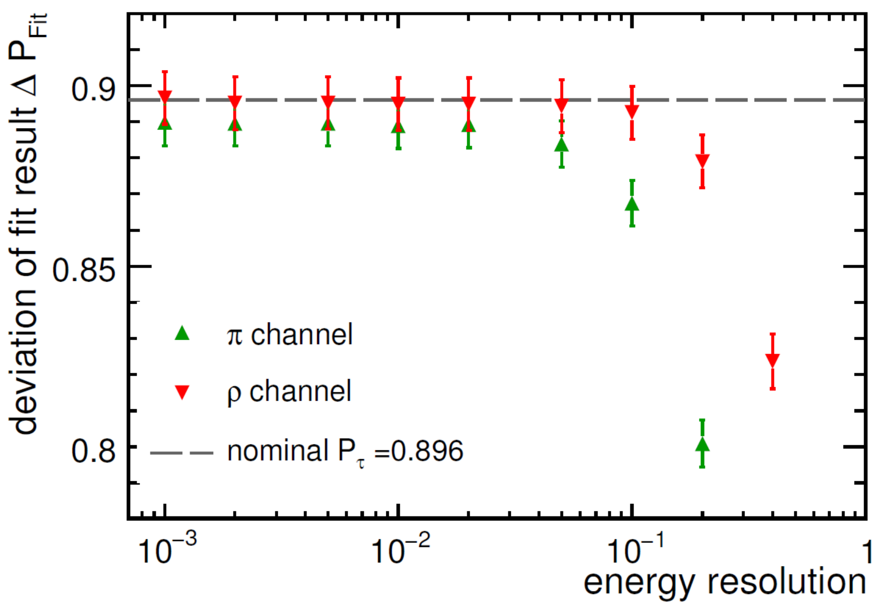


Figure 3.6: The influence of the energy resolution on the result of the measurement of the τ polarization in $\tilde{\tau}_1$ decays [35].

Chapter 4

Time Projection Chambers

4.1 Working Principle of Time Projection Chambers

A time projection chamber (TPC) [37] mainly consists of a volume filled with gas. An electric field (drift field) is applied to this volume. In order to keep the electric field as uniform as possible between anode and cathode a field cage is installed. At the anode the amplification and readout structure is located. If a charged particle

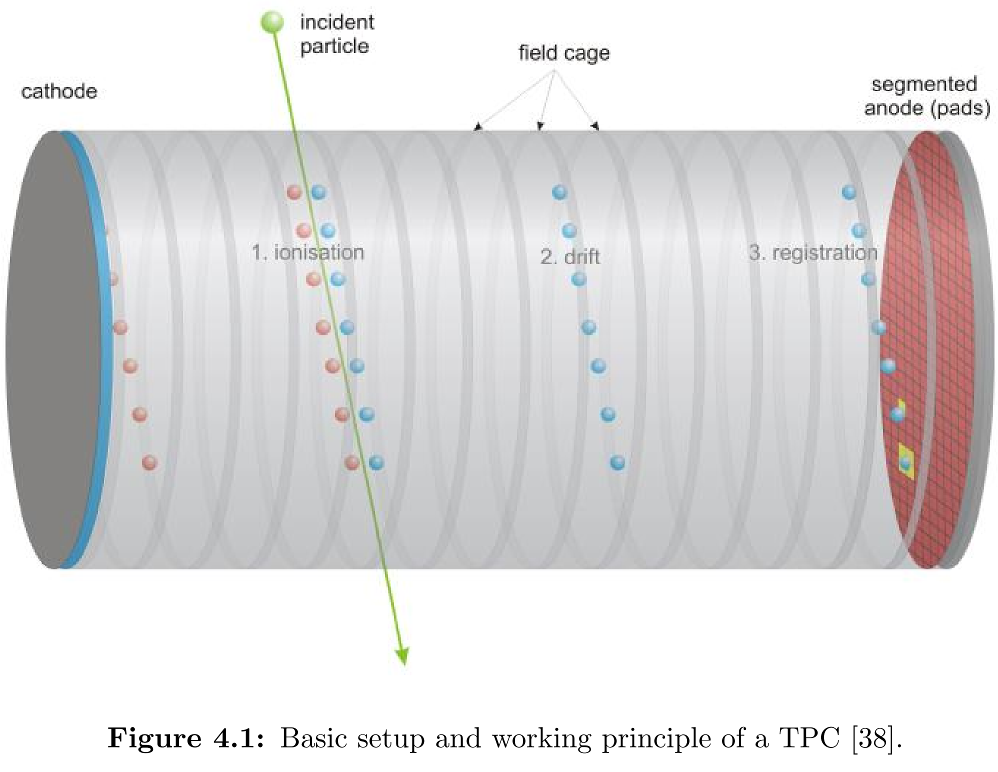


Figure 4.1: Basic setup and working principle of a TPC [38].

travels through the volume it scatters inelastically with the electrons bound in the atoms of the gas, which leads to the ionization of the atoms. Due to the electric field the electrons produced by the ionization drift towards the anode where they are amplified and read out in the two-dimensional readout structure. The collected

Gas	***W* in eV**	***N_T* in 1/cm**
Ar	26	97
CH_4	30	54
iC_4H_{10}	26	220
CF_4	54	120

Table 4.1: Properties of some gases at 20 °C and a pressure of 1 atm [4]. W is the energy needed to build an electron-ion pair. N_T is the number of electrons produced by a minimum ionizing particle per centimeter.

charge and the arrival time of the signal are recorded. This information allows the reconstruction of three-dimensional space points with which tracks can be reconstructed.
Time projection chambers are usually operated in a magnetic field. Its direction is usually chosen to be parallel to the drift direction. The field forces the charged particles on helicoidal trajectories in such a way that the track projected on a plane perpendicular to the drift direction is a circle. From the curvature Ω of the circle the transverse momentum of the particles can be calculated via

$$p_T = \frac{qB}{\Omega} \tag{4.1}$$

where q is the electric charge of the particle and B is the strength of the magnetic field.
The choice of the gas in the TPC is very important. The number of electrons produced by the incoming particle, drift velocity of the electrons as well as the diffusion strongly depend on the atomic and molecular structure of the gas. The properties of some gases are summarized in Table 4.1. The drift velocity of electrons is given by the steady state solution of the Langevin equation [39]:

$$\vec{v}_{\text{drift}} = \frac{e}{m}\tau|\vec{E}|\frac{1}{1+\omega^2\tau^2}\left(\hat{E} + \omega\tau(\hat{E}\times\hat{B}) + \omega^2\tau^2(\hat{E}\cdot\hat{B})\hat{B}\right) \tag{4.2}$$

where $\hat{E}$ and $\hat{B}$ are the unit vectors along the electric and magnetic field. Thus the drift velocity only depends on the direction of the magnetic field, not on its strength. If the electric and magnetic field are parallel the drift velocity only has a component along the fields. It is given by

$$v_{\text{drift}} = \frac{eE\tau}{m_e}, \tag{4.3}$$

where e is the electron charge and m_e is the electron mass. E represents the strength of the electric field. τ is the mean collision time of the drifting electron with the gas atoms in the TPC. In this case the drift velocity is independent of the magnetic field and only depends on the strength of the electric field and on the type of gas used in the TPC.
The spread of the charge cloud at the time t of the first collision is given by the expectation value of the square of the distance an electron can travel along one

direction in space (for example the x-axis) in this time. In case no fields are present this equation reads:

$$\delta_0^2 = \int_0^\infty dt \int_{-1}^{1} d\cos\theta \frac{1}{\tau} \exp\left(-\frac{t}{\tau}\right) \frac{1}{2} \left(\lambda \frac{t}{\tau} \cos\theta\right)^2 \tag{4.4}$$

$$= \frac{2}{3}\lambda^2. \tag{4.5}$$

where λ and τ are the mean free path length and the time before the drifting electron collides with a particle from the gas [40]. θ is the angle between the particle trajectory and the direction is space under investigation (for example the x-axis). For $t \gg \tau$ the number of collisions is $n = t/\tau$ and the spread of the charge cloud after n collisions is

$$\sigma_0^2 = \frac{2}{3}\lambda^2 \frac{t}{\tau} \tag{4.6}$$

The diffusion coefficient in the absence of a magnetic field is then given by

$$D_0 = \frac{\lambda^2}{3\tau}. \tag{4.7}$$

If a magnetic field parallel to the drift direction is present the transverse diffusion is affected [40]. The distance an electron travels before a collision λ stays the same, but the electron moves on a curved trajectory, and thus the projection of the trajectory on one axis is smaller than in the case where no magnetic field is present, see Figure 4.2. In this case the diffusion coefficient is given by

$$D_T = \frac{D_0}{1 + \omega^2\tau^2} \tag{4.8}$$

with

$$\omega = \frac{eB}{m_e}. \tag{4.9}$$

A detailed introduction on the drift of charged particles in gas can be found in [39].

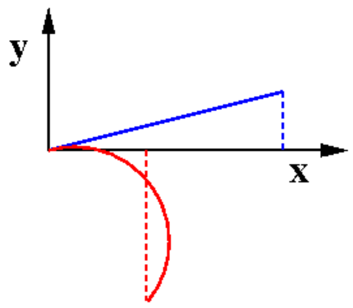


Figure 4.2: The sketch shows the trajectory of a single electron with (red solid line) and without (blue solid line) magnetic field. The spread of the charge cloud is calculated using the projection of the trajectory on one direction in space (here the x-axis). In the presence of a magnetic field this value is smaller because the electrons move on curved trajectories.

Usually noble gases mixed with a small amount of polyatomic gases is used. Nobel gases are chosen because they are chemically stable and small energies are needed to produce electron-ion pairs. The small amounts of other gases in the gas mixture serve as a quencher gas. The quencher gas is needed to absorb photons which are emitted by excited atoms. This process produces a large amount of photons during amplification of the signal (see Section 4.2). The photons have an energy

high enough to produce additional electrons by ionization. In order to keep the amplification proportional to the number of primary electrons the photons need to be removed [41].
Additionally, by adding different gases, properties such as the drift velocity and the diffusion can be adjusted.
The drift velocity and the diffusion do not only depend on the kind of gas used in the TPC but also on the drift field applied to the TPC. After having chosen a gas the drift field is chosen. Usually the maximum drift velocity and the minimum diffusion do not coincide, thus a compromise needs to be found. For the testbeam data which will be analyzed in Chapter 9 the drift field is set such that the diffusion is minimal, accepting a slightly smaller drift velocity. The drift field could also be set such that the drift velocity is maximal. This configuration is stable against changes in the drift field which is useful if strong field distortions are expected.
The number of electrons produced in the primary ionization is too low to be read out. Thus the signal needs to be amplified before it reaches the readout structure. There are different amplification technologies under investigation which are explained in detail in Section 4.2.
Since the TPC basically consists of a volume filled with gas the material budget is low. This is an important requirement for tracking systems because energy losses of the particles have to be kept as low as possible in the tracking detectors. Another advantage of TPCs is the continuous tracking achieved by a finely segmented readout. The relation between the transverse momentum resolution and the single point resolution in the readout plane (for more details see Chapter 9.4.5.1) is given by [42]

$$\frac{\delta p_T}{p_T^2} \propto \frac{\sigma_{r\phi}}{BL^2\sqrt{N}}, \tag{4.10}$$

where p_T is the transverse momentum, $\sigma_{r\phi}$ is the single point resolution in the readout plane, B is the magnetic field, L is the length of the lever arm and N the number of measurements. The single point resolution is much larger than the one of silicon detectors but due to the large amount of measurement points and the long lever arm overall a high momentum resolution can be achieved, which is essential for a good tracking performance.
As can be seen in Table 4.1 only a very small amount of energy is transferred from the incoming particle in the ionization process. The mean amount of energy deposited per a certain distance dE/dx depends on the momentum of the incoming particle and is specific for each type of particle. The energy loss is given by the Bethe-Bloch equation [39, 4])

$$\frac{dE}{dx} = \frac{4\pi e^4 N_A Z\rho}{mc^2 A\beta^2} z^2 \left(\ln\frac{2mc^2\beta^2\gamma^2}{I} - \beta^2 - \frac{\delta(\beta)}{2} \right) \tag{4.11}$$

with electron charge e, electron mass m, speed of light c, Avogadro's number N_A, atomic number Z, atomic weight A and charge of the incoming particle z. β is the speed of the incoming particle divided by the speed of light and $\gamma = 1/\sqrt{1-\beta^2}$. $\delta(\beta)$ describes correction for the effect that polarized atoms shield the field of the incoming particle [39, 4].

By measuring dE/dx particle identification can be done in TPCs. Such measurements were for example performed with the ALEPH TPC [43] and the ALICE TPC [44]. The results are shown in Figure 4.3. Other experiments having used time projection chambers successfully are for example DELPHI [45] and STAR [46].

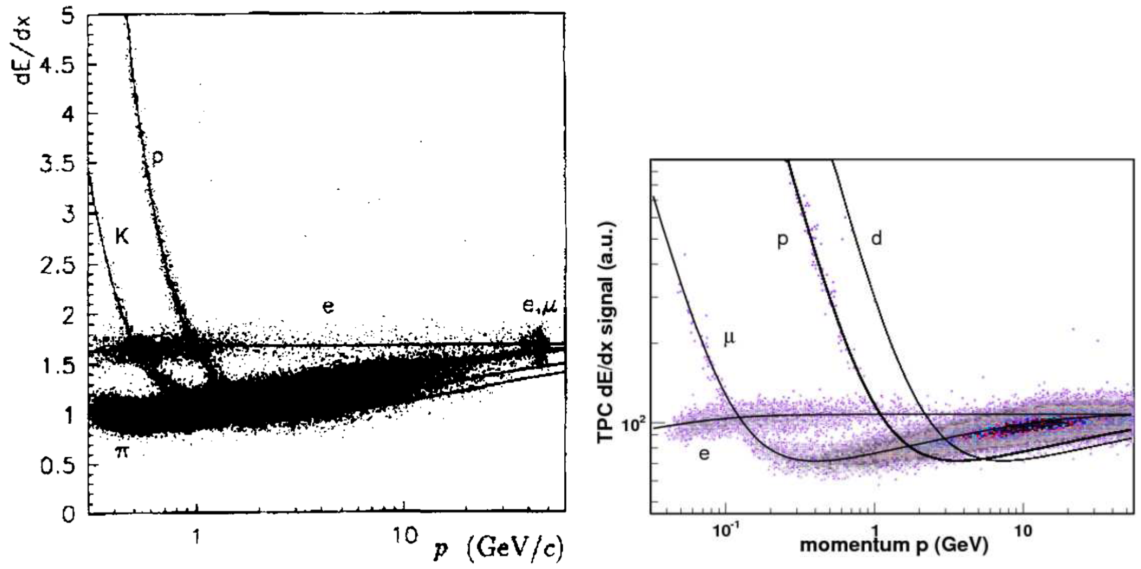


Figure 4.3: dE/dx measurement with the ALEPH TPC [47] and with the ALICE TPC [48].

4.2 Micro Pattern Gas Detectors

Before reading out the signal it needs to be amplified. In the past this has been done with proportional wires [4]. Proportional wires cannot be used for the ILD TPC. The $E \times B$ effects are large, thus the required resolution cannot be achieved. Furthermore the distance between the wires cannot be small enough for mechanical reasons and the material budget for wires (especially for the holding structure) is large. Apart from that gating is needed to avoid the ions produced during amplification drifting back into the drift volume. With the event rates expected at the ILC this is technically not possible because the TPC would miss a number of events while being in the gating cycle. Thus for the ILD TPC a new amplification technique is needed. Currently micro pattern gas detectors (MPGDs) are the preferred option. The two technologies currently under investigation are presented in the following.

4.2.1 Gas Electron Multipliers

Gas Electron Multipliers (GEMs, [49]) are foils consisting of an insulator, often Kapton, covered with copper on both sides. The foil has holes in a regular pattern. A photo of such a foil is shown in Figure 4.4. The Kapton and copper layer usually are about 50 µm and 5 µm thick, respectively. The hole diameters and the pitch are of the order of 100 µm, however, the values can vary.

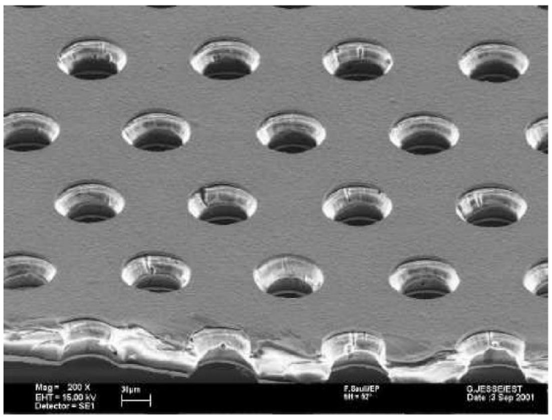


Figure 4.4: Photo of a GEM taken with an electron microscope [50].

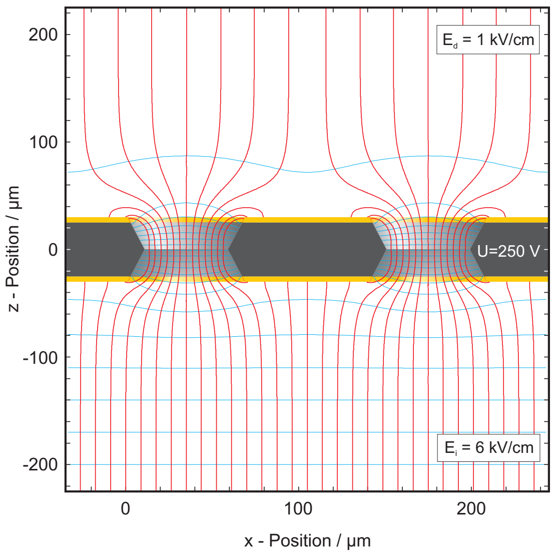


Figure 4.5: Field lines in a typical GEM amplification structure [51].

A voltage is applied between the two copper layers. This results in an electrical field as sketched in Figure 4.5. The solid lines mark the field lines. The field is strongest inside the holes. This is where the electrons gain enough energy for gas amplification in an avalanche process. A simulation of the amplification in one GEM hole is shown in Figure 4.6. The bars represent the GEM foil. The gap between

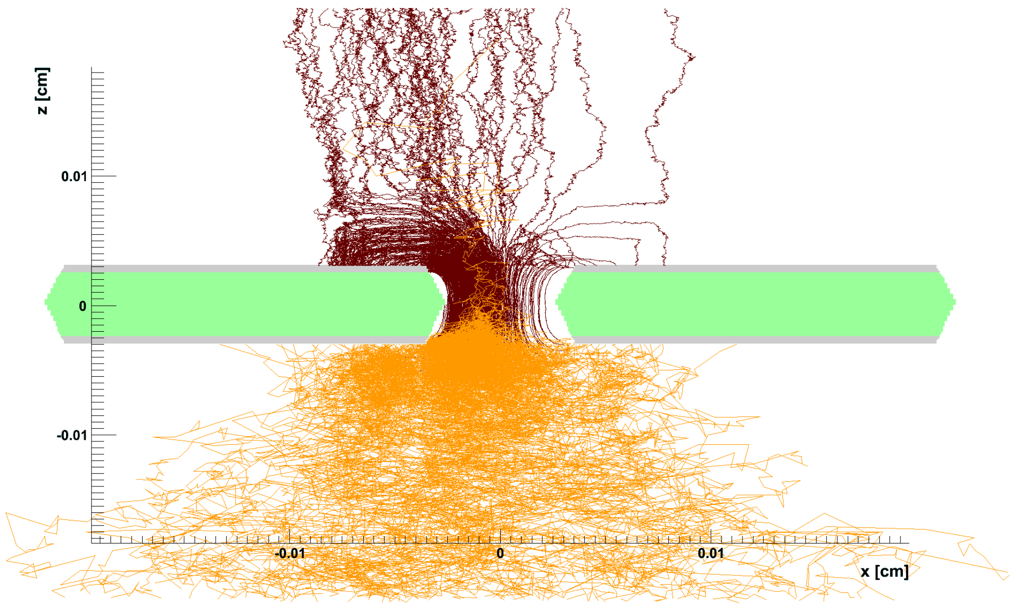


Figure 4.6: Simulation of the amplification in GEM hole [52].

the two bars is a GEM hole. The yellow lines represent the electrons, the red lines represent the ions created during the amplification. The part above the GEM foil is the drift volume where a single electron is started in the simulation. The electron drifts into a GEM hole where gas amplification takes place and new electrons start to drift in the GEM hole. The same amount of ions is produced which drift into the opposite direction, back into the drift volume of the TPC. The ions cause field distortions in the TPC, thus it is necessary to reduce this effect. However, most of the ions follow the electric field lines and end up on top of the GEM. Thus GEMs intrinsically reduce the number of electrons drifting back into the drift volume of the TPC. To achieve the required amplification with one GEM would require a high voltage being applied to it. In this situation the system is not stable. Instead of using only one GEM for amplification several GEMs stacked on top of each other are used with lower voltages applied to each of them. Studies of the behavior of different configurations can be found in [53]. With such a structure additionally the diffusion and the ion back drift can be optimized. The GEM distances (transfer gaps) and voltages are chosen such that the extraction and collection efficiency is as large as possible[1]. The field between the last GEM and the readout structure (induction field) is chosen such that the diffusion is largest to spread the signal over several pads[2]. If more pads measure charges the position of a hit[3] can be reconstructed with higher accuracy.

4.2.2 Micromegas

With GEMs several foils need to be stacked on top of each other to get sufficient amplification. Another possible amplification technology under investigation are micro mesh gaseous structures (Micromegas). The amplification takes place between a thin micro mesh and the readout structure. The two structures are 25 - 150 µm [4] apart.
Micromegas deliver enough amplification with only one structure with little charge spread. For pad based readout this is a clear drawback because the charge must be spread over several pads as was explained in Section 4.2.1. For pixelized readout this is not necessarily a disadvantage, however for pixels other unanswered questions exists, like how the huge amount of channels could be read out at an ILD TPC. A photo of a Micromegas mesh is shown in Figure 4.7. The micro mesh is separated from the readout structure by insulating pillars.

4.3 A TPC for the International Large Detector

For the International Large Detector a time projection chamber is foreseen, which is described in detail in [27]. It will be of cylindrical shape, divided into two half-chambers. The walls of the cylinder will contain a field cage to ensure a homogeneous electric field over the full length of the TPC. It will add a material budget of about 0.05 of a radiation length. The cathode will be placed in the center while there will

[1]As many electrons produced during amplification as possible have to enter the volume between the GEMs and as many as possible of them have to be collected in the holes of the next GEM.

[2]Often the distance between the last GEM and the readout structure is chosen larger than the distances between the GEMs. The electrons travel a longer distance and the electron cloud is spread more.

[3]A point in 3D-space. More details on the reconstruction can be found in Chapter 5.1.3.

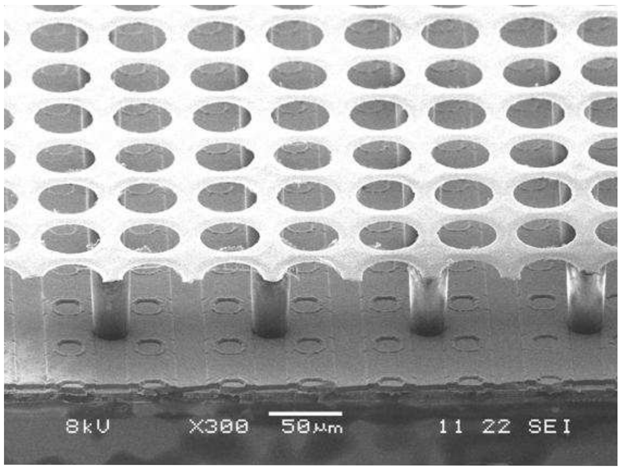


Figure 4.7: Photo of a Micromegas mesh on top of a readout structure (in this case pixel readout). The two structures are separated by insulating pillars [4].

be two anodes, one at each end of the TPC. The anode endplates are designed to have as little mass as possible while being stable at the same time. They will have a material budget smaller than 0.25 radiation lengths. The endplates will hold the readout modules assembled in concentric rings around the beam pipe. Each module will include the gas amplification, a readout plane to collect the electrons, and the readout electronics. The signal will be amplified with micro pattern gas detectors. It has not yet been decided which of the amplification technology under consideration will be used. The electrons will be collected on pads of roughly 1×6 mm^2 size. To minimize the number of ions drifting from the amplification structure into the drift volume of the TPC gating will be used after each ILC bunch train.
A point resolution of about 60 µm in the $r\phi$ plane and about 0.4 mm in the rz plane is required. The two-hit resolution in $r\phi$ is required to be 2 mm while it is 6 mm in the rz plane. The momentum resolution in the presence of a magnetic field of 3.5T will be $\delta(1/p_t) \simeq 10^{-4}/\text{GeV}/c$. A summary of the design parameters and the resolution goals is given in Table 4.2.
The whole detector will be located in a magnetic field of 3.5T which is essential for momentum measurements with the TPC. In order to achieve a good tracking performance the field must be homogeneous. However, for the TPC no such requirements are specified [54]. Instead the magnetic field will be measured precisely and a field map will be created before the inner detectors are mounted in the magnet. A precision of $1 \cdot 10^{-4}$T is required. According to the magnetic field map the data can be corrected and thus inhomogeneities do not worsen the performance of the TPC.

4.4 Prototype development for an ILD-TPC

In order to prove that it is possible to build a TPC for the ILD detector with the given requirements various prototypes have been built for this purpose. Here only two prototypes are presented briefly which were (and are) used at DESY and data taken with those prototypes were used for this thesis.

Design Parameters:	
Inner Radius	329 mm
Outer Radius	1808 mm
Length	4700 mm
Solid Angle Coverage	Up to $\cos\theta \simeq 0.98$
Material Budget (outer field cage)	$\simeq 0.05X_0$
Material Budget (readout endcaps)	$< 0.25X_0$
Number of Pads	$\simeq 1 - 2 \times 10^6$ per endcap
Pad Size	$\simeq 1 \times 6$ mm^2 (about 220 rows)
Resolution Goals:	
Point resolution in $r\phi$	$\simeq 60$ µm for zero drift
	< 100 µm over full drift length
Point resolution in rz	$\simeq 0.4$ mm for zero drift
	$\simeq 1.4$ mm over full drift length
2-hit resolution in $r\phi$	$\simeq 2$ mm
2-hit resolution in rz	$\simeq 6$ mm
dE/dx resolution	$\simeq 5\%$
Momentum resolution (3.5 T)	$\delta(1/p_t) \simeq 10^{-4}$/GeV/$c$, TPC only

Table 4.2: Design parameter and resolution goals for a TPC in the ILD [27].

The first prototype is called MediTPC. It is a small prototype which was built to prove the principle of MPDGs in TPCs and to investigate the achievable point resolution. Details on the prototype and on the analysis of data taken with it can be found in Chapter 5.3.
The second chamber is the Large Prototype. The goals for this prototype are the development of readout modules as they could be used in the ILD TPC, study the technical challenges of building a large TPC and to investigate momentum resolution. A detailed description can be found in Chapter 9.1.2.

Chapter 5

Track Reconstruction

5.1 Track Reconstruction Algorithms

In this chapter the algorithms with which the testbeam data were reconstructed are presented. It should be noted that the development of the algorithms is still ongoing and improvements are constantly done. The algorithms are presented here as they were at the time when the testbeam data (see Chapter 9) were reconstructed.
In TPCs a charge spectrum (charge over time) is measured for each electronics channel. The final goal is to reconstruct tracks from these data. This task is divided into various steps which will be discussed here. The algorithms described are specific for a padbased TPC where the charge is collected on macroscopic copper structures.

5.1.1 Pulse Finding

In the first step the charges measured on each readout channel (so called pads) are combined to pulses (see Figure 5.1). This is done by searching the charge spectrum of each channel until a charge larger than a given value is found in one time bin. If such a bin was found from that bin on the charge spectrum is searched for a time bin in which the charge drops below a given end threshold.
Several requirements can be specified which the pulse has to fulfill, such as the minimum length and the minimum height. Additionally bins before the start and after the end of the pulse are saved to ensure that the full charge information is kept.
A pulse is characterized by its time and its total charge. The total charge is the integral over the charges in all time bins belonging to the pulse. The pulse time is calculated from the mean of the rising edge as sketched in Figure 5.2.
It can happen that double pulses are measured. Such pulses are split in two by investigating at the derivative of the charge development in time. When a falling edge turns into a rising edge the pulse is split at this position. The difference is required to be larger than the random fluctuations in the charge spectrum (which are expected to be significantly smaller (about 10 %) than the required minimum height of a pulse) to ensure that the pulse is not split due to such fluctuation.

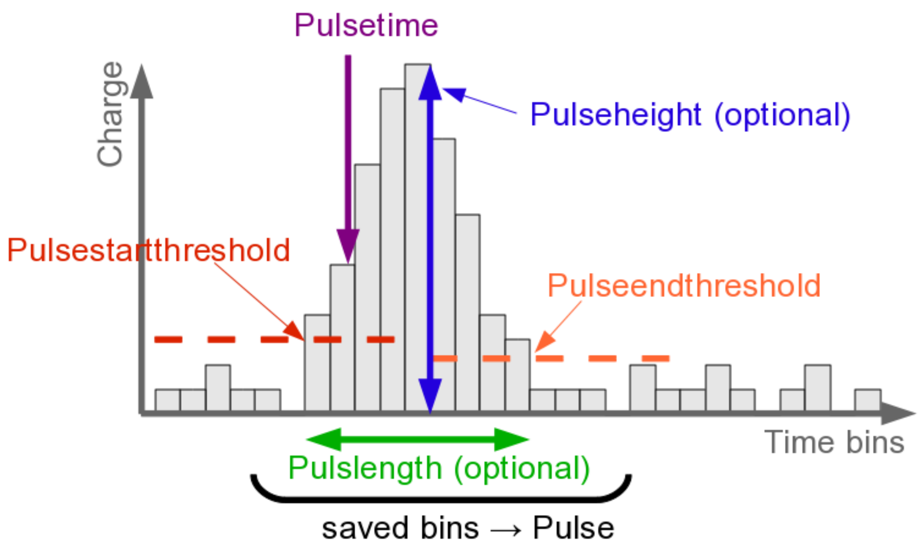


Figure 5.1: The pulse reconstruction algorithm. In the charge spectrum regions need to be found in which the charges in the time bins are higher than certain thresholds. Cuts can be applied on the minimum pulse height and the minimum pulse length. The pulse is defined by the time and the total collected charge in the pulse. In order to get a realistic value for the total charge not only the bins with charges higher than the thresholds but also some bins before and after the pulse are taken into account.

5.1.2 Hit Finding

In the second step the pulses having been found on the pads are combined to 3D-space points (so called hits). First pulses which have been measured on adjacent pads in the same row and have similar pulse times are combined. After that the positions of the hits in 3D-space are calculated. Here two cases must be discriminated: rectangular pad planes and circular pad planes. Both cases are sketched in Figure 5.3.
For rectangular pad planes the coordinate system is chosen such that x-axis points along the pad rows, the y-axis is perpendicular to the x-axis and the z-axis is parallel to the drift direction of the electrons in the TPC. In this coordinate system the x-coordinate of a hit is calculated by the center of gravity of the pulse charges in the hit. The y-coordinate is given by the center of the pad row. The z-coordinate is calculated from the pulse time of the highest pulse and the drift velocity in the TPC.
For circular pad planes cylindrical coordinates are used. The z-axis is parallel to the drift direction as before and the z-coordinate of the hits is calculated as described above. The ϕ-axis points along the pad rows and the R-axis is chosen to be perpendicular to the pad rows. The ϕ-coordinate of a hit is calculated by the center of gravity and the R-coordinate is the center of the pad, analog to the rectangular pad plane.
The position of the hit corresponds the a position in the TPC where an incoming particle crossed the sensitive volume. Thus after the hit finding a number of points along the track in the detector are known.

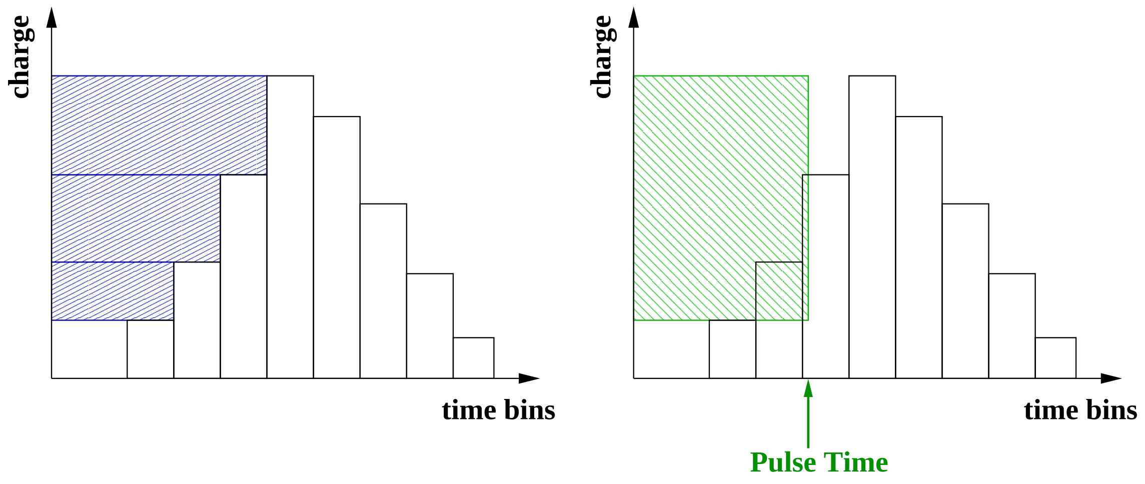


Figure 5.2: Determination of the pulse time. In the left sketch the area between the pulse and the y-axis (charge axis) is marked. In the right sketch a rectangle is constructed which starts at the y-axis and extends in y as far as the hatched region in the left sketch. The extension of the rectangle in the right sketch in x (time axis) is chosen such that the rectangle has the same area as the hatched region in the left sketch. The pulse time is given by the width of the rectangle in the right sketch in x.

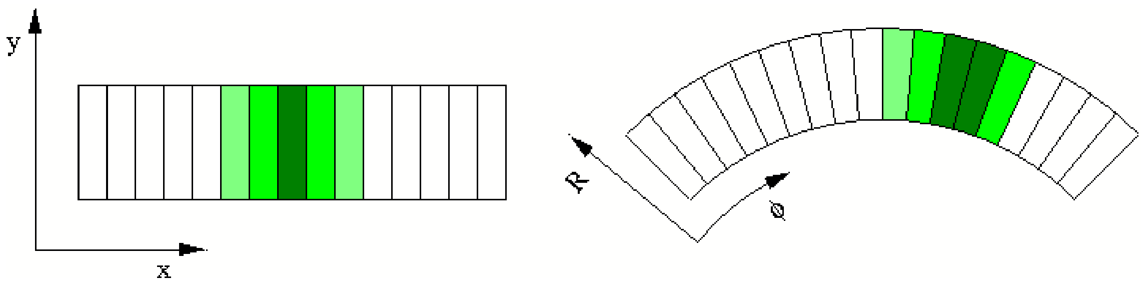


Figure 5.3: Hit reconstruction on rectangular (left) and circular (right) padplanes. One pad row is shown each. The different colors represent measured pulses with different charges.

Additionally to the position a hit has a charge which is the sum of the charges of the pulses in the hit. The hit charge can be used to weight the hits differently in the following reconstruction steps.

5.1.3 Tracking

The hits from the previous section mark the trajectory the initial particle took through the TPC. In the third step of the reconstruction the trajectory is reconstructed by combining hits which were most likely produced by the same particle. The track is fitted afterwards to obtain the track parameters. From the track parameters properties such as the momentum of the particle can be reconstructed which are important for the later event selection and analysis.
Track finding is done by pattern recognition methods. The algorithm used for the reconstruction of testbeam data (see Chapter 9) is based on a Hough transformation.

The algorithm is presented in detail in Chapter 7. For an overview of alternative track finding methods see Chapter 6.
During track finding the hits are only combined in groups belonging to one track. The parameters describing the trajectory need to be determined in an additional step. This is done by a track fit. At the time the reconstruction of the testbeam data was done no reliable track fit was available. Thus the tracks are not fitted. For the analysis described in Chapter 9 the track parameters delivered by the Hough transformation are used. The parameters are only a rough estimate but the results are reasonable.

5.2 Track Reconstruction Software

For the reconstruction and analysis of data taken with detectors at a future linear collider several software packages were developed. In this chapter the packages used for this thesis are presented shortly: the data format in which data are stored (`LCIO`), the detector description (`GEAR`), the software package to reconstruct TPC data (`MarlinTPC`) and a software package providing a full simulation based on `geant4` [55, 56] (`Mokka`).

5.2.1 `LCIO`

`LCIO` (Linear Collider I/O) [57, 58] provides a data format in which data are stored in runs and events. Different classes for different purposes are implemented. For this thesis the following classes are used:

- `TrackerData`: Contains cell ids and ADC values (i.e. the charge spectrum measured on each pad).
- `TrackerPulse`: Contains charge, time and cell ids as well as the TrackerData belonging to the pulse.
- `TrackerHit`: Contains position of hits as well as the hit charge.
- `Track`: Contains all hits on the track as well as track parameters. For more details on `LCIO` track parameters see Section 5.2.1.1.
- `MCParticle`: Class for simulated particles. It holds information such as the energy and the momentum of the particle, its pdg id [4] as well as the parent particles and daughter particles, if there are any.
- `SimTrackerHit`: In principle the same as `TrackerHit`, but it also holds the information which Monte Carlo particle produced the hit.
- `LCRelation`: A class which holds the ids of two other objects (which can be of any type) which are related. For example a simulation creates `SimTrackerHits`. These hits are smeared and merged according to the detector resolution and are saved as `TrackerHits`. `LCRelations` can be used between the `TrackerHit` and the `SimTrackerHit`. Via the relation the Monte Carlo information is accessible for `TrackerHits`.

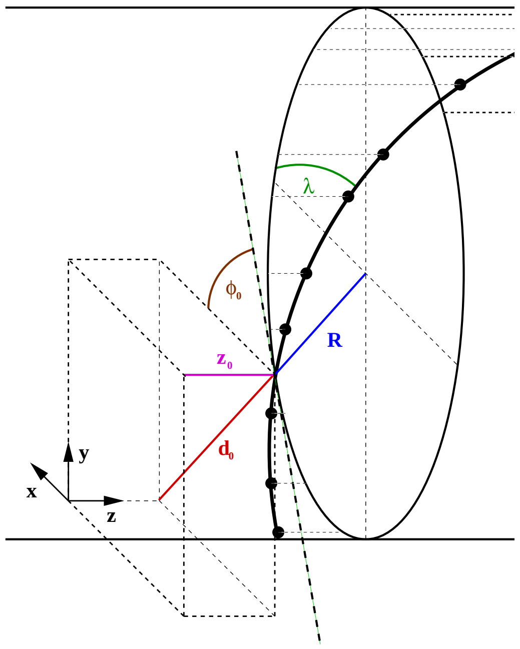


Figure 5.4: `LCIO` Track Parameters.

These are mostly classes related to TPCs or tracking detectors in general. The exceptions are `MCParticle` and `LCRelation` which are not specific to any subdetector. A number of classes for other subdetectors are implemented. For a more detailed list, see the `LCIO` webpage [58].
After the full reconstruction of TPC data the `LCIO` file contains events with a collection of `TrackerData`, a collection of `TrackerPulses`, a collection `TrackerHits` and a collection of `Tracks`. Each collection contains all objects of the corresponding type found in the event.

5.2.1.1 Parametrization of Tracks in `LCIO`

With the `LCIO` track parameters two types of track shape need to be described by the same set of parameters. Since the TPC will be located in a magnetic field tracks will have helicoidal shapes so the track parameters need to describe helices. For particles with high energy the effect the magnetic field has on the particle is small. Trajectories of such particles are almost straight lines. The parameters have to describe tracks in this limit as well.
The set of parameters used in `LCIO` are perigee parameters, parameters which are chosen with respect to the point of closest approach of the track to a reference point (usually the origin of the coordinate system). The parameters are chosen such that they are Gaussian distributed. A detailed description of the parameters is given in [59] and [60].
The coordinate system is chosen such that the z-axis is parallel to the magnetic field. The xy-plane is perpendicular to the z-axis so that the readout plane is in this plane. Projecting the helix onto the xy-plane gives (neglecting losses of energy) a circle which is described by the following parameters:

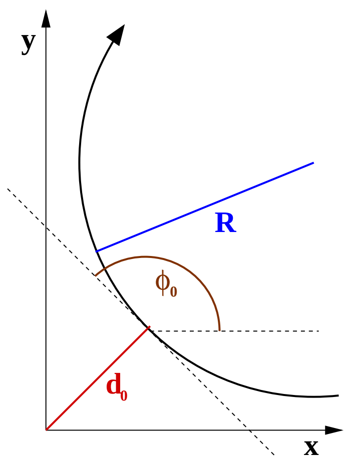


Figure 5.5: Track parameters for a helix in the xy-projection.

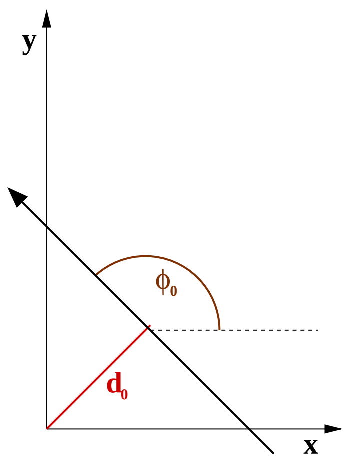


Figure 5.6: Track parameters for a straight line in the xy-projection.

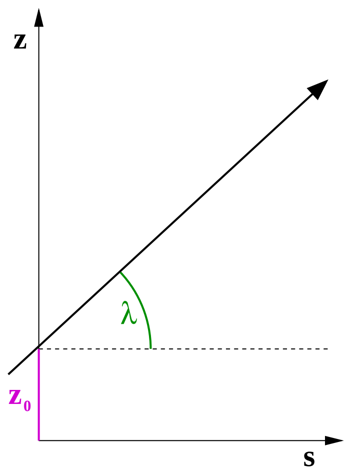


Figure 5.7: Track parameters for a straight line in the sz-projection.

- $\boldsymbol{d_0}$: This is the distance of the point of closest approach (pca) to the reference point, i.e. the shortest distance between track and origin. The sign of d_0 depends on the sign of Ω. d_0 and Ω have the same sign if the reference point is located inside the circle, they have opposite signs if the reference point is located outside the circle.
- $\boldsymbol{\phi_0}$: This is the angle between the momentum of the particle at the point of closest approach and the positive x-axis (azimuthal angle of the momentum).
- $\boldsymbol{\Omega}$: This is the curvature of the circle, $|\Omega| = \frac{1}{R}$. The sign of Ω is defined such that it is positive for clockwise motion of the particle in the xy-plane and negative for counter-clockwise motion of the particle in the xy-plane.

The parameters and their signs are also shown in Figure 5.8. The signs depend on the direction of motion of the particle. So the `LCIO` parameters discriminate between particles having the same trajectory but moving in opposite directions.
In order to describe the whole helix two more parameters are needed. To define these, first the arc length s of the circle in the xy-plane is defined. It is zero at the point of closest approach. s is positive in the direction of motion of the particle and negative in the opposite direction. In the sz-projection a track is a straight line. To describe the straight line in the sz-plane the following parameters are chosen:

- $\mathbf{tan}\,\boldsymbol{\lambda}$: This is the slope of the straight line in the sz-plane.
- $\boldsymbol{z_0}$: This is the z-position of the point of closest approach or, in other words, the offset of the straight line.

In total five parameters are needed to describe a helix in space: three for the circle in the xy and two for the straight line in the sz-plane.
A straight line is a special case of a helix (a helix with curvature $\Omega = 0$). In this limit the previously described parameters describe a straight line in 3D-space.

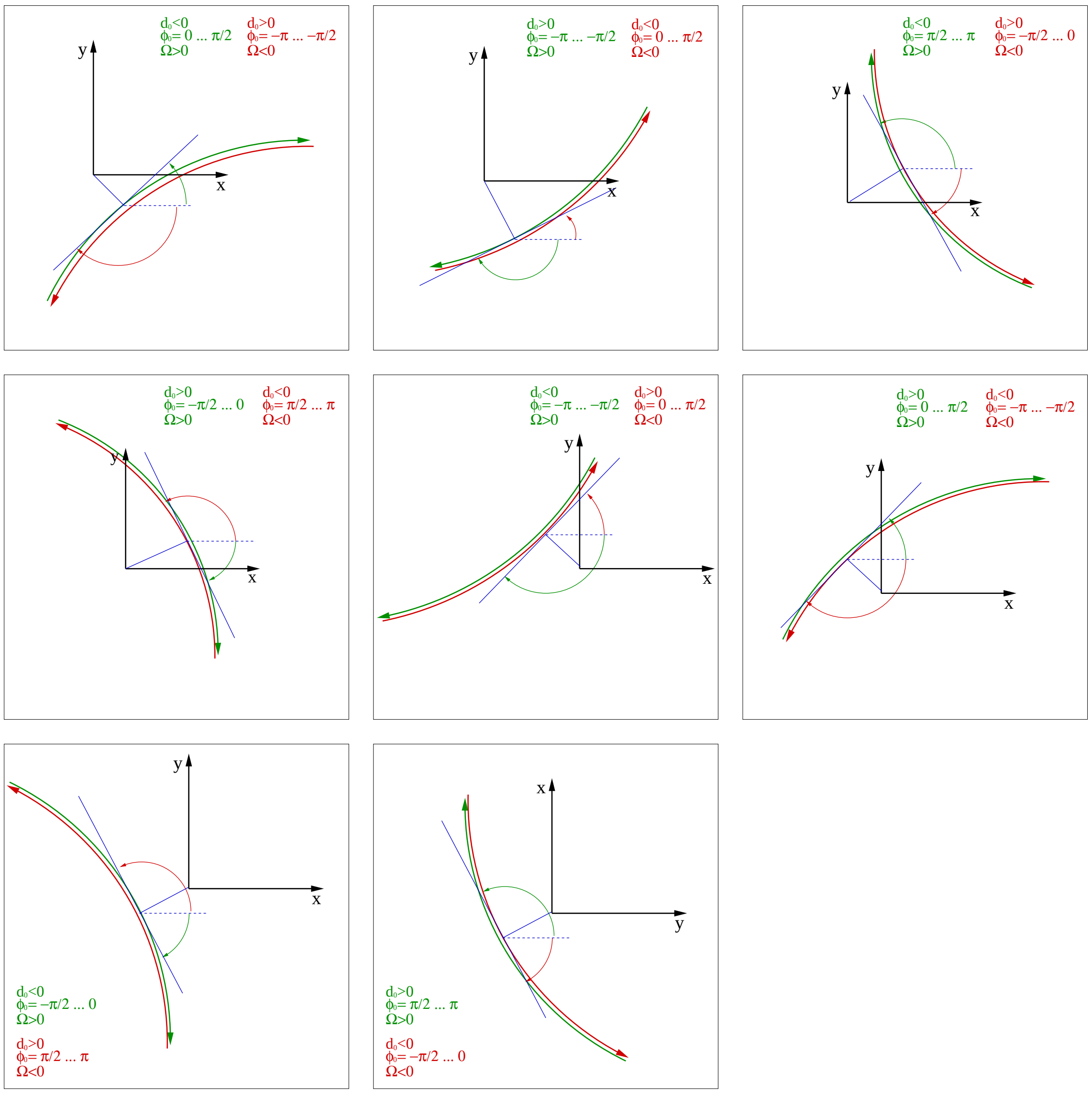


Figure 5.8: `LCIO` Track Parameters and their signs in the xy-plane.

5.2.2 `GEAR`

`GEAR` (GEometry API for Reconstruction) [61] is a software package providing an interface to use detector geometry descriptions during data reconstruction. The detector description has to be provided via an `XML` file [62]. It can easily be exchanged so that the same reconstruction chain can be run for different setups.

5.2.3 `Mokka`

`Mokka` [63, 64] provides a simulation based on `geant4` [55, 56] and uses a realistic detector description. The most commonly used detector models are collected in a database and are thus easily accessible.
`Mokka` automatically stores simulated data (such as `MCParticles` and `SimTrack-erHits`) in an `LCIO` file. Additionally a `GEAR` file is created which containing the

description of the detector used in the simulation. Both files are used in the reconstruction which is performed with `Marlin`.

5.2.4 `Marlin`

`Marlin` [65] stands for Modular Analysis and Reconstruction for the Linear Collider and is a modular `C++` software framework using `LCIO` as data format and `GEAR` for detector descriptions. It provides the possibility to process events written in `LCIO` files. The steering is done via `XML` [62]. The modularity of `Marlin` allows to easily exchange parts of code (so called processors) with other parts or use certain parts several times with different parameters without recompiling after each change.

5.2.5 `MarlinTPC`

The software package used for the analysis and reconstruction of testbeam data taken with TPC prototypes is called `MarlinTPC` [66,67,68] and was developed for exactly this purpose. It is based on `Marlin` and provides a number of software components (so called processors) needed for the reconstruction. For each step in the reconstruction chain also several processors following different algorithms were implemented. They can easily be exchanged. Also the steering parameters for each processor can be changed easily.
An example reconstruction chain for pad-based TPC data is shown in Figure 5.9. Reconstruction starts on `TrackerData` which are the charge spectra measured on each pad. In the `TrackerData` pulses are searched with the `PulseFinderProcessor`. The `TrackerPulses` delivered by this processor are the input for the hit finding which is performed by the `RowBasedHitFinder-Processor`. This reconstruction step gives `TrackerHits` which are then used for the track finding. The track finding used in this thesis is done with processors providing interfaces to external tracking packages in which the track finding algorithms are implemented. There are currently two such packages available: a Kalman Filter and a Hough transformation. This thesis focuses on the package providing a Hough transformation only. A detailed description of the tracking package belonging to it will be given in Chapter 7.

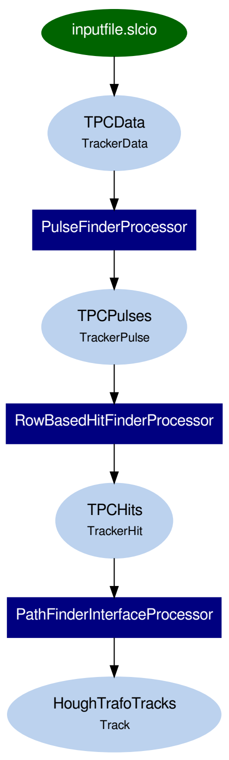


Figure 5.9: Example `MarlinTPC` reconstruction chain.

5.3 Validation of the Reconstruction Algorithms

The reconstruction algorithms for the low level reconstruction (pulse and hit reconstruction) described in Chapter 5.1 were first implemented in a software package called `MultiFit` [69]. This package was used to reconstruct data taken with small prototypes and is well understood. The software package `MarlinTPC` replaces the `MultiFit` package. Due to its modular structure `MarlinTPC` can be used for any prototype and any readout technology. The low level reconstruction (pulse finding and hit finding) as it was implemented in `MultiFit` was reimplemented in `MarlinTPC`. In order to validate that `MarlinTPC` does the same as `MultiFit` the two software packages are compared on pulse and hit level. For the comparison cosmic data taken with a small prototype called MediTPC are used. In the following the results of this comparison will be presented.

5.3.1 The Data

The MediTPC [69,70] is a prototype with a diameter of 27 cm and a maximum drift length of 66 cm. During data taking it was filled with P5 gas (95% Ar, 5% CH_4) and was located in a 4T magnetic field. The pad plane had twelve rows and a total of about 600 pads in a staggered layout. Each pad had a size of 1.27×7 mm^2. A drift field of about 91.5 V/cm was used which yields the maximum drift velocity for P5 gas, as shown in Figure 5.10. A triple GEM stack was used for amplification.

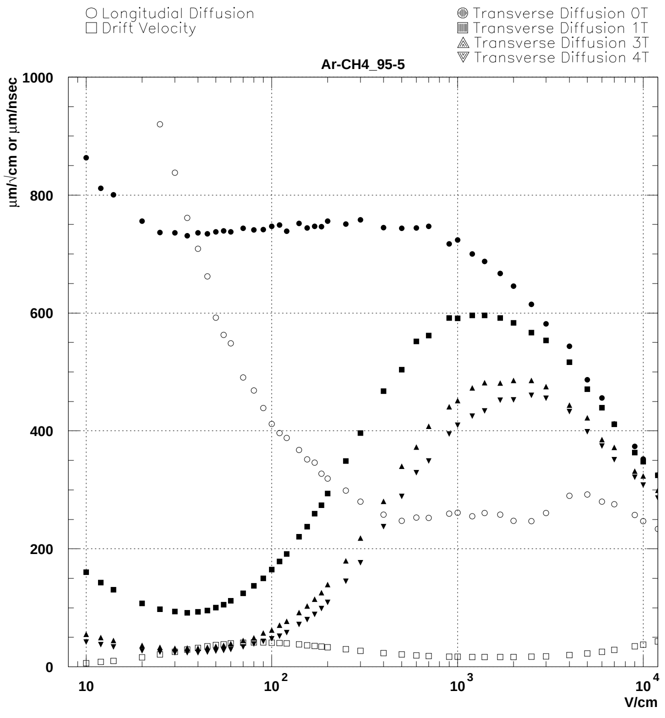


Figure 5.10: Plot showing gas properties simulated with `MAGBOLTZ` for P5 gas [71].

The gap between the GEMs was 2 mm wide, the gap between GEM and pad board was 3 mm wide. The voltages of the GEMs are listed in Table 5.3.1. These values

GEM, Side	Voltage [V]
Anode GEM, anode side	900
Anode GEM, cathode side	1220
Middle GEM, anode side	1520
Middle GEM, cathode side	1840
Drift GEM, anode side	2140
Drift GEM, cathode side	2465

Table 5.1: Voltages applied to the GEMs.

correspond to a field of 1500 V/cm between the GEMs and a field of 3000 V/cm between the anode GEM[1] and the pad board. Experience shows that a GEM stack with these settings can be operated stably [69]. The latter voltage was chosen such that the diffusion is maximal so that the signal is spread over as many pads as possible. The other settings are optimized to keep the charge cloud as compact as possible between the GEMs while at the same time having a proper amount of amplification.
With the setup described above cosmic muons were measured. For the comparison about 10000 of these events are used.
The data are first reconstructed and analyzed with `MultiFit`. It consists of three parts (see Figure 5.11), the cluster finding (which corresponds to pulse and hit finding in `MarlinTPC`), the track finding and the track fitting. This software will be replaced by `MarlinTPC` which is a modular software package allowing the exchange of methods used in different reconstruction steps without recompiling. All steps done in `MultiFit` are implemented in `MarlinTPC` and the output of the two is compared using the cosmic data taken with the MediTPC. Comparisons are done at pulse and hit level. The MediTPC data are reconstructed with `MarlinTPC` using the processors and parameters listed in Table 5.3.1. The same parameters are used in the `MultiFit` reconstruction thus it is expected that the two software packages deliver the same results.

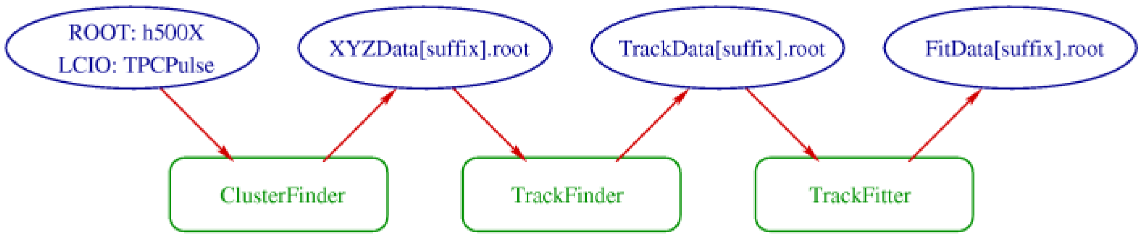


Figure 5.11: Track reconstruction in `MultiFit`. Picture taken from [69].

[1]GEM closest to anode

Steering Parameter	**Value**
Pulse Finder:	
Maximum ADC Value	255 ADC counts
Minimum pulse height	12 ADC counts
Minimum pulse length	5 time bins
Pulse start threshold	7 ADC counts
Pulse end threshold	5 ADC counts
Number of bins saved before start	2
Number of bins saved after end	8
Hit Finder:	
Maximum number of empty consecutive pads	1
Maximum time between pulses in hit	200 ns
Minimum number of pads	1
Drift velocity	39.55 mm/µs

Table 5.2: Steering parameters used for the reconstruction of the MediTPC data.

5.3.2 Validation on Pulse Level

The comparison of `MarlinTPC` and `MultiFit` on pulse level is done pad wise. The most important properties of pulses are compared on each pad. These properties are the number of pulses counted on each pad and the total charge collected on each pad (both integrated over all events), where for the charge only the charge in reconstructed pulses is counted. Other quantities are the average charge, length and height of the pulses on each pad. To calculate the quantities the charge, length and height of all pulses are added for each pad individually and then divided by the number of pulses found on the pad.
First the number of pulses per pad is compared and the result is shown in Figure 5.12. There are large regions where no pulses are found at all which correspond to broken pads. Furthermore a regularly dip appears which corresponds to the end of a row. From the distribution one can see that there is a total of twelve rows and that data from the forth row is missing completely. The overall structure is the same in `MarlinTPC` and `MultiFit`. However, in `MarlinTPC` the number of pulses found per pad is slightly larger. On one pad (pad number 500) many more pulses are found in `MarlinTPC`. The reason for this is that in `MultiFit` this pad is excluded from the reconstruction while it is included in `MarlinTPC`.
The total collected charge in pulses on each pad shown in Figure 5.13 reveals a similar structure. The largest deviation can again be found on pad 500 which is excluded in `MultiFit` but not in `MarlinTPC`. With `MarlinTPC` more charge is reconstructed on each pad. This matches with the previous observation that more pulses are found with `MarlinTPC`. If there are more pulses it is expected that the total charge in pulses on each pad is higher. Thus one would expect that the average charge of the pulses is the same in `MarlinTPC` and `MultiFit`.
The average charge of the reconstructed pulses on each pad in Figure 5.14 shows indeed that the differences between `MarlinTPC` and `MultiFit` are centered around zero. The average pulse length (Figure 5.15) does not show any significant difference

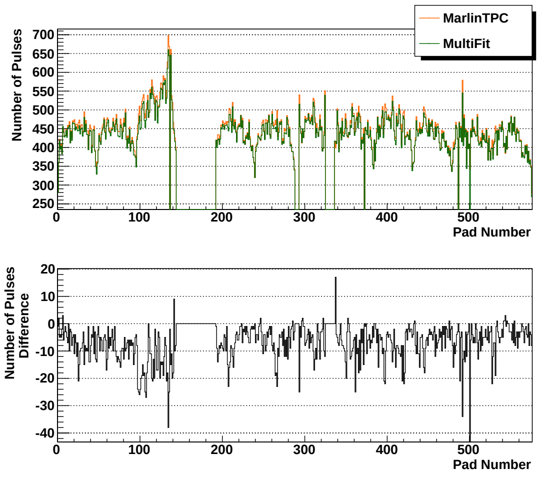


Figure 5.12: The number of pulses found on each pad with `MarlinTPC` and `MultiFit` (upper part) and the differences (`MultiFit` - `MarlinTPC`) between the number of pulses found on each pad (lower part).

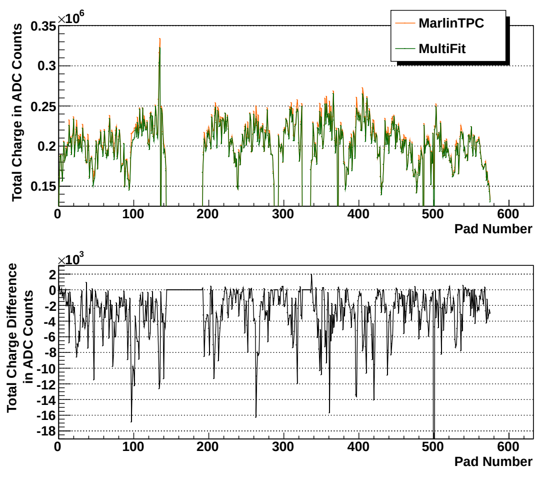


Figure 5.13: Total charge of all pulses reconstructed on each pad with `MarlinTPC` and `MultiFit` (upper part) and the differences (`MultiFit` - `MarlinTPC`) between the total charge on each pad (lower part).

between `MultiFit` and `MarlinTPC` either. The average height of the pulses (Figure 5.16) on the other hand is slightly higher in `MultiFit` but the differences are not as large as the differences in the total charge and the total number of pulses per pad. In the average quantities on two pads significant differences are observed. One of them is again the pad which is included in `MarlinTPC` but not in `MultiFit` (pad number 500). The other one is pad umber 136. In `MarlinTPC` the average pulse length, charge and height are higher than on other pads. A closer look on this pad reveals that it is excluded from the reconstruction with `MultiFit` but not in `MarlinTPC`. But, in contrast to pad 500, in `MarlinTPC` only very few pulses are found. Thus on this pad the number of pulses on the pad and the total charge in pulses on this pad are similar. The average quantities, however, show large differences compared to the other pads. Apart from the pulses on the pads mentioned above the average pulse quantities look the same for `MarlinTPC` and `MultiFit`.
To summarize there are more pulses reconstructed in `MarlinTPC` than in `MultiFit` but the average pulse has the same properties. A close look was taken at the pulses reconstructed with `MarlinTPC` and `MultiFit` to understand the differences. A difference was found between `MarlinTPC` and `MultiFit` on how very long pulses are handled. In `MultiFit` all pulses longer than 40 time bins are cut off. In `MarlinTPC` this is not done and the long pulses are split into several parts. This leads on the one hand to more pulses reconstructed per pad and on the other hand more total charge (only counting charge in a pulse) is reconstructed on each pad. At the same time the average charge, length and height per pulse stay unchanged. All in all the pulse finding in `MarlinTPC` and `MultiFit` gives very similar results and differences are understood.

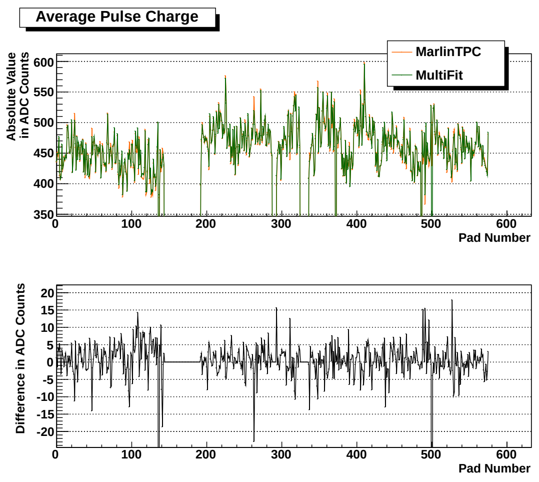


Figure 5.14: Average charge of pulses on each pad reconstructed with `MarlinTPC` and `MultiFit` (upper part) and the differences (`MultiFit - MarlinTPC`) between the average charges on each pad (lower part).

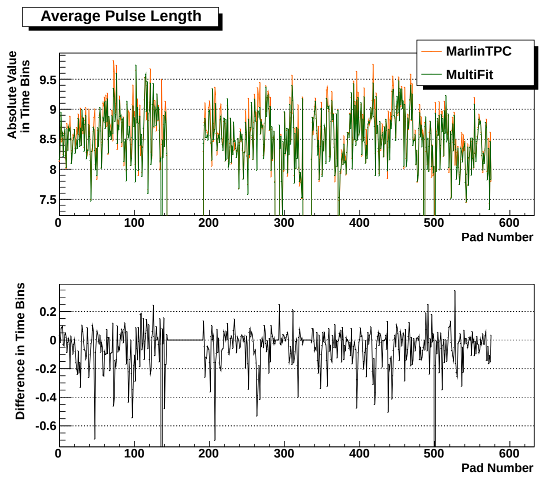


Figure 5.15: Average length of pulses on each pad reconstructed with `MarlinTPC` and `MultiFit` (upper part) and the differences (`MultiFit - MarlinTPC`) between the average length on each pad (lower part).

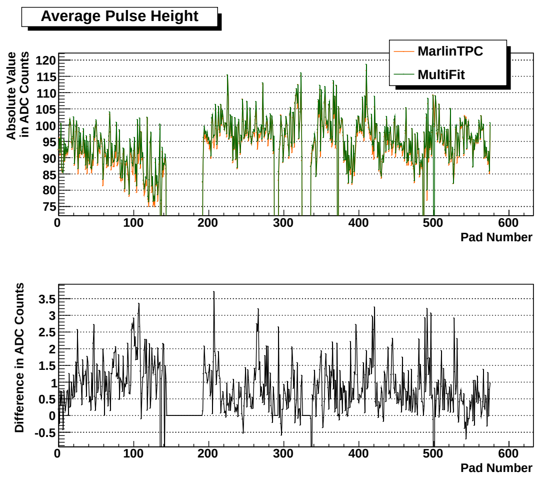


Figure 5.16: Average height of pulses on each pad reconstructed with `MarlinTPC` and `MultiFit` (upper part) and the differences (`MultiFit - MarlinTPC`) between the average height on each pad (lower part).

5.3.3 Validation on Hit Level

For the comparison between `MarlinTPC` and `MultiFit` on hit level the reconstructed hit charges and hit maxima are compared. Since there are small (but well understood) differences on pulse level it is expected that there are also differences on hit level.

The comparison of the hit maxima (the height of the highest pulse in the hit) is shown in Figure 5.17. Overall the shape is the same for `MarlinTPC` and `MultiFit`. The maximum of the distribution is at 90 to 100 ADC counts which is in agreement

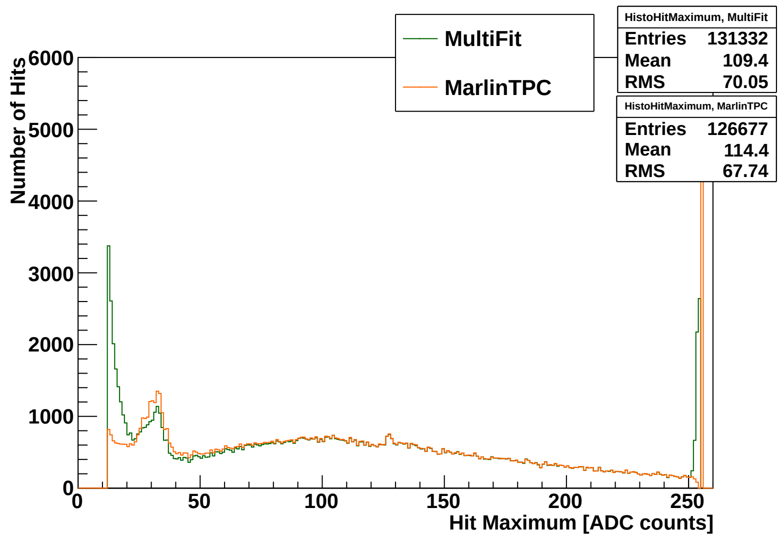


Figure 5.17: Hit maxima reconstructed with `MarlinTPC` and `MultiFit`.

with the average pulse height shown in Figure 5.16. Significant differences occur only for small maxima and for large maxima.

The differences for small hit maxima are shown in Figure 5.18. The distribution starts at 12 ADC counts which corresponds to the minimum height of the hit required during reconstruction. In the reconstruction with `MarlinTPC` less hits with low maxima are reconstructed. Such hits are expected to be noise hits not belonging to any track.

The distribution for large hit maxima is shown in Figure 5.19. The sharp edge at 255 ADC counts is introduced by the readout electronics. No pulses with charges larger than that can be measured. Every signal exceeding this value is cut off. In `MarlinTPC` and `MultiFit` pulses being in overflow are handled differently when pedestals are subtracted. In `MultiFit` pedestals are subtracted in every time bin even if the measured charge in the bin reaches overflow. That is the reason why the distribution of the hit maxima rises a few bins before the maximum ADC value is reached. This is, however, not the correct way to do it because it is unknown by which amount the charge would have increased above 255 ADC counts if the limit had not been there. Furthermore one cannot discriminate between, for example, a time bin which has 255 ADC counts with a subtracted pedestal of 2 ADC counts and a time bin having has 254 ADC counts and a pedestal of 1 ADC count is subtracted. In `MarlinTPC` the pedestal is not subtracted from time bins in overflow. For this reason the shape of the distributions of the hit maxima for `MarlinTPC` and `MultiFit` are different for ADC counts close to 255 ADC counts.

The comparison of the total hit charges is shown in Figure 5.20 where the charges (the sum of all pulse charges in the hit) of all hits are filled into a histogram. The overall shape of the distributions is the same. In the reconstruction with `MarlinTPC` less hits with low charges are reconstructed. These hits correspond to hits with a small maximum (see Figure 5.18) and are expected to be noise hits.

To summarize for the hit finding `MarlinTPC` and `MultiFit` give very similar results. There are some differences for hits with small charges but since they are expected to

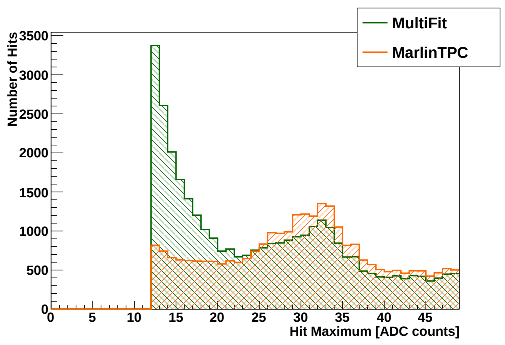


Figure 5.18: The lower range of the hit maxima reconstructed with `MarlinTPC` and `MultiFit`. The sharp edge at 12 ADC counts corresponds to the minimum pulse height required in the reconstruction.

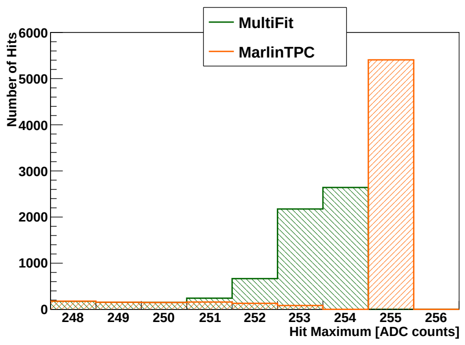


Figure 5.19: The higher range of the hit maxima reconstructed with `MarlinTPC` and `MultiFit` for hit maxima. The sharp edge at 255 ADC counts is due to the fact that in a time bin no higher charge value can be measured (overflow). The differences between `MarlinTPC` and `MultiFit` in the region between 250 and 255 ADC counts is caused by the different handling of the pedestal subtraction in the two algorithms. In `MultiFit` pedestal subtraction is applied to all time bins while in `MarlinTPC` it is only applied to time bins having not been in overflow.

be noise hits not belonging to any track the discrepancies between `MarlinTPC` and `MultiFit` are not problematic for the track reconstruction. The pedestal subtraction of time bins in overflow is done differently in `MarlinTPC` and `MultiFit` which gives differences in the distribution of the hit maxima.

5.3.4 Summary

Overall, on pulse level and on hit level the algorithms implemented in `MarlinTPC` and `MultiFit` give very similar results. There are only small differences in the handling of very long pulses, pulses in overflow (pulses having time bins in which the charge exceeds the maximum value) and hits with small charges. These are all extreme cases which partly need extra algorithms to handle them. Overall the results given by `MarlinTPC` and `MultiFit` are in agreement. Thus the new implementation of pulse and hit finding proved to work.

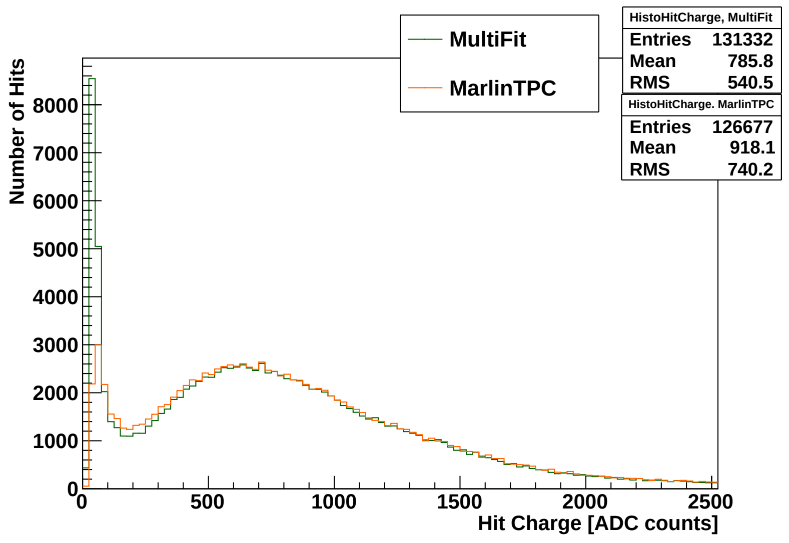


Figure 5.20: Hit charges reconstructed with `MarlinTPC` and `MultiFit`.

Chapter 6

Track Finding

Particles in a magnetic field move on helicoidal trajectories and the measurements taken with a detector are positioned along the trajectories. The task of track finding is to reconstruct the path a particle took through the detector by combining the individual measurements (in TPCs 3D-space points, so called hits) to larger objects called tracks. This is done with pattern recognition methods. From the parameters describing the tracks values important for event selections and further analysis are obtained, such as the transverse momentum. By extrapolating tracks to the inner part of the detector secondary vertices can be reconstructed, by extrapolating tracks to the outer part of the detector the entering point into the calorimeter can be calculated so that measurements in the calorimeter can be matched to the particle which produced the track. The determination of the track parameters is done in a track fit, not during track finding, but in order to get a reliable fit result the measurements must be assigned correctly to the tracks, thus the track finding is an important step in the reconstruction.

Typical challenges are to find tracks with any track parameters reliably. In particle physics experiments tracks with a wide range of momenta occur, from very curved tracks that curl in the detector to almost straight tracks. As many trajectories belonging to real particles must be reconstructed, while the number of reconstructed tracks coming from false combinations (i.e. measurement combinations that are on a helix by chance) has to be kept as low as possible. Also the number of misassigned measurements must be as low as possible because this would lead to a wrong estimation of the track parameters. Another challenge is to discriminate between tracks being close to each other which is particularly difficult in regions with high track density which is especially the case in the center of the detector. Finally track finding needs to work reliably if measurements are missing (detector inefficiencies). A number of pattern recognition algorithms have been developed to face these challenges and some of them are presented in the next sections.

Pattern recognition methods can in general be divided in two groups: local and global methods. In global methods the whole information of the event is used at the same time and in the same way. The most commonly used global methods are the Hough transformation [72] which is a special case of the Radon transformation [73]. These methods will be discussed in more detail in this chapter. Apart from these methods other global pattern recognition methods based on neural network tech-

niques like the Hopfield network [74,75,76], template matching methods [77,78] and conformal mapping [79] are available but will not be discussed here.
Local pattern recognition methods are methods which start with a track hypothesis consisting of very few hits only. Step by step hits are added to the track hypothesis until the track is complete. As an example for local track finding algorithms the combinatorial Kalman Filter [80] will be presented in more detail in this chapter. Other methods are track following algorithms [81,82] and triplet chains [83], which will both not be discussed here.
Local track finding methods are faster than global methods. To apply local methods a detailed description of the detector layout is needed because the algorithms need to know where the next measurement point is expected. Compared to global methods local track finding algorithms cannot cope as well with detector inefficiencies. Also the result of such algorithms strongly depend on the seed tracks used. The reason for this is that local algorithms calculate the position at which the next measurement is expected based on the measurements already included in the track. If the seed track is wrong a track will not be found. In case for detector inefficiencies (i.e. missing measurements) the prediction is not accurate and thus the track cannot be found either.
In practice usually a combination of global and local methods is used. For example first a fast local method can be applied to find easy tracks with only few missing measurements. On the remaining measurements a global method can be applied. It is also possible to apply a global method to create a track hypothesis needed for a local method. A global method can be used on the outer part of the tracking device to find track segments in this region, each consisting of only a few hits. The track segments can then serve as input for a local track finding method which step by step adds hits going to the center of the detector (towards regions where the track density is larger).
Some track finding methods are presented in the following. A more detailed overview on tracking can be found in [81] and [84].

6.1 Radon Transformation

The Radon transformation is a global track finding algorithm. In the measurement space or pattern space tracks are a line (be it a straight line, a circle, a helix, ...) which can be described by one set of parameters. In parameter space tracks are points. The Radon transformation is a transformation between the pattern space and the parameter space. Thus by knowing the space points in pattern space one can calculate the parameter space from which track parameters can be obtained.
The pattern space contains the hit information. Its shape can therefore be described by a hit density function $\rho(x)$ where x is a multidimensional parameter and gives the hit positions in pattern space. This function is transformed to another function which describes the shape of the parameter space $D(p)$, where p is a multidimensional parameter giving the track parameters. A third function is needed which connects the two spaces. It contains information about the track shape, the detector layout and the resolution: $\rho_p(x)$. The functions $\rho(x)$ and $\rho_p(x)$ are known. From

this the function $D(p)$ is calculated:

$$D(p) = \int_X \rho(x)\rho_p(x)dx, \tag{6.1}$$

where X is the full pattern space. For each possible track which is of the shape specified in $\rho_p(x)$ the hits on the track are counted. $D(p)$ has a maximum at the track parameters describing the track with the most hits on. The track finding is thus equivalent to finding the maximum of $D(p)$.
As a simple example a straight line in 2D-space can serve. The hit density function $\rho(x)$ is chosen to be a sum of delta functions, one function for each hit which is non-zero at the position of the hit. The hit density function thus reads

$$\rho(x,y) \quad = \quad \sum_i \delta(x - x_i, y - y_i). \tag{6.2}$$

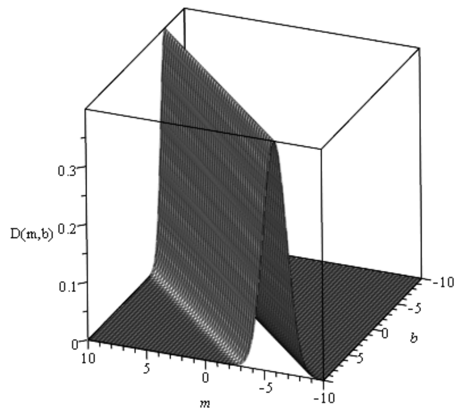
where (x_i, y_i) denotes the hit position of the ith hit and i runs over all hits in the event. The function $\rho_p(x)$ is given as:

$$\rho_{m,b}(x,y) \quad = \quad \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y-mx-b)^2}{2\sigma^2}}. \tag{6.3}$$

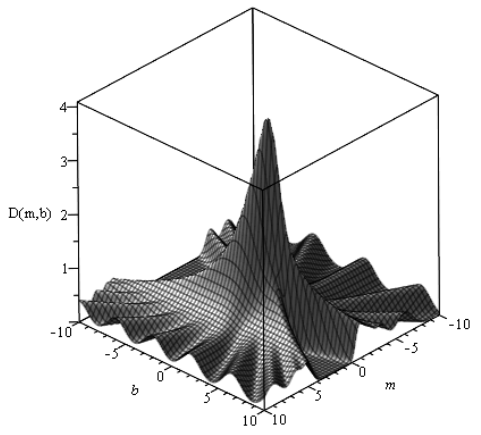
This function has a Gaussian shape with a width σ which represents any kind of effects from detector layout and resolution. m and b are the slope and the offset of the straight line respectively.
With these two functions the function $D(p)$ in parameter space can be calculated:

$$\begin{aligned} D(m,b) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \rho(x,y)\rho_{m,b}(x,y)dxdy & (6.4)\\ &= \sum_i \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y_i-mx_i-b)^2}{2\sigma^2}}. & (6.5)\end{aligned}$$

This function is a sum of functions which are built by inserting different hit position into equation 6.3. In parameter space one individual function has a Gaussian shape with the maximum along a straight line. An example function is depicted in Figure 6.1(a).
In Figure 6.1(b) the function $D(m, b)$ is shown for eleven hits being exactly on the straight line. σ was chosen to be one. Comparing Figures 6.1(a) and 6.1(b) shows that the function $D(m, b)$ develops a maximum when more hits are added. Even though it seems rather straight forward to find a maximum of a function this can still be a challenging task. For realistic events where the hits are not exactly on the track, the function $D(m, b)$ can have secondary maxima. Also it is essential to know the resolution of the detector. If it is assumed to be to large the maximum is washed out, if it is assumed to small the function has many secondary maxima.
Since the Radon transformation corresponds to counting how many hits are on a track candidate it can cope with missing measurements. Even if one hit being compatible with a track is far away from the other hits on the track in pattern space it is counted nevertheless. This is the big advantage of the Radon transformation.



(a)



(b)

Figure 6.1: The radon transform for a straight line and hits being exactly on that straight line. In (a) the function $D(m, b)$ is shown for one hit, in (b) it is shown for eleven hits.

6.2 Hough Transformation

If σ in Equation 6.3 is very small the Gaussian turns into a delta shaped function:

$$D(m, b) \;=\; \sum_i \delta(y_i - mx_i - b). \tag{6.6}$$

This function is not zero if at least one of the delta functions is not zero. This is the case if

$$b = y_i - mx_i. \tag{6.7}$$

This special case of the radon transformation is called Hough transformation and is thus also a global pattern recognition algorithm. As in the Radon transformation any track shape which can be parametrized by one set of parameters can be found and, most important, it can cope with detector inefficiencies. The detector resolution is not used in the Hough tranformation which introduces new challenges which will be discussed below.
In the Hough transformation for each hit all possible tracks are constructed (here straight lines) on which the hit can be. In the parametization chose here the functions are straight lines in the parameter space. If hits are on the same track the functions intersect in parameter space and the point of intersection gives the track parameters.
The Hough transformation was first introduced in [72]. The first application in high energy physics was in 1959 on bubble chamber photographs [85]. An example picture of such a picture is shown in Figure 6.2.
The advantage of the Hough transformation over the Radon transformation is that no information on detector layout and resolution are needed as input. Additionally not the full parameter space needs to be calculated but only those parts which are

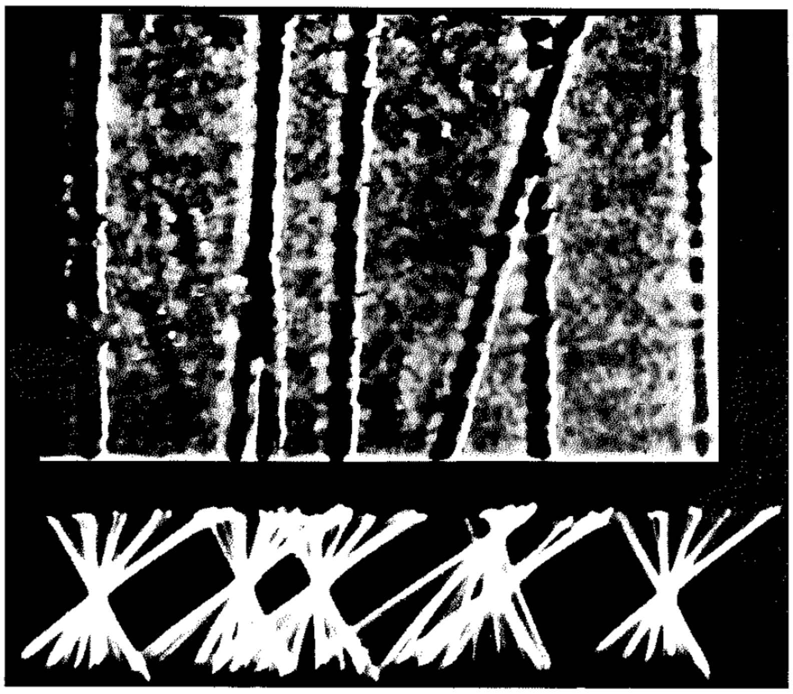


Figure 6.2: First Hough transformation in high energy physics. The picture is taken from [85]. The upper part shows the tracks to be found. In the lower part the transformation is shown.

crossed by one of the functions. This saves time and memory during the computation. Furthermore the calculations needed to be done in the Hough transformation are less complex than for the Radon transformation for which integrals need to be calculated. A disadvantage is that in realistic cases the hits are not exactly on a track which means that the functions do not intersect in exactly one point. This disadvantage is introduced by neglecting the detector resolution. Practically one can circumvent this disadvantage by create a histogram with which one counts how often a bin is crossed by a function. The bin with the maximum number of entries most likely contains the point of intersection. More details on the actual implementation of a Hough transformation can be found in Chapter 7.
Calculating the Hough space can be a time-consuming task, especially for many hits. This can be avoided by applying an adaptive Hough transformation [86]. In this method the Hough space is first calculated rather coarsely and regions which most likely contain points of intersections are identified. The regions are recalculated with a finer binning. This method is much faster because not the full Hough space needs to be calculated.

6.3 Kalman Filter

The combinatorial Kalman Filter [80] is a local track finding method and performs the track finding as well as the track fitting by including new measurements step by step and updating the track parameters after each inclusion. For each hit two steps

need to be done: the prediction step which estimates the new track parameters at the new measurement based on the already included measurements and the filtering step where the new track parameters are calculate including the new measurement. Before going more into detail the notation needs to be clarified. A state vector x_k is used which contains the track parameters after having included k measurements. In the following the predicted state vector at the kth measurement based on the previous $k-1$ measurements will be denoted as x_k^{k-1} and the prediction of the state vector at the kth measurement based on the kth measurement will be denoted as x_k^m. Accordingly, the covariance matrices will be denoted as C_k, C_k^{k-1} and C_k^m.
In the prediction step first the state vector is predicted based on all other previous measurements:

$$x_k^{k-1} = F_{k-1} x_{k-1}. \tag{6.8}$$

This is a linear transformation where F_{k-1} is the matrix propagating the state vector to the position of the next measurement. The covariance matrix is propagated by

$$C_k^{k-1} = F_{k-1} C_{k-1} F_{k-1}^T + Q_{k-1}. \tag{6.9}$$

Q_{k-1} describes random perturbations of the track such as multiple scattering. This prediction is only based on the measurements already included in the track. Another prediction for the state vector can be done by using the kth measurement being included next. The expectation of the kth measurement from the state vector x_k is

$$m_k = H_k x_k \tag{6.10}$$

The predictions for the state vector and the covariance matrix are then formally given by

$$x_k^m = H_k^{-1} m_k \tag{6.11}$$
$$C_k^m = H_k^{-1} V_k (H_k^{-1})^T \tag{6.12}$$

where V_k is the covariance matrix of the measurements. It should be noted that the matrix H_k in general is not invertible because the dimension of the state vector usually is not the same as the dimension of the measurement. However, as will be seen the inversion is not needed.
Now there are two predictions of the state vector at the position of the kth measurement, one based on the track parameters at the point of the previous measurement where all $k-1$ measurements are included, one based on the kth measurement.
The next step is the filtering step. To calculate the state vector the weighted mean of the two predictions is calculated. The weights are the inverse covariance matrices

$$W_k^{k-1} = (C_k^{k-1})^{-1} \tag{6.13}$$
$$\begin{aligned} W_k^m &= (C_k^m)^{-1} \\ &= (H_k^{-1} V_k (H_k^{-1})^T)^{-1} \\ &= H_k^T V_k^{-1} H_k. \end{aligned} \tag{6.14}$$

The weighted mean (thus the track parameters after including the new measurement) is then calculated by

$$\begin{aligned} x_k &= C_k(W_k^{k-1}x_k^{k-1} + W_k^m x_k^m) \\ &= C_k((C_k^{k-1})^{-1}x_k^{k-1} + H_k^T V_k^{-1} H_k H_k^{-1} m_k) \\ &= C_k((C_k^{k-1})^{-1}x_k^{k-1} + H_k^T V_k^{-1} m_k) \end{aligned} \tag{6.15}$$

with

$$\begin{aligned} C_k &= (W_k^{k-1} + W_k^m)^{-1} \\ &= ((C_k^{k-1})^{-1} + H_k^T V_k^{-1} H_k)^{-1}. \end{aligned} \tag{6.16}$$

Thus the inversion of H_k is not needed.
Since in the Kalman Filter a prediction is calculated where the next measurement is expected it cannot cope very well with detector inefficiencies. When measurements are missing the prediction is less accurate. In such a case the predictions needed to add a new hit are not precise enough and tracks cannot be found reliably.

6.4 Summary

In the last three sections some track finding methods were presented. There are two big groups of methods: local and global methods. Local methods are usually very fast methods. The combinatorial Kalman Filter has the additional advantage of doing the track finding (pattern recognition) and track fitting (parameter determination) at the same time. But they need the detector layout as input so that they know where the next measurement is expected. They also need seed tracks to start with and the result depends on the seed tracks. A correct seed track must be chosen to find the track reliably. Lastly local methods might have problems with detector inefficiencies (missing measurements). If a measurement is missing, the prediction of where the next measurement is expected is not accurate and the track might not be found.
Apart from local methods, a number of global methods are available. The simplest global method, the Template Matching [77, 78] is only useful for rather simple detector geometries and good hit efficiencies. Conformal Mapping [79] is independent of the detector layout but it only works for tracks coming from the vertex. Neural Networks [74] are also independent of the detector geometry, but these techniques favor straight tracks and often need a good initialization. The Radon transformation works for any track shape that can be described by one set of parameters (also for tracks with large curvatures) but the resolution of the detector needs to be known. The Hough transformation finally is a special case of the Radon transformation and neglects detector resolutions. In the Hough transformation not the full parameter space is calculated as is done in the Radon transform. The disadvantage is that, due to neglecting detector resolutions, the points describing a track in the calculated parameter space are not well defined.

6.5 Choice for Track Finding in Testbeam Data

A Hough transformation was implemented to find tracks in testbeam data taken with a TPC prototype. Curved tracks should be found, so an algorithm has to be chosen which can deal with such tracks reliably. Since it is a testbeam it is possible that parts of the readout do not work properly or that some measurement points do not look as expected and need to be cut out. This is a problem in general and not restricted to testbeam data, however, it is more likely to happen in testbeam data because usually the detector technology is still under development. Since the precision of the detector is high the detector resolution can be neglected. Thus out of the options available the Hough transformation is chosen since it promises to be the best compromise. The Hough transformation, however, has two disadvantages. Firstly, it is a global method and thus slower than local methods. The speed with which tracks can be found decreases with increasing number of measurements in an event. In testbeam data there are typically only a few measurements. Thus the computing time should be reasonably low. For testbeam data it is in any case more important to find tracks reliably (even in the presence of detector inefficiencies) than having a very fast algorithm. The speed becomes important when track finding needs to be done very fast. This will be the case in the ILD where an online event building is foreseen to mark “bunches of interest” which will then be used for physics analysis [27] as proposed in [87]. The other disadvantage of the Hough transformation is that if measurements are not perfectly on a track (which is the case in real data) the functions in parameter space will not intersect in exactly one point. This disadvantage is introduced by neglecting the detector resolution. However, a solution is discussed in Chapter 7.

Chapter 7

Pathfinder

Pathfinder (PAckage for Tracking with a Hough Trafo FINDER) is a software package which provides a global track finding algorithm based on a Hough transformation. This method can be used for any patterns which can be parametrized by one set of parameters. It is implemented to find tracks in 3D-space in data taken with the Large Prototype TPC [88]. The search is done in two projections, first in the xy-projection (readout plane) and then in the sz-projection (perpendicular to the readout plane, where s is the arc length of the track in the xy-projection). The package also includes a very simple track generation algorithm.
This chapter will first describe some of the mathematics used in the algorithm. After that the search algorithm will be described in detail followed by a presentation of the implemented classes of Pathfinder. Finally first tests of the algorithm will be presented.

7.1 Some Mathematics

The idea behind the Hough transformation is that for each hit all possible tracks are constructed on which the hit can be on. This is an infinite number of tracks. If several hits are on the same track there is one of the infinite number of tracks which is common to all hits. This is illustrated in Figure 7.1 for a straight line. The dots represent hits which are located on a straight line. For two hits some possible straight lines are shown on which the hit is on. There is one straight line common to all hits.
For straight lines in a two-dimensional plane two parameters need to be varied to construct all possible tracks and for circles in a two-dimensional plane three parameters are needed. For helices in a three-dimensional space (which is the expected track shape in the presence of a magnetic field) five parameters are needed. One possible set of helix parameters is presented in Chapter 5.2.1.1. Practically the search for helices has to be done in two steps: in the xy-projection and in the sz-projection. In order to calculate the arc length s the parameters describing the track in the xy-plane are needed. The magnetic field is chosen to be parallel to the z-axis. In this case the projection of the track into the xy-plane is a circle. If no magnetic field is present (or the particle which produced the track had high transverse momentum) the track is a straight line. These two shapes are implemented in Pathfinder. In

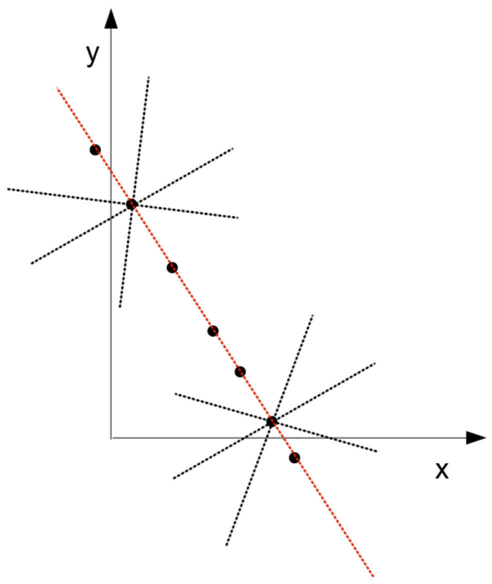


Figure 7.1: Explanation how the Hough transformation works, here in case for a straight line in a 2D-plane. The black dots represent points in 2D to which the Hough transformation is applied. For each point all possible straight lines crossing the point are calculated. For two points a few of these straight lines are shown (dashed black lines). If some points are on the same track, the same straight line can be found for these hits (dashed red line).

the sz-projection the track is always a straight line, no matter if the track is a helix or a straight line.
Thus only straight lines and circles need to be found with the Hough transformation. In the following the equations for these two cases are given.

7.1.1 Hough Transformation for Straight Lines

To describe a straight line in a 2D-plane two parameters are needed. Here the distance of closest approach d_0 is used as well as the angle between the positive x-axis and the direction of closest approach θ. A sketch of the parameters is shown in Figure 7.2. In this case the equation for a straight line reads

$$y(x) = -\frac{\cos\theta}{\sin\theta}x + \frac{d_0}{\sin\theta}. \tag{7.1}$$

This equation is transformed into a function $d_0(\theta)$ which describes all straight lines crossing the position of the hit (x, y).

$$d_0(\theta) = \cos\theta \cdot x + \sin\theta \cdot y. \tag{7.2}$$

For each hit one function $d_0(\theta)$ is calculated. If the hits are on a straight line these functions intersect and the point of intersection yields the parameters of the track. Since cosine and sine are periodic function (period 2π), the functions $d_0(\theta)$ are periodic (same period) and an infinite number of intersections exist. In each 2π wide range in θ two intersections exist, one with positive and one with negative d_0

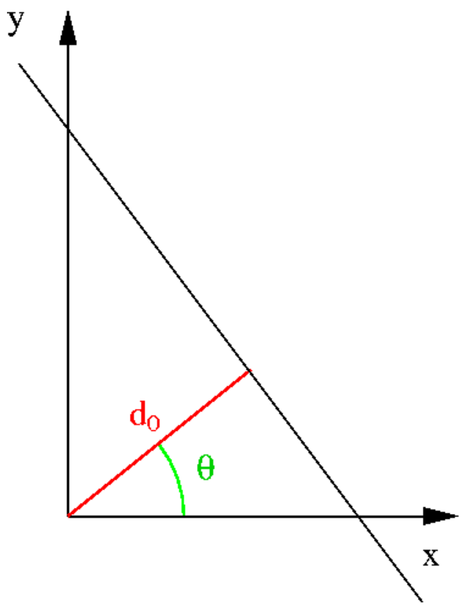


Figure 7.2: Parametrization of a straight line used in `Pathfinder`.

and separated by π. Both points of intersection are in principle describing the same track, only the direction of motion of the particle is different. At this stage of the reconstruction information about the direction the particle traveled is not available. θ can thus be limited to a π-wide range so that only one point of intersection is found. The track parameters can be corrected in a later stage of the reconstruction, if necessary.

7.1.2 Hough Transformation for Circles

The Hough Transformation for circles is split into two parts. First the search for the center of the circle is done. After that the radius of the circle is determined. The idea is depicted in Figure 7.3 and taken (with small modifications) from [86]. First a straight line is constructed through pairs of hits. Let the positions of these two hits be (x_1, y_1) and (x_2, y_2). The point which lies on the straight line half way between the two hits is calculated by the mean of the two x positions and the two y positions.

$$x_\mathrm{h} = \frac{x_1 + x_2}{2} \tag{7.3}$$

$$y_\mathrm{h} = \frac{y_1 + y_2}{2} \tag{7.4}$$

A straight line between the center of the circle (a, b) and x_h, y_h is perpendicular to the straight line between the two hits (x_1, y_1) and (x_2, y_2). The slope of the straight line between x_h, y_h and the center of the circle is

$$m = \frac{x_2 - x_1}{y_1 - y_2}. \tag{7.5}$$

The offset is given by

$$n = y_\mathrm{h} - m \cdot x_\mathrm{h}. \tag{7.6}$$

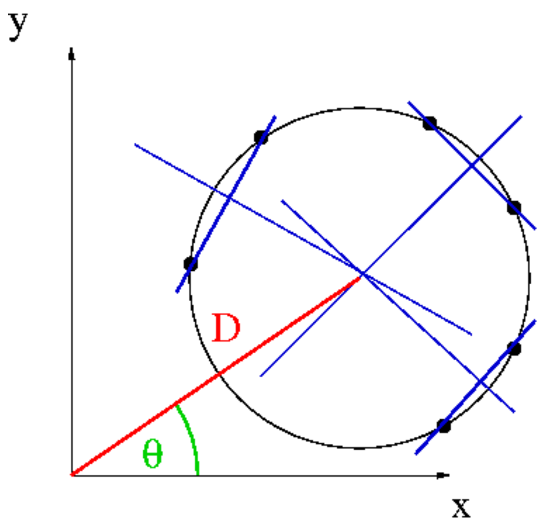


Figure 7.3: Finding the center of a circle. The hits (here: points in 2D-space) are represented as black dots. Between all pairs of hits (only a few combinations shown here) a straight line is constructed. A second straight line is constructed such that it is perpendicular to the first straight line and crosses it half way between the two hits. If the two hits are on the same circle the second straight line crosses the center of the circle.

Inserting this into the equation for straight lines one obtains the following function for the Hough transformation:

$$b(a) = \frac{x_2 - x_1}{y_1 - y_2} a + \frac{1}{2} \frac{(y_1^2 - y_2^2) + (x_1^2 - x_2^2)}{y_1 - y_2} \tag{7.7}$$

This function describes all possible position of the center of the circle. Instead of the center of the circle (a, b) the distance of the center of the circle from the origin D and the angle θ between the positive x-axis and the direction of D are chosen as parameters:

$$a = D \cdot \cos\theta \tag{7.8}$$
$$b = D \cdot \sin\theta. \tag{7.9}$$

This then leads to the following function:

$$D(\theta) = \frac{1}{2} \frac{(y_1^2 - y_2^2) + (x_1^2 - x_2^2)}{(y_1 - y_2)\sin\theta + (x_1 - x_2)\cos\theta} \tag{7.10}$$

Using this function in the track finding algorithm produced low track finding efficiencies for tracks with small curvatures which corresponds to large values for D. Additionally, this function has discontinuities in places which depend on three parameters: the differences of the hit positions and θ. With the inverse of this function

$$\frac{1}{D(\theta)} = 2 \cdot \frac{(y_1 - y_2)\sin\theta + (x_1 - x_2)\cos\theta}{(y_1^2 - y_2^2) + (x_1^2 - x_2^2)} \tag{7.11}$$

tracks with small curvatures can be found reliably. It has discontinuities as well, but they depend on the differences of the hit positions only and can thus be predicted

better. The denominator is zero only if the hit positions in the xy-plane are the same for both hits. This is only the case if the hits are identical (this will not happen because combinations of a hit with itself are not built) or in a curler where two hits in different turns have the same x and y position (this is unlikely to happen because in curlers the losses of energy have a big effect) or if two hits have identical x and y but are on different tracks (which is also unlikely to happen and is anyway not a combination of interest). In any of these cases, the function $\frac{1}{D(\theta)}$ has infinitely large values for any θ, because θ only appears in the enumerator. By limiting the allowed values for D these combinations are filtered out.
The function is now used for the Hough transformation. Similarly as for straight lines there is also an infinite amount of intersections for these functions because they are periodic. As for straight lines the range of θ can be limited to a range of width π so that only one point of intersection can be found.
Once the center of the circle is found the distance of the hits to the center of the circle can easily be calculated:

$$R = \sqrt{(a - x_{\text{hit}})^2 + (b - y_{\text{hit}})^2}. \tag{7.12}$$

The value calculated most often is the radius of the circle.
The advantage of this method (searching the center of the circle first) is that one function for each pair of hits is used. For more than three hits per event the number of functions $N_{fct} = \sum_{k=1}^{N_{hits}-1} (N_{hits} - k)$ is larger than the number of hits N_{hits} in the event. For this reason the points of intersection in the Hough space are much better defined.
However, this advantage turns into a disadvantage for events with many tracks and/or many noise hits. In such a case the number of functions from wrong hit combinations (hits not on the same track) increases faster then the number of functions from correct hit combinations (hits on the same track).
For example: Assuming three tracks with 222 hits each (typical ILD tracks) there are about two thirds of all functions coming from wrong hit combinations. The situation is getting even worse if more tracks are in the event and noise hits are included.
In the ILD TPC tracks are expected to start at the same point (the vertex). This information can be used in the track finding algorithm by introducing a vertex constraint. Instead of calculating functions for all hit combinations only functions for the combinations of one hit with the vertex are calculated. Since the vertex is expected to be on all tracks, the amount of correct functions (hits on same track) compared to the number of wrong functions (hits not on the same track) is significantly increased. For zero noise hits and all tracks coming from the vertex there are no wrong hit combinations. Furthermore, since it is no longer necessary to calculate all combinations, the algorithm is much faster. The drawback is that the introduction of a vertex constraint limits the algorithm to finding tracks coming from the vertex region only.

7.1.3 Track Fitting

Due to the binning which needs to be introduced for the Hough space the track parameters obtained are only a rough estimate. The estimate is not precise enough to assign hits to a track reliably. Thus a simple track fitting algorithm is implemented in `Pathfinder` which improves the estimate of the track parameters. The fitting algorithms for straight lines and circles are given in the following.

7.1.3.1 Straight Lines

The straight line fit is done with a linear regression. If the slope of the straight line is smaller than one in the Hough transformation, the following equations are used (distance of hits to the track in y is minimized):

$$m = \frac{n\sum_i x_i y_i - \sum_i x_i \sum_i y_i}{n\sum_i x_i^2 - (\sum_i x_i)^2} \tag{7.13}$$

$$b = \frac{\sum_i x_i^2 \sum_i y_i - \sum_i x_i \sum_i x_i y_i}{n\sum_i x_i^2 - (\sum_i x_i)^2} \tag{7.14}$$

If the slope is larger than one, the following equations are used (distance of hits to track in x is minimized):

$$m' = \frac{n\sum_i x_i y_i - \sum_i x_i \sum_i y_i}{(\sum_i y_i)^2 - n\sum_i y_i^2} \tag{7.15}$$

$$b' = \frac{\sum_i y_i \sum_i x_i y_i - \sum_i x_i \sum_i y_i^2}{(\sum_i y_i)^2 - n\sum_i y_i^2} \tag{7.16}$$

The conversion of m' and b' to m and b is

$$m = \frac{-1}{m'} \tag{7.17}$$

$$b = -mb' \tag{7.18}$$

The conversion to `LCIO` track parameters in xy is then

$$\phi_0 = \arctan(m) \tag{7.19}$$

$$= \arctan\left(\frac{-1}{m'}\right) - \frac{\pi}{2} \tag{7.20}$$

$$d_0 = \cos(\phi_0) \cdot b \tag{7.21}$$

$$= -\sin(\phi_0) \cdot b'. \tag{7.22}$$

For the track parameters describing the track in the sz-projection the conversion is

$$\tan\lambda = m \tag{7.23}$$

$$= -\frac{1}{m'} \tag{7.24}$$

$$z_0 = b \tag{7.25}$$

$$= \frac{b'}{m'} \tag{7.26}$$

7.1.3.2 Circles

For the circle fit a method is used as described in [89]. A short summary is given in the following.
As a first step the following weighted averages are calculated: $\langle x\rangle$, $\langle y\rangle$, $\langle x^2\rangle$, $\langle xy\rangle$, $\langle y^2\rangle$, $\langle xr^2\rangle$, $\langle yr^2\rangle$, $\langle r^2\rangle$, $\langle r^4\rangle$. Here x and y denote the hit positions in the xy-plane and r is calculated via

$$r = \sqrt{x^2 + y^2}. \tag{7.27}$$

For all hits a weight of 1 is used.
In the next step the following variables are defined:

$$\begin{aligned}
C_{xx} &= \langle x^2\rangle - \langle x\rangle^2 && (7.28)\\
C_{xy} &= \langle xy\rangle - \langle x\rangle\langle y\rangle && (7.29)\\
C_{yy} &= \langle y^2\rangle - \langle y\rangle^2 && (7.30)\\
C_{xr^2} &= \langle xr^2\rangle - \langle x\rangle\langle r^2\rangle && (7.31)\\
C_{yr^2} &= \langle yr^2\rangle - \langle y\rangle\langle r^2\rangle && (7.32)\\
C_{r^2r^2} &= \langle r^4\rangle - \langle r^2\rangle^2. && (7.33)
\end{aligned}$$

From this two new variables are calculated:

$$\begin{aligned}
q_1 &= C_{r^2r^2}C_{xy} - C_{xr^2}C_{yr^2} && (7.34)\\
q_2 &= C_{r^2r^2}\left(C_{xx} - C_{yy}\right) - C_{xr^2}^2 + C_{yr^2}^2. && (7.35)
\end{aligned}$$

With q_1 and q_2 the fitted ϕ_0 is calculated via

$$\phi_0 = \frac{1}{2}\arctan\left(\frac{2q_1}{q_2}\right). \tag{7.36}$$

Two more variables are defined as

$$\begin{aligned}
\kappa &= \frac{\sin\phi_0 C_{xr^2} - \cos\phi_0 C_{yr^2}}{C_{r^2r^2}} && (7.37)\\
\delta &= -\kappa\langle r^2\rangle + \sin\phi_0\langle x\rangle - \cos\phi_0\langle y\rangle. && (7.38)
\end{aligned}$$

From this the two parameters still missing can be calculated:

$$\begin{aligned}
\Omega &= \frac{2\kappa}{\sqrt{1-4\delta\kappa}} && (7.39)\\
d_0 &= \frac{2\delta}{1+\sqrt{1-4\delta\kappa}}. && (7.40)
\end{aligned}$$

As a last step d_0 has to be multiplied with -1 to get the correct sign since it is defined differently in [89] and in the `LCIO` track parameters.

7.2 The Search Algorithm

In real data the hits are not exactly on a straight line or helix but shifted off slightly due to field distortions and detector resolution effects. The functions presented in

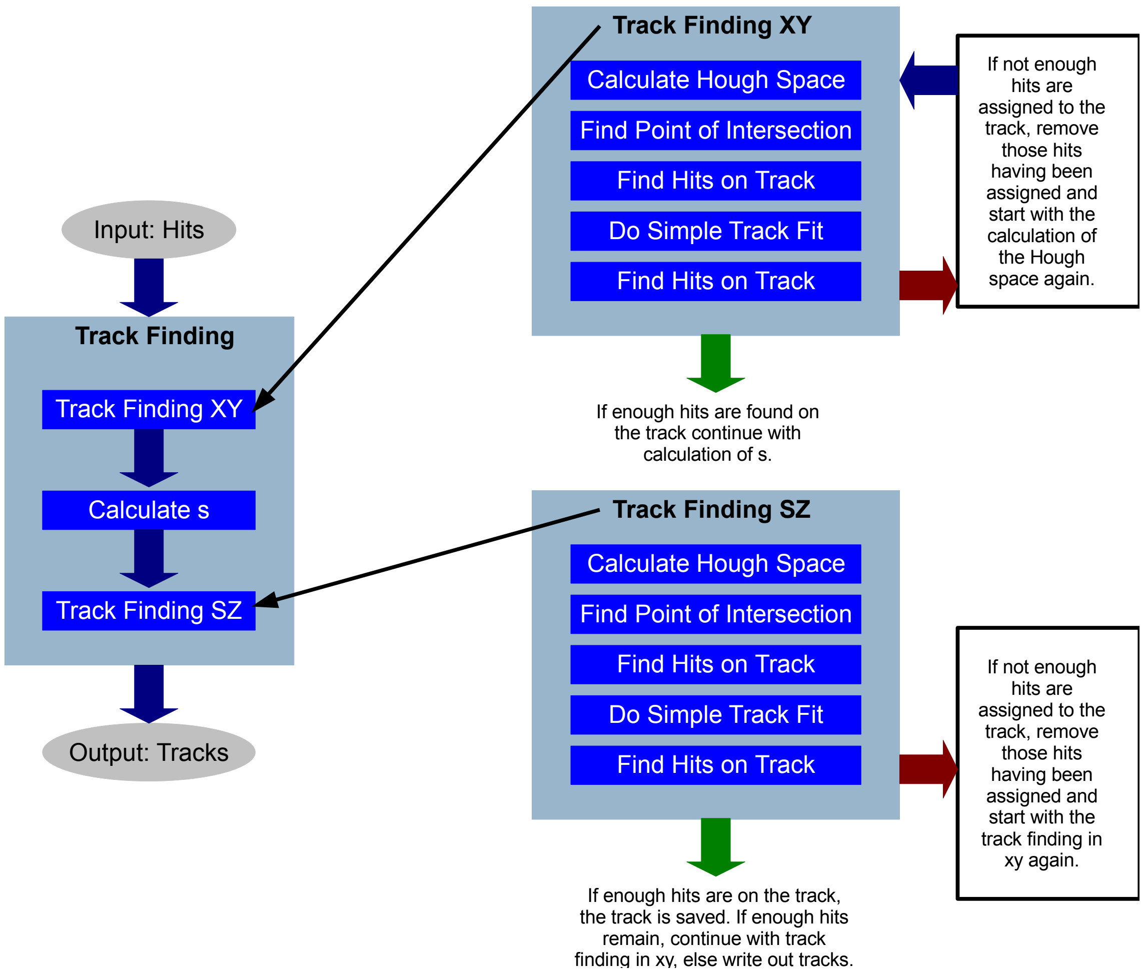

Input: Hits
Track Finding
Track Finding XY
Calculate s
Track Finding SZ
Output: Tracks
Track Finding XY
Calculate Hough Space
Find Point of Intersection
Find Hits on Track
Do Simple Track Fit
Find Hits on Track
If not enough hits are assigned to the track, remove those hits having been assigned and start with the calculation of the Hough space again.
If enough hits are found on the track continue with calculation of s.
Track Finding SZ
Calculate Hough Space
Find Point of Intersection
Find Hits on Track
Do Simple Track Fit
Find Hits on Track
If not enough hits are assigned to the track, remove those hits having been assigned and start with the track finding in xy again.
If enough hits are on the track, the track is saved. If enough hits remain, continue with track finding in xy, else write out tracks.

Figure 7.4: Track finding algorithm as implemented in `Pathfinder`.

the previous section do not intersect in exactly one point. However, there are regions where they approach each other. How one can anyway find tracks is explained in this section.
The algorithm is shown graphically in Figure 7.4. As already indicated the search is done in two steps: The search in xy and the search in sz.
First the search in the xy-plane is performed. The Hough space is calculated by choosing values for θ in a range between 0 and π. As discussed in Sections 7.1.1 and 7.1.2 this way only one point of intersection in the Hough space can be found. How many such values for θ are chosen depends on the number of bins given as steering parameter. Then for each of those values for θ and for each hit the values of the functions (7.2) or (7.11) (depending on the track shape to be found) are calculated. The result of this is a continuous spectrum of values. The values are then binned according to the steering parameters set (number of bins and range in that direction). It is counted how often each bin is hit by one of the functions[1].

[1]All hits are weighted with 1, but in principle it is also possible to weigh them with the error of the hit position or with the charge of the hit.

The point of intersection corresponds to the bin which is crossed most often. In other words the bin with the maximum number of entries has to be found. Since there might be ambiguities (bins with the same number of entries), an option is implemented to not only look for the bin with the maximum number of bins but also at the adjacent bins. The maximum bin gives a rough estimate of the track parameters. For circles the radius must be found in an extra step. Using these parameters one can determine which hits are on the track by calculating the shortest distance of the hit to the track and applying a cut on this value (which is specified in the steering parameters of `Pathfinder`).
To improve the track parameters a very simple fit is applied using the hits found on the track. The fitting methods are describe in Section 7.1.3. In this step the errors on the track parameters are calculated as well. Especially in the xy-plane the fit is important because the track parameters are used to calculate the arc length s for each hit which is needed to continue the search for tracks. If the track parameters are not precise enough the calculated s is wrong and thus tracks cannot be found in the sz-projection.
The determination of the hits being on the track is repeated with the fitted track parameters. In the steering the maximum allowed distance of the hit to the fitted track is specified. It can have a different value than the value used before the fit because the distance between hit and fitted track is expected to be smaller.
At this point the search in xy is complete. To do the search in the sz-plane s (the arc length between the point of closest approach (pca) and the hit) needs to be calculated for each hit. To calculate s the track parameters in the xy-plane are needed. It can thus only be done when the search in the xy-plane is complete. s is defined to be zero at the point of closest approach. This point can easily be calculated from the track parameters d_0 and ϕ_0 (see Appendix A). For each hit the distance between the point of closest approach and the hit position in xy along the track is calculated. The equations for the calculations are given in Section 7.3.2.2 and Appendix A.
After having done this the search in sz is performed in an analog way as in the xy-projection. Only those hits are taken into account which are assigned to a track in the xy-search. Finally, when the sz-search is complete, a track is built, which consist of the hits on the track and the rough estimation of the track parameters coming from the Hough transformation[2]. To match the parameters described in 5.2.1.1 the track parameters obtained by the Hough transformation are converted to `LCIO` parameters before saving the track.
The hits added to a track are removed and the search algorithm is started from the beginning with the remaining hits. This procedure is repeated until no more tracks can be found or all hits are assigned to tracks.

7.3 The Implementation of `Pathfinder`

`Pathfinder` comprises several classes. There are two groups of classes. The first group is visible to the outside for user interaction. These are the classes via which

[2]The parameters coming from the simple fit, to be precise.

the input for `Pathfinder` is specified (steering parameters and input data (hits)) and classes with which the output of `Pathfinder` is handled (tracks and track parameters). Example code explaining the usage of these classes can be found in Appendix B. The second group of classes are those classes needed by the actual track finding. They do not need to be known by the user. In this chapter all classes are presented shortly.

7.3.1 General Classes

General classes are those classes which are needed for both track finding and track fitting or which are visible for the user.

7.3.1.1 Steering

From the users point of view track finding starts with the steering parameters. For this purpose a class called `FinderParameter` is implemented. It is used to specify the search parameters. An overview of these parameters is given in the following.

- `_isStraightLine`, `_isHelix`: These steering parameters define what kind of track shape is to be found. For straight lines `_isStraightLine` has to be true, for helices `_isHelix` has to be true.
- `_findCurler`: `_findCurler` has to be true if curlers (helices with more than one turn) are to be found. It only has an effect if `_isHelix` = true. If it is set to false a curler will be found as several tracks, one track for each turn. For tracks with high energy losses the curler will not be found in one piece even if this parameter is set to true.
- `_minimumHitNumber`: Via this parameter the minimum number of hits on the track can be specified. If there are less hits on the track, the track is rejected and the hits on that track are counted as noise hits.
- `_maxXYDistance`, `_maxSZDistance`: The maximum allowed distance of the hits to the track in the xy-plane and the sz-plane before the fit can be set via these two parameters. They are needed to determine which hits are close enough to the track to be counted to be on the track.
- `_maxXYDistanceFit`, `_maxSZDistanceFit`: Basically these two parameters are the same as the previous ones but are used after the fit. Since the track parameters after the fit are a better estimate than the ones before the fit the values for the maximum allowed distance after the fit can usually be set lower than those used before the fit.
- `_numberXYDzeroBins`, `_numberXYThetaBins`, `_numberXYOmegaBins`, `_number-SZDzeroBins`, `_numberSZThetaBins`: These steering parameters set the binning of the Hough spaces. θ here means the angle between the direction to the point of closest approach and the x-axis. The parameters θ and d_0 are also used for the straight lines in the sz-plane instead of $\tan\lambda$ and z_0. The slope

and the offset can in principle have infinitely large values which is avoided by using θ and d_0 as parameters.
`_numberXYOmegaBins` only has an effect if `_isHelix` = true. The number of bins cannot be larger than 1000. How the values are chosen best depends on the data. If the bins are too wide, the track parameters are not precise enough for the following step (determination of hits close to the track). In this case no hits can be assigned to the track. If the binning is too small the point of intersection might not be defined well enough to be found correctly. Also, the more bins are chosen, the longer the computation takes.

- `_maxDxy`, `_maxDsz`: These are the maximum ranges of the Hough spaces and depend on the layout of the readout plane and the setup. The range is chosen symmetrically around zero. At this point the user needs to think about how the data and the setup the data were taken with look like. For straight lines `_maxDxy` is the maximum distance of closest approach possible so that the track is still visible on the readout plane. For circles it is the maximum possible distance of the center of the circle in the xy-plane so that the track could still be seen on the pad plane with a significant curvature[3] or the inverse of this[4]. `_maxDsz` is the maximum possible distance of closest approach in the sz-plane.

- `_useVertex`, `_VertexPosition`: The default of `_useVertex` is false. `_VertexPosition` is a pair giving the vertex position in the xy-plane. If it is set, `_useVertex` is set to true and the vertex constraint is used. If `_VertexPosition` is not set, the vertex constraint is not used. The vertex information is added by not using all combinations of hits (compare section 7.1) but combining every hit with the vertex position. This speeds up the computation but only tracks coming from the origin can be found. It only has an effect if `_isHelix` = true.

- `_searchNeighborhood`: This is an option to improve the search for the point of intersection. Instead of searching the bin with the maximum number of entries only, the surrounding area is taken into account as well. This option improves the track finding result in case for tracks with a small number of hits (order of ten) where the intersection in Hough space is not well defined.

- `_saveRootFile`: This is an option to save the Hough spaces in a `ROOT` Tree [90] (for debugging purposes only).

7.3.1.2 `basicHit` and `candidateHit`

The class called `basicHit` basically contains the x-, y- and z-position of a hit but also a covariance matrix and a flag. As input for the track finding algorithm a vector containing such hits is needed. The class `candidateHit` inherits from `basicHit` and additionally contains the s-value for each hit which is calculated after the track search in the xy-plane.

[3]`Pathfinder` versions v00-01-00, v00-01-01 and v00-02
[4]From `Pathfinder` version v00-03 on

7.3.1.3 Track Parameters

There are tree different classes of track parameters.
The class `TrackParameterXY` contains only those parameters that are needed to describe the track in the xy-projections. These parameters are ϕ_0, d_0 and Ω as defined in section 5.2.1.1.
The analog class for the track parameters in the sz-plane is called `TrackParameterSZ`. This class contains $\tan\lambda$ and z_0 as defined in in section 5.2.1.1.
Finally there is the class `TrackParameterFull` which inherits from `TrackParameterXY` and `TrackParameterSZ`. This class contains the full parameter set.

7.3.1.4 Track

The object which is returned after the whole process of track finding is a vector containing `TrackFinderTracks`. `TrackFinderTrack` contains the full parameter set of the track, those hits being found on the track and the rejected hits.
Since the track finding is an iterative process only the last track in the vector contains the hits not belonging to any track. For all other entries the rejected hits contain all hits having not been assigned to a track at the time the track was found (which means that some of the rejected hits might be assigned to a track in one of the next iterations of track finding).

7.3.1.5 `RootFileWriter`

There is a class called `RootTreeWriter` which writes the Hough spaces in a `ROOT` tree. This was mainly written for debugging. The Hough space is written only for the first processed event.

7.3.2 Track Finding

In this section the classes performing the track finding are presented.

7.3.2.1 Hough Space and Point of Intersection

To calculate the Hough space a container is needed which can hold all needed information. A `C++` [91] `std::map` was chosen. Each entry consists of a key and a value. In this case each key represents a position in Hough space and is a simple integer. The conversion from a point in Hough space (a pair of doubles) to an integer is done the following way: first the pair of doubles is converted to a pair of integers, each integer representing a bin number. Ten bits from another integer are used for each of the two integers calculated before to save them. This third integer is used as key for the map. This method limits the possible number of bins to 1024, however this is a binning which usually works fine and for finer binning the point of intersection is not well defined anymore. The value belonging to each key serves as a counter which counts how often a function crossed a certain region of the Hough space. Two different maps are needed, one for straight lines, one for circles.

- The class `HoughMap2Dint` for straight lines.

- The class `newHoughMap3Dint`[5] for circles without vertex constraint.
- The class `newHoughMap3DintVertex`[5] for circles with vertex constraint.

Each Hough map has a function called `fill` which calculates the Hough space by filling it with the functions 7.2 and 7.11 given in Section 7.1, depending on the track type given as steering parameter. Practically this is done by creating a list of all possible values for θ, using the binning given in the steering parameters. The range for θ is always between 0 and π. For each of these values d_0 and $1/D$ (or D in older versions) are calculated and the calculated value is rounded according to the binning given in the steering parameters. The pair of θ and d_0 or $1/D$ is then converted to an integer as described above and used as a key for the map. If the key exists in the map the value is counted up by one, if it does not exist it is added and the value is set to one.
The next step is to find the point of intersection of all functions, or, in other words, the maximum value in the map. This is done by a function called `findMaximum`. First the maximum value is found and then the map is searched for other entries with the same or a similar number of entries. This is needed to avoid binning effects. For example one peak can be distributed over various bins. To find the best maximum an option is implemented which does not compare the maxima only but also the surrounding area. This is helpful for tracks with a low number of hits (order of ten). It can be switched off because it does not improve the result for longer tracks with more hits. For the circle finding one more step has to be included at this point. Up to now only the center of the circle is known. The radius is determined by calculating the distance between each hit and the center of the circle. The value occurring most often is the radius. For each maximum the number of hits close to the track is counted. What ‘close’ means can be specified in the steering parameters. The maximum with the most hits close to it is accepted. Practically this is done by calculating the shortest distance between hit and track (see Appendix A on how the equations are derived). For straight lines this is

$$d = |x_{\text{hit}} \cdot \sin\phi_0 - y_{\text{hit}} \cdot \cos\phi_0 + d_0|. \tag{7.41}$$

For circles first the center of the circle is needed:

$$\begin{aligned} x_{\text{center}} &= (R - d_0) \cdot \sin\phi_0 & (7.42)\\ y_{\text{center}} &= -(R - d_0) \cdot \cos\phi_0. & (7.43) \end{aligned}$$

The distance between hit and track is then

$$d_{1,2} = | \pm \sqrt{(x_{\text{hit}} - x_{\text{center}})^2 + (y_{\text{hit}} - y_{\text{center}})^2} - R| \tag{7.44}$$

The smaller one of the two values is chosen.
The maximum of the Hough map then gives an estimate of the parameters for the track. The maximum for straight lines is stored in a class called `HoughMaximum-2D`. This class contains two doubles for the parameters. The class `HoughMaximum3D` inherits from `HoughMaximum2D` and has a third double for the additional parameter needed to describe circles.

[5]The name has historical reasons.

7.3.2.2 Calculate Arc Length s

After having done the search in the xy-plane the arc length s needs to be calculated for each hit. This is done in the class `candidateHitCombination`. From the track parameters obtained from the xy search the arc length s is calculated for each hit. This is the point where `basicHits` are transformed into `candidateHits`.
In order to calculate the arc length one first needs to calculate the position of the point of closest approach.

$$x_{\text{pca}} = -d_0 \cdot \sin\phi_0 \tag{7.45}$$
$$y_{\text{pca}} = d_0 \cdot \cos\phi_0. \tag{7.46}$$

For straight lines the arc length is calculated for each hit by

$$s = \sqrt{(x_{\text{hit}} - x_{\text{pca}})^2 + (y_{\text{hit}} - y_{\text{pca}})^2}. \tag{7.47}$$

Depending on which side of the point of closest approach the current hit is located, s has to be multiplied with -1. s is zero at the point of closest approach and is counted positive along the direction of motion of the particle. If a hit was produced before the particle crossed the point of closest approach it has a negative s, else the value for s is positive. This is needed to avoid kinks in the sz-projection.
For circles the arc length is calculated by

$$s = r \cdot \left(\arctan\left(\frac{y_{\text{pca}} - y_{\text{center}}}{x_{\text{pca}} - x_{\text{center}}}\right) - \arctan\left(\frac{y_{\text{hit}} - y_{\text{center}}}{x_{\text{hit}} - x_{\text{center}}}\right)\right). \tag{7.48}$$

where x_{center} and y_{center} are given in Equations 7.42 and 7.43. More details on how the equations are derived can be found in Appendix A.

7.3.2.3 Simple Track Fitting

The parameters given by the maxima of the Hough maps are only rough estimates. A simple fit can improve them. In the classes `XYLinearFit`, `XYCircularFit` and `SZFit` the fitting algorithms explained in section 7.1.3 are implemented. Also uncertainties on the track parameters are calculated. Output are `TrackParameterXY` or `Track-ParameterSZ`.

7.3.2.4 Track Finding

Finally the class `HoughTrafoTrackFinder` brings the whole track finding together. The central function in the class is called `find`. It provides a loop which ends if there are not enough hits left to build a track. In this loop the following steps are done:

- Search in xy with function `_doXYPlaneProjection`.
- Check if enough hits are found in xy to build a track. If there are enough hits, the search is continued in the sz-projection. If there are not enough hits on the track the hits are saved as noise hits.

- Calculate the arc length s for hits on the track in xy.
- Search in sz with function `_doSZPlaneProjection`.
- Check if enough hits are found in sz to build a hit. If there are enough hits the track is saved as `TrackFinderTrack`. If there are not enough hits on the track the hits are saved as noise hits.
- If a track is found the hits on the track are removed from the initial hits so that the search can be done again without the already found hits. The search is continued only if enough hits are left to build a track.

In the function `_doXYPlaneProjection` the following steps are done:

- Fill Hough map, depending on the track model, depending on the track model different maps are used.
- Find Maximum in Hough Map.
- Find hits close to the track described by the maximum by calculating the shortest distance between hit and track (See Equations 7.41 and 7.44).
- Do a simple fit.
- Find hits close to the track described by the fitted track. The equations 7.41 and 7.44 are used.
- Fill a vector with rejected hits on which the track finding can be run to find more tracks.
- Return track parameter in xy and the hits on the track in xy.

After s is calculated for the hits which are found to be on the track in the xy-projection the search in the sz-plane is done in the function `_doSZPlaneProjection`. This function basically performs the same steps as `_doXYPlaneProjection`. But in this projection only straight lines need to be taken into account. In the end the final hits on the track and the track parameter in sz are returned. When determining which hits are close to the track also a special treatment for curler is implemented. In the sz-plane curlers are no straight lines because s is reset after each turn. The reason for this is that s is defined as the arc length in the xy-plane. Hits in different turns can have the same value for s. This can be taken into account by adding integer multiples of the circumference of the circle to the value for s.
After the track finding is complete the parameters used in the Hough transformation are converted to `LCIO` track parameters (see Chapter 5.2.1.1). d_0 and Ω are the same in both definitions. The other parameters are converted with the following equations:

$$\phi_0 = \theta - \frac{\pi}{2} \tag{7.49}$$

$$\tan\lambda = -\frac{1}{\tan\theta} \tag{7.50}$$

$$z_0 = \frac{d_0}{\sin\theta}. \tag{7.51}$$

After all these steps are completed all tracks are found and one can proceed with the next reconstruction step: track fitting.

7.3.3 Track Generation

In order to test the track finding algorithm the class `TrackGenerator` was implemented. It is a very simple track generator which calculates hit positions along a trajectory. The track parameters are chosen randomly and uniformly distributed. To use this class three other classes are needed which are called `TGEventType`, `TG-DetectorType` and `TGParameterLimits`.
The class `TGEventType` contains all kinds of parameters describing how the event should look like:

- `nhits`: Number of hits per track.
- `ntracks`: Number of tracks in event.
- `nnoise`: Number of noise hits in event.
- `smearing`: Smearing is done by shifting the hits by Gaussian random numbers.
- `nevents`: Number of events. The loop over a number of events must be provided by an external program. The `TrackGenerator` uses nevents only as a seed for the random number generator[6].

The class `TGDetectorType` contains all parameters describing the detector in which tracks are to be generated (only rectangular pad geometries are available):

- `padplaneXmin`, `padplaneXmax`: x-range of pad plane. `padplaneXmax` has to be larger than `padplaneXmin`.
- `padplaneYmin`, `padplaneYmax`: y-range of pad plane. `padplaneYmax` has to be larger than `padplaneYmin`.
- `padplaneZmin`, `padplaneZmax`: z-range of the detector. `padplaneZmax` has to be larger than `padplaneZmin`.
- `padsizeY`: The length of a pad in y-direction. This is needed because the hit position is calculated such that the y-position is always the center of a pad.

With the class `TrackParameterLimits` limits on the parameter ranges for tracks are set. The definition of the track parameters is given in section 5.2.1.1.

- `minphi`, `maxphi`: Minimum and maximum ϕ_0.
- `mind0`, `maxd0`: Minimum and maximum d_0.
- `minr`, `maxr`: Minimum and maximum radius of the track $R = \frac{1}{|\Omega|}$.
- `mintanl`, `maxtanl`: Minimum and maximum $\tan\lambda$.

[6] This has historical reasons.

- `minz0`, `maxz0`: Minimum and maximum z_0.

To fix a parameter upper and lower limit need to have the same value. Apart from these parameters, also the track type needs to be defined. One can chose between straight lines, helix segments and curler.
The class `TrackGenerator` has private members `generatorTracks`, `hitsInEvent` and `noiseHits`. `generatorTracks` is a vector of `TrackFinderTrack` which is described in section 7.3.1.4. `hitsInEvent` and `noiseHits` are vectors of `basicHit` as described in section 7.3.1.2. These members are calculated by the function `generateTracks`. The track parameters are chosen randomly and uniformly in the given ranges. For each set of parameters the positions of the hits on this track are calculated. First the y-positions are chosen to be the center of the pads. With these positions and the track parameters first the x-positions and then the z-positions are calculated. To avoid having half tracks in the event, all tracks are rejected which contain hits not being in the detector volume. The track parameters together with the hits on this track are added to `generatorTracks`. Also the hits are added to `hitsInEvent`. As a last step, random noise hits are generated (y-positions is again the center of a pad) and added to `hitsInEvent` and `noiseHits`. In the end the `TrackGenerator` delivers a vector containing all simulated tracks and a vector containing all simulated noise hits as well as a vector with all hits in the event which is the input for the track finding. To check the distributions of the track parameters, `ROOT` histograms are filled and saved in a `ROOT` file.

7.3.4 `Pathfinder` Interface in `MarlinTPC`

In order to use `Pathfinder` to reconstruct tracks an interface is needed. This is provided by a `Marlin` processor which is part of `MarlinTPC` (see Chapter 5.2.5) and is called `PathFinderInterfaceProcessor`. It takes a collection of `LCIO TrackerHits` as input and gives produces a collection of `LCIO Tracks` and a collection of `TrackerHits` that are not assigned to a track. The steering parameters of this processor are listed in Table 7.1.

7.4 `Pathfinder` Design Choices

`Pathfinder` is the result of the development of an algorithm to find tracks. During the development several challenges had to be faced. In the following the most important ones will be described together with the solution implemented in `Pathfinder` and the limits introduced by the solution. The first challenge was to find a suitable container to hold the Hough spaces calculated in the algorithm. The second challenge was to improve the algorithm in terms of speed. Thirdly the track finding efficiencies needed to be improved to a reasonable value.

7.4.1 Finding a Suitable Container

Intuitively the logic choice for a container to hold the Hough space is a histogram with θ, d_0 (and Ω) on the axis. In each bin an entry is added when it is crossed by

Steering Parameter	**Description**
Mandatory Parameters:	
`TrackModel`	Specify the track model (straight line, helix, curler).
`MaxDxy`	Range of Hough space in xy.
`XYThetaBins`	Number of θ bins in xy.
`XYDZeroBins`	Number of d_0 or $\frac{1}{D}$ bins in xy (depending on track model).
`XYOmegaBins`	Number of Ω bins in xy.
`XYDistance`	Maximum distance between hit and track in xy.
`MaxDsz`	Range of Hough space in sz.
`SZThetaBins`	Number of θ bins in sz.
`SZDZeroBins`	Number of d_0 bins in sz.
`SZDistance`	Maximum distance between hit and track in sz.
`MinHitNumber`	Minimum number of hits on a track.
`ShiftHits`	If true hits are shifted closer to the reference point (default (0,0,0)) before track finding.
`SearchNeighborhood`	If true the neighborhood of a maximum in Hough space is taken into account.
`SaveRootFile`	If true Hough space is written to a `ROOT` file.
Optional Parameters:	
`ReferencePointXOverride`	Specify reference point to which hits are shifted.
`ReferencePointYOverride`	Specify reference point to which hits are shifted.
`ReferencePointZOverride`	Specify reference point to which hits are shifted.
`XYDistanceFit`	Maximum distance between hit and track in xy after the fit.
`SZDistanceFit`	Maximum distance between hit and track in sz after the fit.
`VertexPositionX`	Set vertex position to use vertex constraint.
`VertexPositionY`	Set vertex position to use vertex constraint.

Table 7.1: Steering parameters of the `PathFinderInterfaceProcessor`.

a function. The bin with most entries corresponds to the point where all functions intersect. However, memory problems occur especially for fine binning. The reason is that in a histogram all bins exist, even if the bin was not crossed by any function. The solution is to not use histograms to save the Hough space but maps [91]. A map is a container which consists of pairs of a key and a value. Values stored in the map can easily be accessed via the key. As keys the position in the Hough space is used and the value counts how often the point in Hough space is crossed by a function. An entry in the map is created only if at least one function crossed the point.

The drawback of maps is that whenever a new entry is created the map is sorted by keys. This sorting process is slow. The choice of keys has a large influence on how fast the map can be sorted. This topic will be covered in 7.4.2.

7.4.2 Decreasing Computing Time

As discussed in Section 7.4.1 a map is chosen as a suitable container to hold the Hough space. The disadvantage is that the map is sorted by keys when a new entry is inserted which is the most time-consuming part of the whole track finding algorithm. However, the time needed to calculate the Hough space can be changed by choosing different keys. Three different keys for the maps are compared in the following. The most intuitive one is to chose pairs of doubles (one for θ and one for d_0) for straight lines and triple of doubles for circles (one for θ, one for d_0 and one for Ω)[7]. This version is the slowest. Instead of pairs and triple of doubles pairs and triple of integers can be used. In this case θ, d_0 and Ω are converted to bin numbers (which are between zero and the maximum number of bins). Maps with these two keys are compared in Figure 7.5. It shows a significant decrease in computing time by more than a factor of three.
The third key created is an integer. Integers have, depending on the system, 32 or 64 bits. In `Pathfinder` ten bits are used for each direction to save the bin numbers in θ, d_0 and Ω-direction. The pairs and triple of integers are thus converted to integers. The comparison of computing time between triple of integers and integers as keys are shown in Figure 7.6. Calculating the Hough space with maps using integers as keys is about three times faster than when using triple of integers. However, since only ten bits are available per direction in this version, the maximum number of bins is limited to 1024 for each direction. Despite this limitation integer keys are chosen in the end. 1000 bins per direction proved to work so that tracks can be found reliably.
Another approach was followed to decrease the computing time. The idea was to use an adaptive Hough transformation [86] which starts with a coarse binning and calculates only those regions of the Hough space with a fine binning which most likely contain a point of intersection. This saves computing time (as is shown in Section 7.5), but, however, other technical difficulties occurred and this solution was abandoned.

7.4.3 Improvement of Track Finding Efficiencies

For straight lines the Hough transformation is done by starting with the function for straight lines $y(x)$ with parameters d_0 and θ and transforming this function to a function $d_0(\theta)$ with parameters x and y. For circles an analog way was chosen. The equation for circles is

$$\left(\frac{1}{\Omega}\right)^2 = (x-a)^2 + (y-b)^2 \tag{7.52}$$

[7]In one of the first versions of `Pathfinder` a function $\Omega(d_0, \theta)$ was used to calculate the Hough space for circles, see Section 7.4.3.

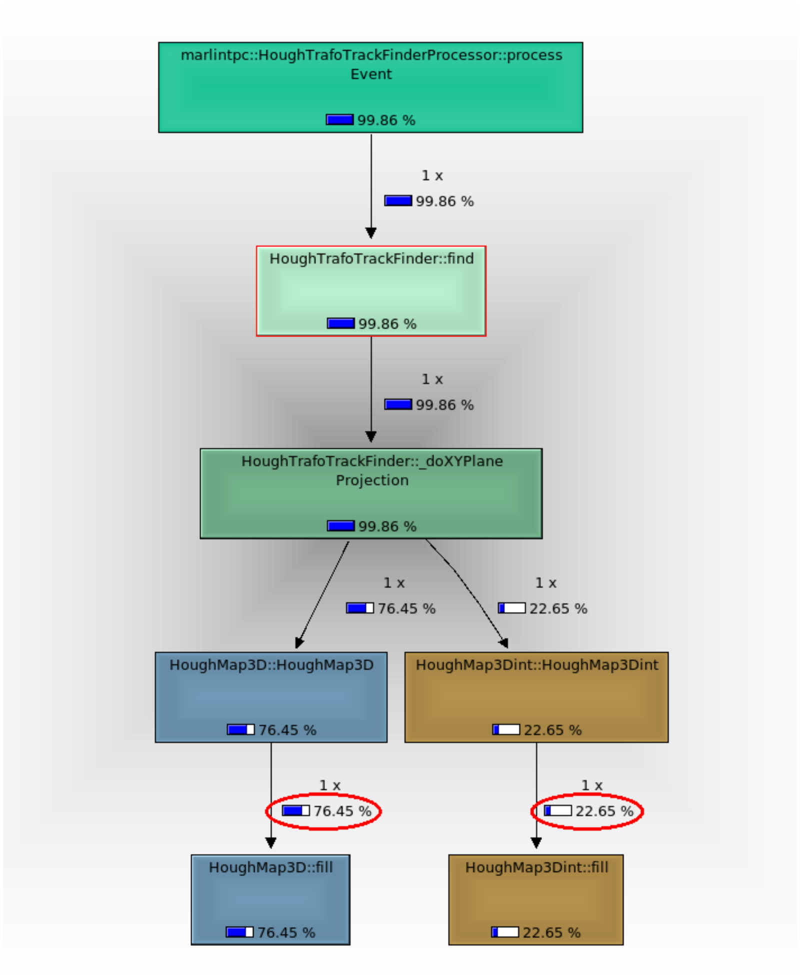


Figure 7.5: Profiling result for a 3D-Hough space using triple of doubles (left column) and triple of integers (right column). The map with triple integer keys uses less computing time than the one with double keys.

where (a, b) denotes the center of the circle. The center of the circle can be expressed by the parameters Ω, d_0 and θ:

$$a = (\frac{1}{\Omega} + d_0) \cdot \cos\theta \tag{7.53}$$

$$b = (\frac{1}{\Omega} + d_0) \cdot \sin\theta \tag{7.54}$$

Where Ω is the curvature, d_0 is the distance of closest approach and θ is the angle between the x-axis and the distance of closest approach. With these three equations one gets a function for the Hough transformation, which reads:

$$\Omega(d_0, \theta) = \frac{2 \cdot x \cdot \cos\theta + y \cdot \sin\theta - d_0}{x^2 + y^2 + d_0^2 - 2d_0 \cdot (x \cdot \cos\theta + y \cdot \sin\theta)}. \tag{7.55}$$

Drawing the functions for different hits reveals that they have similar values for large regions. It is thus difficult to find the point of intersection which results in low track finding efficiencies. Thus this method does not work well for circles. To overcome

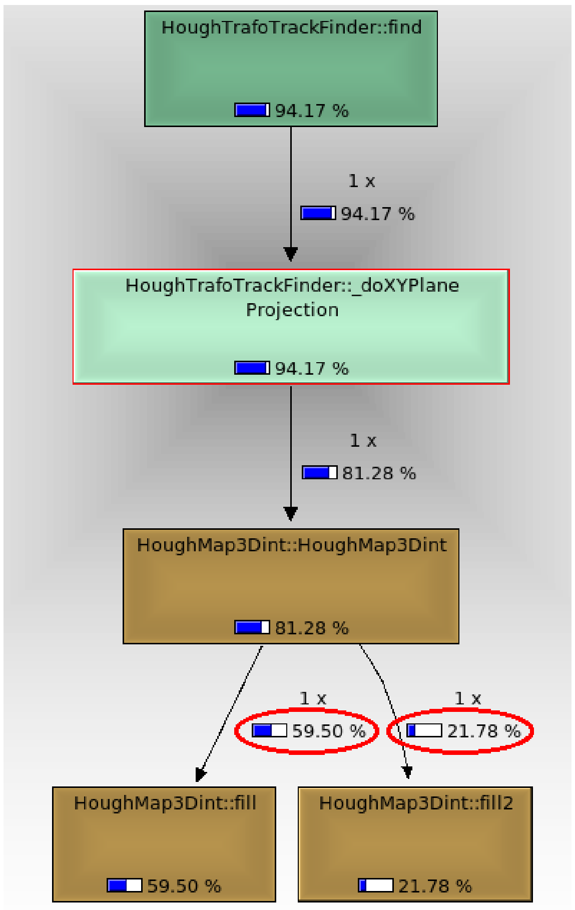


Figure 7.6: Profiling result for a 3D-Hough space using triple of integers (left column) and integers (right column) as keys. The map with integer keys uses less computing time.

this problem a different approach is followed, which is described in Section 7.1.2. The search for circles is split in two steps. First the center of the circle is searched (2-dimensional Hough transformation, parametrized by the distance between the center of the circle and the origin and θ) and after that the radius is determined (1-dimensional). With this method tracks can be found reliably. However, this method has two drawbacks. The first disadvantage is that pairs of hits are needed for one function. In environments with many tracks and noise hits combinations of hits not being on the same tracks can cause difficulties which can be overcome by introducing a vertex constraint. The second disadvantage is that the track finding efficiency drops for tracks with large momenta (which corresponds to large radii and thus to large distances between the origin and the center of the circle). This disadvantage can be overcome by using the inverse of the distance between the center of the circle and the origin instead of the distance itself, which is the solution implemented in `Pathfinder`.

7.5 Estimation of Computing Time

The part of the algorithm implemented in `Pathfinder` which takes longest is calculating the Hough space. To get an estimate on how the needed computing time evolves with number of tracks and number of noise hits it is counted how often the access operator of the maps is called while calculating the Hough space. Let N_T be

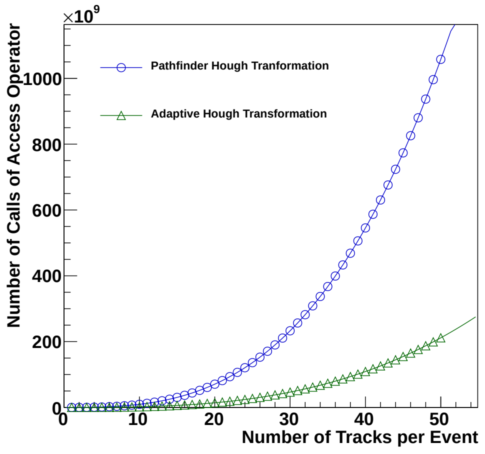


Figure 7.7: Number of calls of the access operator for different track multiplicities and for two different algorithms without vertex constraint.

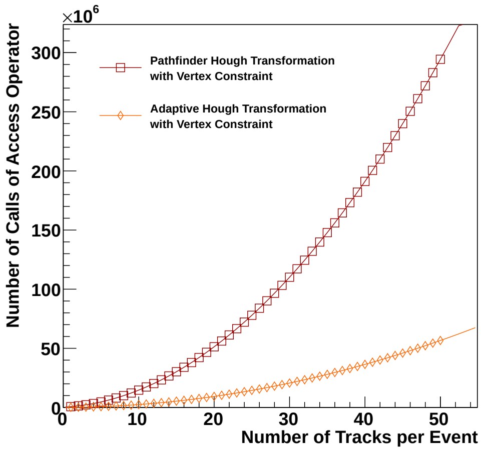


Figure 7.8: Number of calls of the access operator for different track multiplicities and for two different algorithms with vertex constraint.

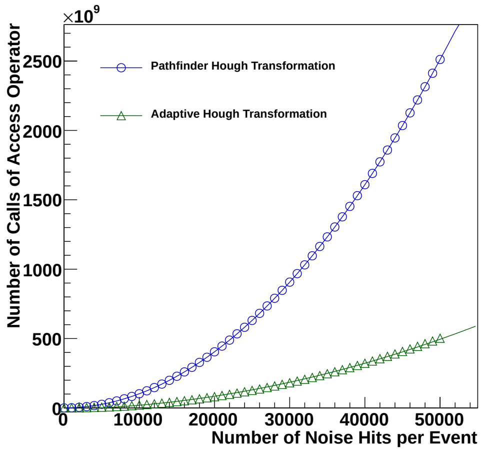


Figure 7.9: Number of calls of the access operator for different number of noise hits and for two different algorithms without vertex constraint.

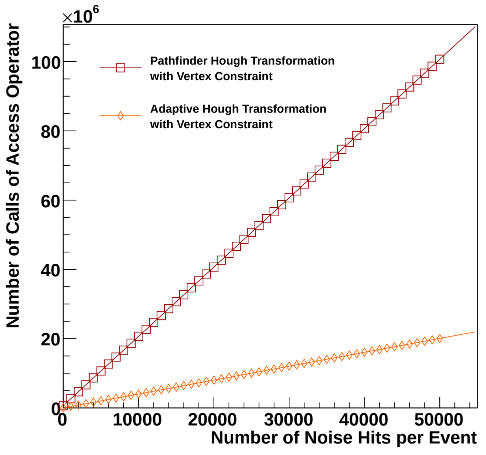


Figure 7.10: Number of calls of the access operator for different number of noise hits and for two different algorithms with vertex constraint.

the number of tracks per event, N_H the number of hits per track, N_N the number of noise hits, and N_B the number of bins in θ-direction. In case a vertex constraint is used or straight lines are to be found for each hit and each bin a value is calculated and added to the Hough space in xy. The search is continued in sz for those hits, that have been assigned to the track. After that the hits assigned to the track are removed and the search is done again in xy and sz until no more tracks can be found. Thus the number of calls of the access operator N_{calls} is given by

$$N_{\text{calls}} = \sum_{t=0}^{N_T} \left(\left(t \cdot N_H + N_N \right) N_B + N_H N_B \right) . \tag{7.56}$$

$N_{\text{calls}}(N_T)$ has the form of a polynomial of order $\mathcal{O}(2)$ while $N_{\text{calls}}(N_N)$ is a polynomial of order $\mathcal{O}(1)$. The functions $N_{\text{calls}}(N_T)_{N_N=0}$ and $N_{\text{calls}}(N_N)_{N_T=1}$ with $N_B = 1000$ and $N_H = 222$ (corresponds to a typical ILD event) are shown in Figure 7.8 and 7.10, respectively. `Pathfinder` needs about 2.5 seconds (on an average computer) to find one ILD-like track in one event when using the vertex constraint. Finding ten ILD-like tracks in one event is thus expected to take about one minute. An event with 50 tracks (for example $t\bar{t}$ events, no noise included) would take about 18 minutes and an event with 100 tracks would take about 1 hour.
In case no vertex constraint is used the situation gets more complicated because pairs of hits are built to calculate the Hough space. In this case the number of calls of the access operator is given by

$$N_{\text{calls}} = \sum_{t=0}^{N_T} \left(\frac{1}{2} \left(t \cdot N_H + N_N \right) \left(t \cdot N_H + N_N - 1 \right) N_B + N_H N_B \right) . \tag{7.57}$$

In this case $N_{\text{calls}}(N_T)$ is a polynomial of order $\mathcal{O}(3)$ and $N_{\text{calls}}(N_N)$ is of order $\mathcal{O}(2)$. The functions $N_{\text{calls}}(N_T)_{N_N=0}$ and $N_{\text{calls}}(N_N)_{N_T=1}$ for this case are shown in Figures 7.7 and 7.9 ($N_B = 1000$ and $N_H = 222$).
In the Figures the functions for the Hough transformation as implemented in `Pathfinder` are compared to the number of calls of the access operator in an adaptive Hough transformation [86] in which less bins are used and the regions which most likely contain a point of intersection are recalculated. In this case N_B in Equations 7.56 and 7.57 is replaced by $N_B' \cdot N_I$ where $N_B' \ll N_B$ is the number of bins and N_I is the number of iterations. Here N_B' is set to 10. In this case two iterations are needed to achieve the same precision as with the Hough transformation implemented in `Pathfinder`. The order of the polynomials describing the number of calls stays the same. However, since fewer bins need to be calculated the adaptive Hough transformation promises to be faster. Based on the estimate given before the adaptive Hough transformation would be faster by a factor of 50 for an event with one ILD-like track, about 0.05 seconds. Running the adaptive Hough transformation for one event with ten, 50 and 100 tracks would take about 3 seconds, 1 minute and 4 minutes, so well below one hour.
The computing time could be decreased further by calculating the Hough space once and finding all points of intersection in it. Thus the iteration is not needed and the order of the polynomials describing the number of calls for different N_T is reduced by one. Another possibility is to calculate each intersection between two functions analytically and fill only those points into the Hough space. On the one hand this introduces one additional loop over the hits and thus the order of the polynomials is increased but on the other hand the functions given in Equations 7.56 and 7.57

do not depend on the number of bins because the Hough space is not calculated for each possible value for θ. Additionally the map holding the Hough space is smaller. For this reason finding the intersection is expected to be faster.
These studies are only a rough estimate of the computing time. The algorithm implemented in `Pathfinder` does not only consist of calculating the Hough space but many other steps are needed which were described previously in this chapter. Also the time needed to access an element of a map depends on the number of entries it has. Both is not taken into account here. Thus the estimate is expected to be the lower limit.

7.6 Track Finding Efficiency Studies with `Pathfinder`

A variety of systematic track finding efficiency studies with tracks generated by the simple track generator implemented in `Pathfinder` are presented in the following. The studies show that the algorithm implemented in `Pathfinder` can find tracks reliably and that any inefficiencies are understood. Studies on realistic tracks are presented in Chapter 8.
For the studies random tracks are created with parameters in the ranges given in Table 7.2. Hit positions are calculated according to the chosen parameters. A rectangular pad plane with 200 pad rows is assumed. Thus for straight lines 200 hits are simulated per track. For helices, depending on the radius of the circle, up to 400 hits are created per track. The y positions are chosen such that they are in the center of the pad rows, the other positions are calculated according to the track parameters, as is sketched in Figure 7.11. For helices only one loop is simulated.
The simulated hits are then used as input for the Hough transformation. The steering parameters for the Hough transformation are listed in Table 7.3. If not stated otherwise these steering parameters are used.

Track Parameter	**Lower Bound**	**Upper Bound**
ϕ_0	$-\pi$	π
d_0	-300	300
Ω	-0.001	0.001
R	-1000	1000
$\tan\lambda$	-2	2
λ	-1.107	1.107
z_0	-300	300

Table 7.2: Parameter ranges of simulated tracks.

7.6.1 Parameter Scans

First it is investigated if there are any regions of the parameter space for which the track finding algorithm does not work properly. To investigate this 1000 events with one track each are generated. One parameter is fixed to a certain value while the

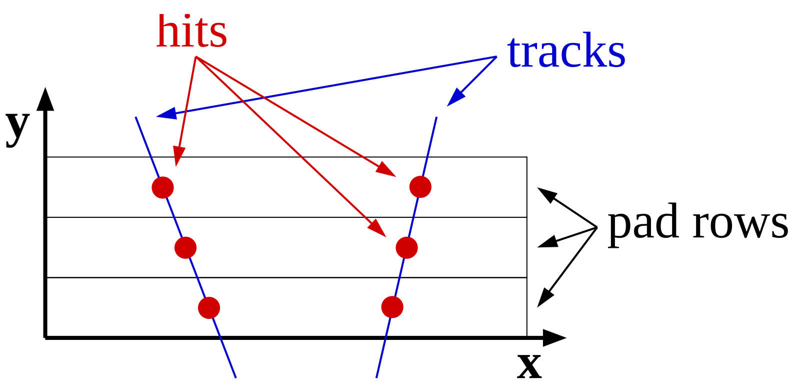


Figure 7.11: Sketch on how the simplified track generation implemented in `Pathfinder` works. The track parameters are chosen randomly in the given ranges. The hits are then distributed along the track. The y-psoitions are chosen to be in the center of the pad rows, the x psoitions are calculated from the y positions and the track parameters.

others are chosen randomly in the ranges given in Table 7.2. The study is done for straight lines and helices separately. The track finding efficiency is defined as the ratio between the number of correctly found tracks and the number of simulated tracks. A track is defined to be found correctly if all the hits on the simulated track are also on the found track. This is a strict requirement but since the hits are created exactly on a straight line or helix and no smearing is applied it is expected that almost all tracks are found correctly (which means that all hits are assigned to the track correctly). In more realistic data (including smearing, noise hits and multiple tracks) a looser requirement is more appropriate.

7.6.1.1 Straight Lines

The results of the parameter scans for straight lines are shown in Figures 7.12 to 7.15. In Figure 7.15 λ is plotted instead of $\tan\lambda$. λ is more intuitive because it is the polar angle of the track. The triangles represent the efficiencies without any cuts on track parameter ranges. In most parameter ranges the efficiencies are equally high at about 99.5 %. The only exceptions are $\phi_0 = \pm\pi, 0$ and $\lambda = \pm\frac{\pi}{2}$. Cutting out a region (about 0.18 rad wide) symmetricaly around $\phi_0 = \pm\pi, 0$ leads to efficiencies of 100 % and are shown as circles in figures 7.12 to 7.15. A cut on λ is not needed because tracks in the critical λ range are only simulated for the λ scan. In the other scans λ was chosen between -1.107 and 1.107.
Tracks with $\phi_0 = \pm\pi, 0$ are tracks which are parallel to the x-axis, which means that the track is parallel to a pad row as shown in Figure 7.16. Such tracks look like one single hit on the pad row and a track cannot be found. To check that this hypothesis is true and this is not a feature in the code which shows its influence for these parameter ranges only the tracks are rotated around the z-axis by $\frac{\pi}{2}$ and the reconstruction is run again. As shown in Figures 7.17 now for $\phi_0 = \pm\pi, 0$ the efficiencies are 100 % while it is zero for $\phi_0 = \pm\frac{\pi}{2}$ which now correspond to tracks running along one pad row. The results for the other parameter scans (not shown) are very similar to those shown before.

Steering Parameter	**Value**
minimum Number of hits on track	5
maximum distance hit - track (xy)	10
maximum distance hit - track (sz)	10
number of bins θ_{xy}, straight lines	500
number of bins θ_{xy}, circles	300
number of bins $d_{0,xy}$, straight lines	500
number of bins $d_{0,xy}$, circles	300
number of bins Ω_{xy}	300
number of bins θ_{sz}, straight lines	1000
number of bins θ_{sz}, circles	1000
number of bins $d_{0,sz}$, straight lines	1000
number of bins $d_{0,sz}$, circles	1000

Table 7.3: Steering parameters used for the Hough transformation. The parameters used for the Hough spaces and the LCIO track parameters are related in the following way: $\theta_{xy} \mathrel{\widehat{=}} \phi_0$, $d_{0,xy} \mathrel{\widehat{=}} d_0$, $\Omega_{xy} \mathrel{\widehat{=}} \Omega$, $\theta_{sz} \mathrel{\widehat{=}} \lambda$, $d_{0,sz} \mathrel{\widehat{=}} z_0$.

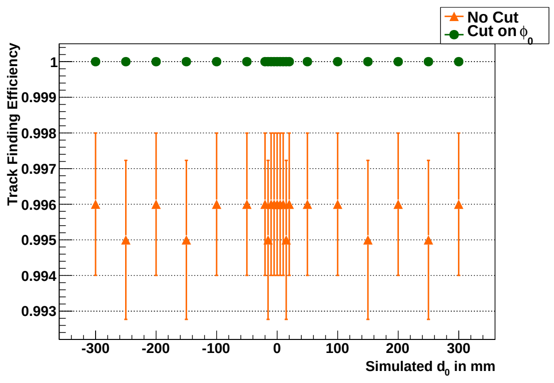


Figure 7.12: Parameter scan d_0 (straight lines).

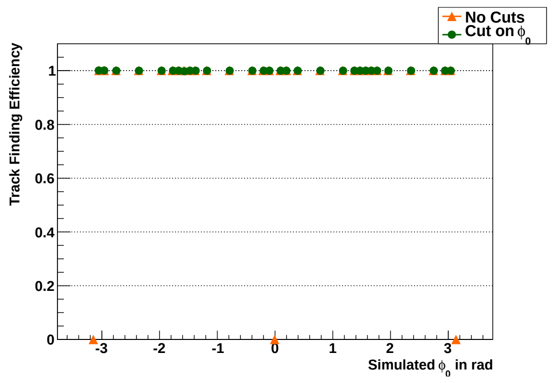


Figure 7.13: Parameter scan ϕ_0 (straight lines).

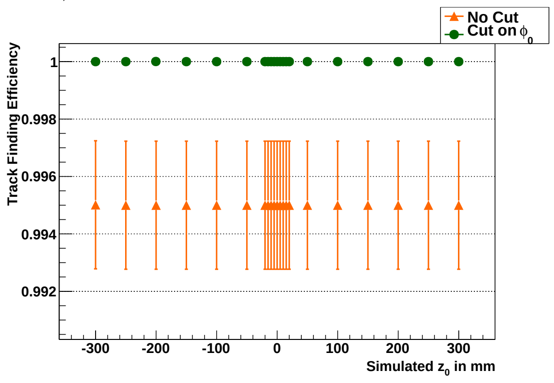


Figure 7.14: Parameter scan z_0 (straight lines).

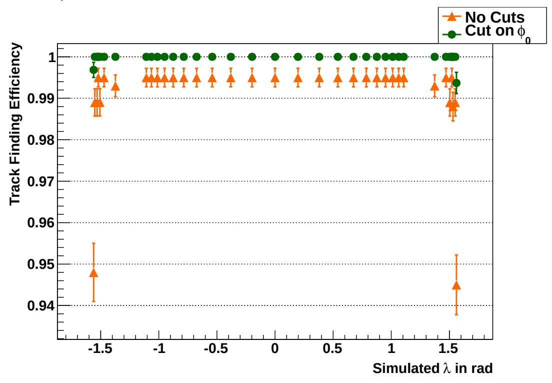


Figure 7.15: Parameter scan λ (straight lines).

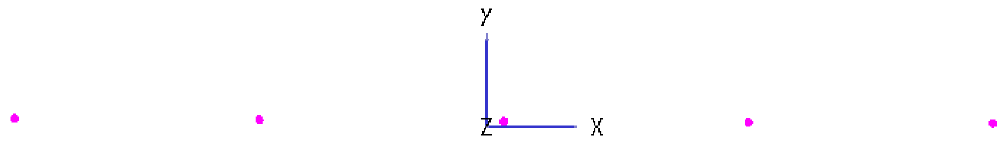


Figure 7.16: Example track with $\phi_0 = 0$.

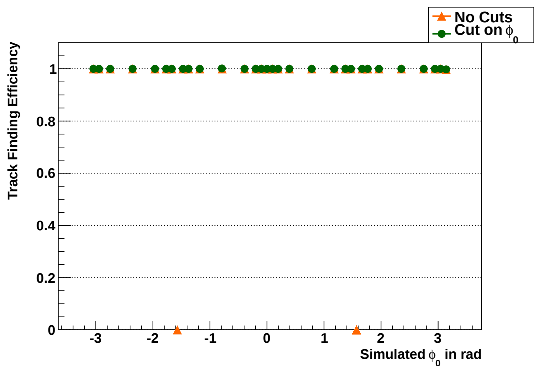


Figure 7.17: Parameter scan ϕ_0 (straight lines) for rotated track.

Tracks with $\lambda = \pm\frac{\pi}{2}$ are such tracks that are parallel to the z-axis. These tracks would have ended up on one single pad and would produce only one hit. Such tracks cannot be found with this algorithm.

7.6.1.2 Helices

The results of the parameter scans for helices are shown in Figures 7.18 to 7.22. The triangles represent the efficiencies without any cuts. Mostly the efficiencies are equally high at about 99 %. There are no structures visible that show low track finding efficiency for certain parameter regions. To get an idea why the efficiencies are still well below 100 % the number of hits on the simulated tracks are plotted for tracks that are not found correctly. The result is shown in Figure 7.23. Most of the unfound tracks are comprised of less than 5 hits. These tracks do not meet the requirement given as steering parameter (at least 5 hits must be on the track). Such tracks should not be included in the efficiency calculation. Cutting out these tracks leads to the efficiencies shown by the dots in Figures 7.18 to 7.22. This cut is in principle a cut on tracks with very small radii. An example for a track with four hits is shown in Figure 7.24. Small radii are not simulated in the R-scan so no drop in efficiency can be seen in the corresponding plot. Still the efficiency is not at 100 % in all parameter ranges which would be expected since perfect helices are investigated. The reason for this is, that, compared to straight lines, helices are more complex objects forwhich one additional parameter must be estimated. For helices (to be precise for the circle in the xy-plane) the search is done in two steps (for straight lines in xy only one step is needed). Thus there is one more step for helices where binning effects can be introduced. To find more tracks correctly the definition for correctly found tracks must be chosen less strict.

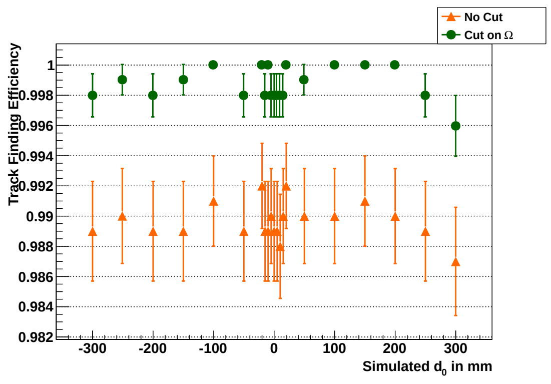


Figure 7.18: Parameter scan d_0 (helices).

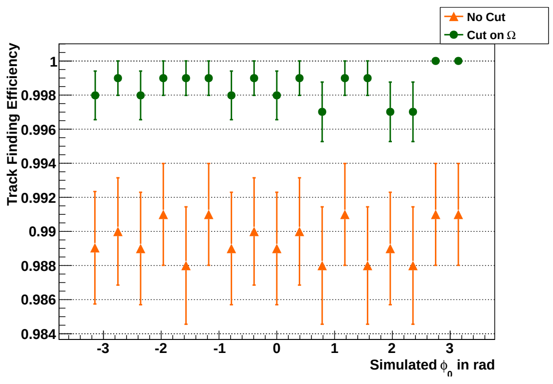


Figure 7.19: Parameter scan ϕ_0 (helices).

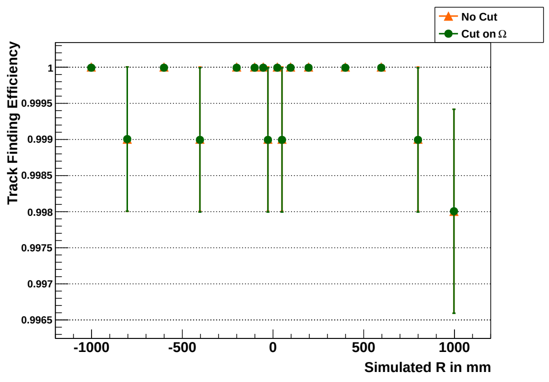


Figure 7.20: Parameter scan R (helices).

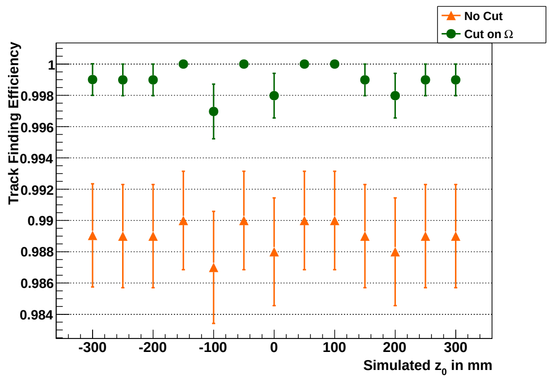


Figure 7.21: Parameter scan z_0 (helices).

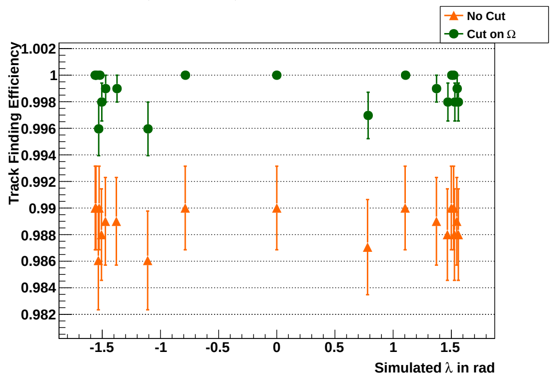


Figure 7.22: Parameter scan λ (helices).

7.6.1.3 Parameter Reconstruction

In the previous sections it was shown that for most track parameter ranges tracks can be found reliably. Now it will be investigated how well the track parameters can be reconstructed by the algorithm. The same simulation and the same definition of efficiency is used as before. The difference between the reconstructed track parameters and the simulated track parameters are plotted for all simulated parameter ranges. However, differences are expected because no proper track fit is applied and the parameters from the Hough transformation are only a first estimate.

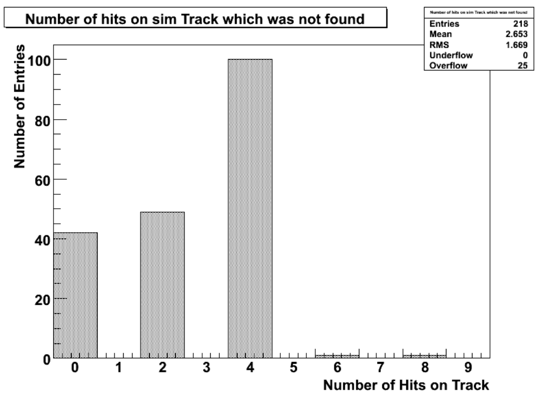


Figure 7.23: Number of hits on simulated tracks which were not found by `Pathfinder`.

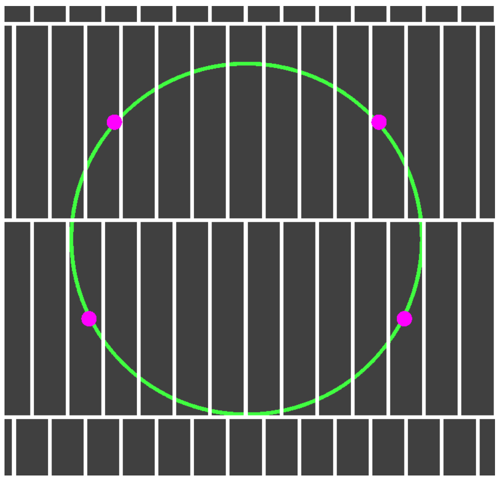


Figure 7.24: Example track with 4 hits on a virtual pad plane.

Figures 7.25 through 7.28 show how well the track parameters for straight lines can be reconstructed. The mean deviation of the reconstructed track parameters from the simulated value is plotted against the simulated value. The error bars represent the standard deviation. The differences between the reconstructed and the simulated track parameter are all in agreement with zero. For d_0 and z_0 the differences tend to be larger for large simulated values. In such events the positions of the hits are rather far away from the origin of the coordinate system. In such cases the track finding algorithm suffers from numerical uncertainties. This will be shown in Chapter 9.3. They can, however, be circumvented by shifting the hits closer to the origin before running the track finding algorithm. For λ the largest deviations occur for such tracks that are almost parallel to the drift direction and thus difficult to find with the algorithm, but overall λ is reconstructed well. ϕ_0 is reconstructed well over the full parameter range.
Figures 7.29 through 7.33 show the same plots for helices. Also in this case the differences between the reconstructed and the simulated track parameters are in agreement with zero. For d_0 the deviations are largest for large simulated values. Also the standard deviation is huge. This can on the one hand again be explained by numerical uncertainties in the algorithm. On the other hand for helices a wrong point of closest approach can be picked. In the algorithm the real point of closest approach and the one on the opposite side of the circle cannot be distinguished. This naturally leads to a wrong z_0 which is the z position of the point of closest approach. Additionally in case for curlers the wrong z_0 might be picked because the track is crossing the point of closest approach more than once at different z.
The differences of ϕ_0 and Ω and the standard deviations are numerically zero, thus the simulated value of the parameter is reproduced precisely. This effect can be explained by the fact that the hits are simulated exactly on the helix and that a very simple circle fit is applied. The parameters describing the track in the sz-projection show very small differences in almost all cases.

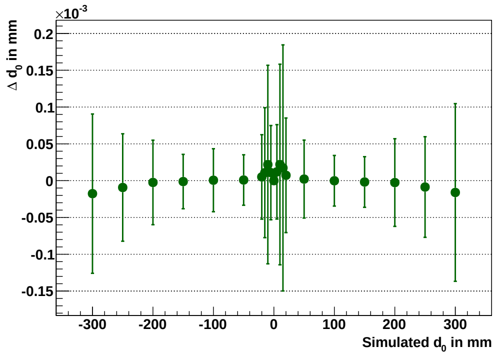


Figure 7.25: Difference between reconstructed and simulated d_0 (straight lines).

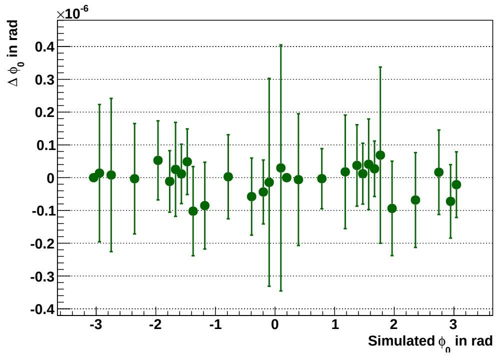


Figure 7.26: Difference between reconstructed and simulated ϕ_0 (straight lines).

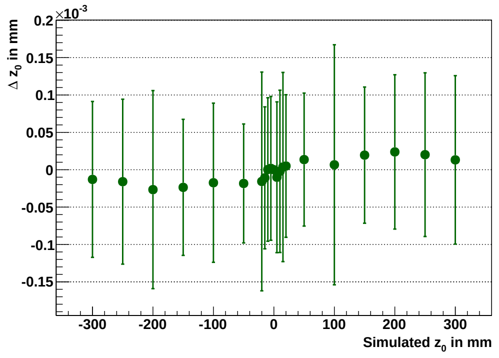


Figure 7.27: Difference between reconstructed and simulated z_0 (straight lines).

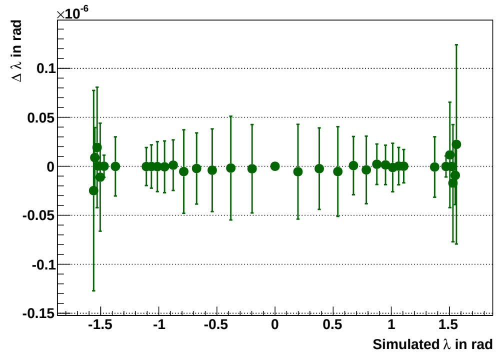


Figure 7.28: Difference between reconstructed and simulated λ (straight lines).

Even though no proper track fit is applied all in all the simulated track parameters are reproduced well by the algorithm.

7.6.2 Smearing

As a next step it is investigated what influence different amounts of smearing have on the track finding algorithm. It is important that tracks can be found reliably if the hits are not exactly on the track. For example in real TPC data hit positions are reconstructed from charge clouds. Due to diffusion the charge cloud is broadened and due to field distortions it is shifted. Thus in real data the hits are not perfectly on a straight line or helix.
To investigate the behavior of `Pathfinder` in this case events with one track each are simulated with random track parameters in the ranges given in Table 7.2. The hit positions are not created such that they are exactly on the track but are shifted by a certain amount off the track in x and z direction. The amount of smearing is determined by a Gaussian function centered around zero with the width of 1 mm. The single point resolution to be achieved in the ILD TPC is smaller than 100 µm (in $r\phi$) and about 1.4 mm (in rz) over the full drift length (see Table 4.2). Thus the

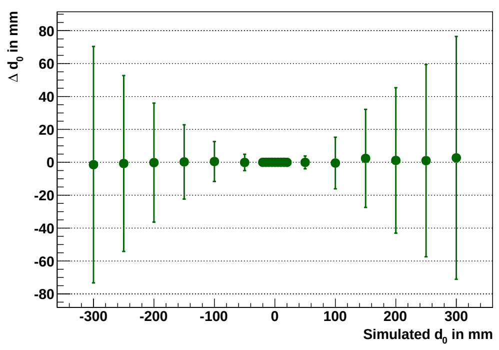


Figure 7.29: Difference between reconstructed and simulated d_0 (helices).

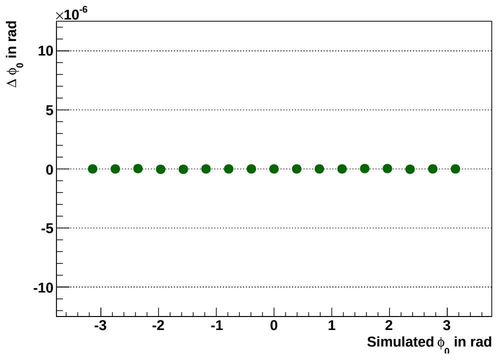


Figure 7.30: Difference between reconstructed and simulated ϕ_0 (helices).

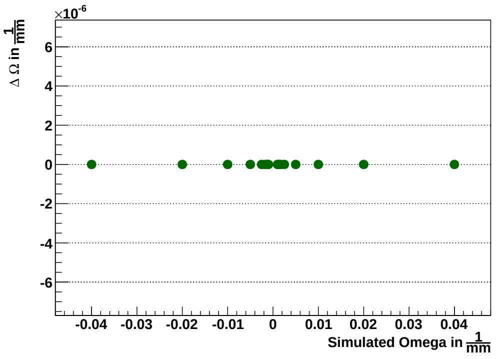


Figure 7.31: Difference between reconstructed and simulated Ω (helices).

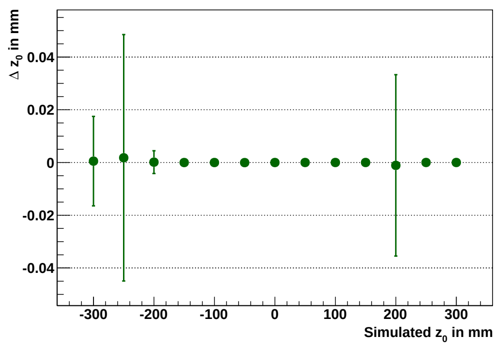


Figure 7.32: Difference between reconstructed and simulated z_0 (helices).

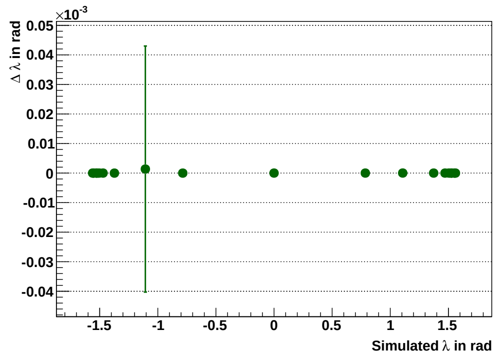


Figure 7.33: Difference between reconstructed and simulated λ (helices).

smearing of 1 mm is about the value expected in rz but is larger than the expected single point resolution in $r\phi$.

In this study the track finding is done with a finer binning (1000 bins in each direction) than before. The efficiency is again defined as the ratio between the number of correctly found tracks and the number of simulated tracks. But here a track is defined to be found correctly if the reconstructed track parameters are similar to the reconstructed ones. The allowed deviations are summarized in Table 7.4. These

Parameter	allowed deviation
d_0	3 mm
ϕ_0	0.01 rad
Ω	0.01 1/mm
z_0	3 mm
$\tan\lambda$	0.01

Table 7.4: Allowed deviations between simulated and reconstructed track parameter to count the track as correctly found.

deviations lead to a wrong extrapolation of the track. Assuming a track in the ILD detector coming from the origin the displacement adds up to about 3 cm at the inner radius of the electromagnetic calorimeter ($r = 1843$ mm [27]), where the deviation of the curvature is neglected. The deviation of the curvature has a big effect on the reconstructed transverse momentum. Especially for small curvatures (high transverse momenta) small deviations have a large influence on the reconstructed transverse momenta.
One should keep in mind that there is no proper fitting applied yet. The track parameters coming out of the Hough transformation are used, which are only a rough estimate of the real track parameters. Rather large differences in the track parameters are allowed in this study (several millimeters). To prove that the new definition for correctly found tracks is as good as the previous one where the hits assigned to the found track were compared to the simulated hits, a smearing of 0 mm was investigated. This corresponds to the previous simulation for the parameter scans where the hits were simulated exactly on the track. The values are shown in Figure 7.34 for helices and straight lines. Comparing these points with the previous

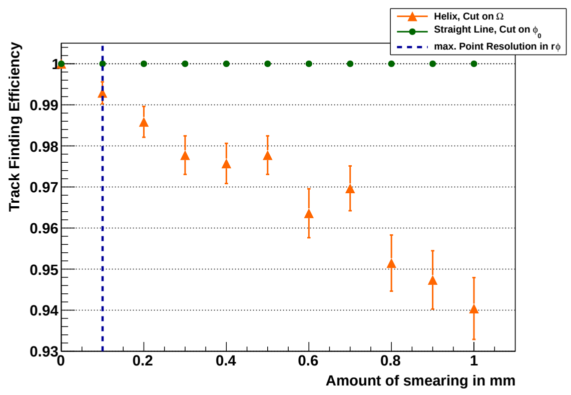


Figure 7.34: Efficiencies for different amounts of smearing for straight lines and straight lines.

plots on the parameter scans show that both efficiencies are in agreement. The new definition for correctly found tracks is therefore comparable to the previous one.
The circles represent the efficiencies for straight lines with different amounts of smearing, the triangles represent the same thing for helices. Tracks with parameters in ranges where the algorithm cannot find tracks (see Section 7.6.1) are not taken int account. The dashed line represents the maximum expected single point

resolution in $r\phi$ for the ILD detector (see Table 4.2). For straight lines the track finding efficiency is one over the full range. For helices, however, the efficiency gets significantly worse for large amounts of smearing. This behavior is expected because if hits are not exactly on the track the point of intersection in Hough space is not as well defined. The individual functions do not all intersect in exactly the same point, they rather approach each other in a certain region. This makes it more difficult to find the correct point and to assign the hits to the track properly.
Helices are more sensitive to smearing. One more parameter needs to be estimated which makes the process of finding helices more complex. Compared to straight lines there is one additional step needed to find helices, which is determining the radius of the circle. This is one additional step where binning effects can be introduced. This is the reason why the efficiency drops faster for helices than for straight lines.

7.6.3 Multi Tracks

It is investigated how well tracks can be found in events with more than one track. Events with various track multiplicities are created randomly with track parameters in the ranges given in Table 7.2 and the hits are positioned exactly on the track. The tracks are reconstructed again, using a fine binning (1000 bins for each direction) and the track finding efficiency is defined as described in Section 7.6.2 by comparing track parameters. Tracks with parameters in the ranges given in Section 7.6.1 are not regarded because these tracks cannot be found by the algorithm.
The track finding efficiencies for this study are shown in Figure 7.35 for straight lines and helices. For both, straight lines and helices, the track finding efficiency

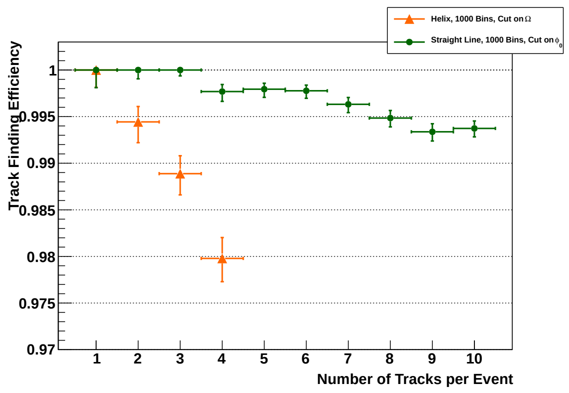


Figure 7.35: Efficiencies for multi track events for straight lines and helices.

drops with increasing number of tracks per event. The reason for this is, that close tracks cannot be separated properly which leads to a wrong estimate of the track parameters. For straight lines the track finding efficiency stays above 99.7% (which is the goal for the track finding efficiency for $t\bar{t}$ events the ILD detector [27]) for up to seven tracks, however, the number of tracks per event investigated here are still much smaller than what is expected in the ILD TPC. In real ILD events several tens of tracks are expected in one event [26]. As was shown in Section 7.5 it is not feasible to test `Pathfinder` for that many tracks. Reconstructing a sufficient amount of such events needs a lot of computing time with the current implementation of

`Pathfinder`. For testbeam data, however, the result is satisfying because events with low track multiplicities are expected, as will be shown in Chapter 9.

7.6.4 Noise Hits

Lastly the effect of noise hits on the track finding efficiency is investigated. A noise occupancy o is defined as

$$o = \frac{N_{\text{Noise Hits}} \cdot N_{\text{Voxel in Hit}}}{N_{\text{Voxel}}} \tag{7.58}$$

and is used as a measure for the amount of noise hits in the TPC. A voxel is a 3D-space bucket with the size (pad width) × (pad height) × (time bin length). The occupancy increases if more noise hits are present. $N_{\text{Noise Hits}}$ is the number of noise hits in the event. $N_{\text{Voxel in Hit}}$ gives the number of voxels covered by each hit. In real data the value is different for each hit. A mean value of 100 voxels is chosen (which is typical value obtained in TPC testbeam data). N_{Voxel} is the total number of voxels available. This can be calculated from the number of pads, the readout frequency and the drift length. Here this value is set to 11082800 (A pad plane with 200 row, 788 pads per row and about 70 time slices is assumed).
Events with one track each are simulated with parameters chosen in the ranges specified in Table 7.2, where tracks in the ranges where `Pathfinder` cannot find them (see Section 7.6.1) are excluded. The hits are positioned exactly on the track. Additionally noise hits are positioned randomly in the detector volume. The reconstruction is done with a fine binning (1000 bins in each direction) and the efficiency is calculated as before by comparing the simulated and found track parameters. Correctly found tracks are defined as in the two previous sections: The parameters of the simulated and reconstructed track must match in the ranges given in Table 7.4.
Figure 7.36 shows the efficiency plot for straight lines and helices. For higher occupancies the track finding efficiency drops. The reason for this is misassignment of

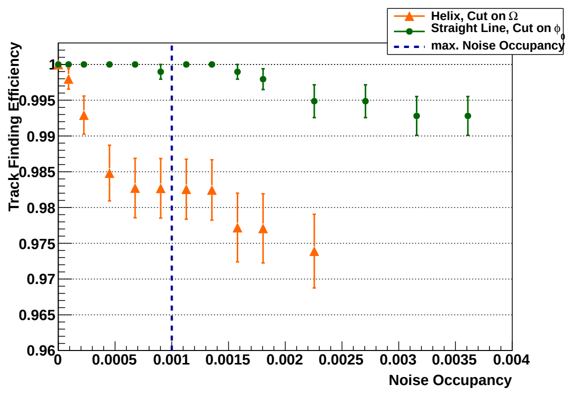


Figure 7.36: Efficiencies for different noise occupancies (amount of voxels with charge) for straight lines and helices. The dashed line shows the maximum expected amount of noise hits in the ILD TPC, in the outer region it will be smaller.

hits to tracks. This includes noise hits being assigned to the track and correct hits missing on the found track. The wrong hits pull the track towards them which leads to wrong track parameters and tracks thus are not marked as found correctly.
In the ILD TPC an occupancy of 0.001 (fraction of voxels with charge) is expected on average [26,92]. For this occupancy the track finding efficiencies for `Pathfinder` is above about 98%, for small occupancies (below 0.002 for straight lines and below 0.0001 for helices) the track finding efficiencies exceed those planned in the ILD of 99.7% [27].

7.6.5 Hough Space Binning

The track finding efficiency is influenced by the chosen binning. To study this tracks are simulated in the parameter ranges given in Table 7.2 and tracks with parameters in the ϕ_0 and Ω regions which cannot be found are not used. A track is found correctly if the parameters match those of the simulated track in the ranges given in Table 7.4. For the track reconstruction the maximum allowed distance between hit and track is set to 10 mm and at least five hits are required to be assigned to the track. The number of bins is the same in each direction and is varied between 25 and 1000 bins.
The results of the binning study are shown in Figure 7.37. The influence of the binning is shown for straight lines and helices, without smearing and without noise. In case for straight lines track finding efficiency is 1 even when using 200 of more bins. Below that the efficiency drops. For helices is stable above 98 % when using more than 500 bins, however, a it slowly increases with number of bins.
If the binning is too coarse, the estimation of the track parameters is not precise enough to assign hits to the track correctly. The track finding efficiency can be influenced via the maximum allowed distance between hit and track. If the binning is chosen to be coarse the allowed distance between hit and track can be chosen larger in order to still find tracks. This, however increases the chance that wrong hits are assigned to a track and thus track separation does not work properly anymore. If the binning is getting too fine the point of intersection is not well defined (which is particularly true if the hits are far away from the track) and thus this could lead to

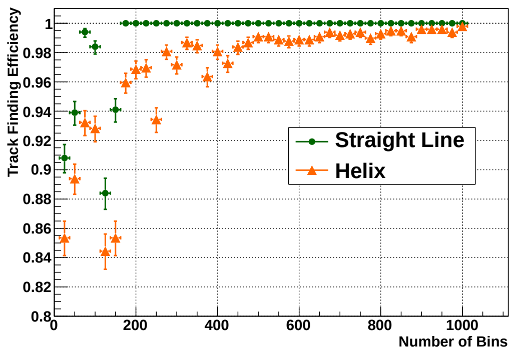


Figure 7.37: Track finding efficiency for different Hough space binning for events with a single track (straight line and helix) each without smearing and without noise.

low efficiencies as well. In the study shown here such fine binning is not reached. The binning should be chosen as fine as possible (so that the intersection is well defined) so that the maximum distance between hit and track can be chosen small to get the optimal result. A fine binning, on the other hand, means more computing time. Thus a compromise needs to be found between acceptable computing time and acceptable inefficiency.

7.7 Limitations of the Algorithm

As shown in the previous sections, `Pathfinder` can find tracks reliably in almost all parameter ranges. One exception is, that it cannot find tracks which are parallel to the z-axis. This is not regarded as a real problem since the algorithm is implemented for track finding in time projection chambers. In such a detector tracks parallel to the z-axis look like a single point in the readout plane.
Another exception is that tracks passing parallel to a pad row cannot be found. Such tracks are reconstructed as one hit covering the full row. A different approach is needed to find such tracks. However, in real data it is expected that this is unlikely to happen.
Another known feature of the algorithm is the ambiguity in track parameter combinations, which is due to the definition of the track parameters and not caused by the algorithm itself. There is more than one set of parameters which can describe the same geometrical object. The only difference is the direction of motion of the particle. In the current implementation of `Pathfinder` they cannot be distinguished.
For helices a wrong point of closest approach can be calculated. The real point of closest approach and the point lying at the opposite position on the circle cannot be distinguished in the algorithm. Since the arc length s is zero at the point of closest approach a wrong value for z_0 is found. Finally, closely related, when trying to find a curler the point of closest approach can have various possible z positions. This leads to a wrong z_0 as well. Nevertheless, hits can be assigned correctly to the track and this is thus not causing inefficiencies.
Simulated and reconstructed track parameters are compared and in almost all cases they are the same or the differences can be explained by one of the features mentioned above, even without a proper track fit applied. Thus the track parameters are good enough to serve as start parameters for a track fit.
The current implementation of `Pathfinder` has one more issue which needs to be addressed in the future. The computing time is high especially for complex events with many tracks and many hits per track. `Pathfinder` is mainly implemented to be applied on testbeam data which usually have low track multiplicities. In such cases the algorithm is sufficiently fast. In multiple track events in an ILD-like detector the computing time rises significantly with the number of of tracks (and thus the number of hits) per event, as was shown in Section 7.5. This can be resolved by changing the algorithm by either implementing an adaptive Hough transformation [86] or by changing the current algorithm in such a way that all intersections in Hough space are found in one go so that the iteration until all tracks are found can be abandoned. Both solutions mean additional programming work. The effort is not reasonable at this stage because `Pathfinder` was mainly written to reconstruct

testbeam data with few (usually one) track with only a few hits per track. Another option to improve the speed of the algorithm is to calculate all intersections of the functions in Hough space analytically and fill those into the Hough space. This on the one hand means one more loop over hits but on the other hand the computing time does not depend that strongly on the chosen binning[8] and the map containing the Hough space is much smaller.
Additional limitations are expected if the track is not a perfect helix (due to effects such as energy losses). In this case the curvature is not constant over the full track. This causes inefficiencies especially for tracks with small momenta because the tracks are no perfect circles in the xy-projection. However in the systematic tests presented in this chapter this case was not investigated. In Chapter 8 more realistic events are investigated which take these effects into account.

[8]In such a case the binning only affects the speed of the algorithm when the search for intersections is done.

Chapter 8

Track Finding with `Pathfinder` on simulated Data

In Chapter 7.6 `Pathfinder` was used on simplified simulated data. The simulation was creating random track parameters and putting hits along this track. It was shown that it is possible to find track in such events reliably. However it is important to see if the track finding algorithm works for realistic data. In this chapter studies on more realistic simulated events are presented. Muons and taus were simulated in the ILD detector using the simulation package `Mokka` [63, 64] which is based on geant4 [55, 56]. Also `Pathfinder` was used on Large Prototype testbeam data, which is presented in Chapter 9.

In this chapter the track finding algorithm used in `Pathfinder` will be compared to another track finding algorithm implemented in a software package called `Clupatra` [93]. This is a local track finding method based on a nearest neighbor clustering algorithm and a Kalman Filter [80, 94]. When looking at the comparisons one should keep in mind that `Pathfinder` and `Clupatra` follow different track finding approaches and thus differences are expected, especially because `Clupatra` gives back fitted track parameter while the track parameters calculated by `Pathfinder` are only a rough estimate since no proper track fit is applied. The track parameters given back by `Clupatra` are expected to describe the data better than those calculated by `Pathfinder`. This again leads to a better reconstructed transverse momentum for `Clupatra` meaning that the differences between the simulated and the reconstructed transverse momenta are smaller for the results obtained with `Clupatra`.

Furthermore `Clupatra` uses a track matching algorithm which `Pathfinder` does not yet do. One track can be found in two parts with similar track parameters. This can happen if the particle lose energy in the tracker (which leads to a change in curvature) or by scattering. Losses of energy are mainly relevant for particles with low kinetic energies. The lower the kinetic energy the more energy the particles lose (Bethe-Bloch equation [39, 4]) and the stronger the curvature changes. This effect is larger for light particles. In such a case where several tracks are found with similar track parameters the tracks are merged in `Clupatra` but not in `Pathfinder`. Thus a higher efficiency is expected when using `Clupatra`.

8.1 Evaluating the Performance of Track Finding Algorithms

In order to evaluate how well a track finding algorithm works various quantities have to be defined [81]. The definition of the ones used will be given here.
First of all one needs to know the maximum number of tracks which can be found. This information is obtained by the `TruthTracker` [95] which takes the Monte Carlo information and the hit positions into account and builds tracks based on this information. This gives a good estimate of which tracks maximally can be found, at least in principle. However, the `TruthTracker` does not necessarily deliver correct tracks. This is especially true for curlers, particles with low momentum which travel along more than one loop of a helix.
The track finding efficiency is defined as

$$\epsilon \;=\; \frac{N_{\mathrm{correct}}}{N_{\mathrm{total}}}, \tag{8.1}$$

where N_{total} is the number of tracks in the event according to the `TruthTracker`. N_{correct} is obtained by comparing the found tracks to the true tracks. There are two ways how to do this. First the track parameters can be compared. This method is not chosen to evaluate the performance of the track finding algorithm because the tracks delivered by `Pathfinder` are not fitted and therefore only a rough estimate of the track parameters is available. The second option is chosen to compare the hits which are assigned to the tracks. A track is defined to be found correctly (if not stated otherwise) if not more than 25% of the hits are assigned wrongly[1].
As a next step the tracks which are marked as not found correctly can be investigated further. Most tracks that are not found correctly are split into two parts. This will especially happen for particles with low transverse momentum which suffer large energy losses while traveling through the sensitive volume. This results in a large variation of the curvature which again leads to split tracks. One can thus define a split rate

$$s \;=\; \frac{N_{\mathrm{split}}}{N_{\mathrm{total}}}, \tag{8.2}$$

where N_{split} is the number of split tracks.
A very similar quantity is the fake rate. The fake rate describes how many of the reconstructed hits could not be assigned to a simulated particle trajectory due to other reasons than track splitting. This usually are tracks consisting of hits being produced by many different particles. These hits are located by chance in such a way in space that the track finding algorithm finds them to be on the same track. The fake rate is defined as

$$f \;=\; \frac{N_{\mathrm{fake}}}{N_{\mathrm{total}}}, \tag{8.3}$$

where N_{fake} is the number of fake tracks.
The two quantities (split rate and fake rate) are similar in the sense that they

[1]These can be either missing hits or additional hits.

look at reconstructed tracks that cannot be assigned to a single simulated particle trajectory. To discriminate between a split track and a fake track a quantity called purity is defined. This is a measure for how many hits on the reconstructed track are created by the same particle. For split tracks this quantity is expected to be high because most hits are expected to come from the same particle, however, not all hits produced by the particle are assigned to the track. For a fake track on the other hand the purity is expected to be small because such tracks are a collection of hits from various particles which by chance seem to have been created by the same particle.
Note that for the split rate and the fake rate the value N_{total} represents the total number of reconstructed tracks not the number of truth tracks as for the track finding efficiency. The split rate and the fake rate are the amount of all reconstructed tracks that are split tracks and fake tracks, respectively.
A track finding algorithm should have a track finding efficiency as large as possible while at the same time the split rate and the fake rate should be as small as possible. The split rate can usually be decreased by applying a track matching algorithm which means that split tracks are merged to one track. This at the same time increases the track finding efficiency.
There is one more quantity of interest which describes the performance of track finding algorithms in general. This is the clone rate

$$c = \frac{N_{\text{clone}}}{N_{\text{total}}}. \tag{8.4}$$

This quantity is sensitive to tracks that are reconstructed correctly more than once and are thus redundant. In case for the track parameters used in `Pathfinder` in principle an infinite number of clones can be found for each track. With the track parameters (d_0 being the distance of closest approach and θ being the angle between the positive x-axis and the distance of closest approach) chosen, the transformed functions have sine shape in Hough space which means that they are periodic and thus in every interval of 2π length there are two points of intersections of the functions, one at d_0 and θ, one at $-d_0$ and $\theta + \pi$ (see also Chapters 7.1.1 and 7.1.2). However the parameters obtained from all of these intersections describe the same track and lead to clone tracks. However, this ambiguity is resolved in the algorithm directly by limiting the Hough space in θ to a range of width π and thus all clone tracks are rejected automatically. For this reason the clone rate is not investigated to evaluate the performance of `Pathfinder`.

8.2 Simulation and Reconstruction of Particles in the ILD

A `Mokka` simulation of muons and taus is done with the detector model ILD_O1_v03 [96]. The muons for the momentum scan (see Section 8.3.1) and the track separation studies (see Section 8.3.3) are created in the center of the detector and shot with an angle of a few degrees into the TPC (approx. 10 degrees between the initial particle momentum and the plane perpendicular to the beam pipe). This is

Parameter	**Value**
`DiffusionCoeffRPhi`	$0.025\ \frac{\text{mm}}{\sqrt{\text{cm}}}$
`DiffusionCoeffZ`	$0.08\ \frac{\text{mm}}{\sqrt{\text{cm}}}$
`DoubleHitResolutionRPhi`	2 mm
`DoubleHitResolutionZ`	5 mm
`PointResolutionPadPhi`	0.9 mm
`PointResolutionRPhi`	0.05 mm
`PointResolutionZ`	0.4 mm

Table 8.1: Parameters used for the `TPCDigiProcessor` to smear and merge the hits simulated with `Mokka`.

Steering Parameter	**Value**
minimum Number of hits on track	30
maximum distance hit - track (xy and sz)	10
number of bins (all directions[2])	1000

Table 8.2: `Pathfinder` steering parameters used for the reconstruction of tracks produced by muons and taus decay products.

necessary to prevent the particles from crossing the cathode of the TPC located at the center of the detector. For the θ scan muons with a momentum of 50 GeV are created. The taus are also created in the center of the detector with a momentum of 30 GeV but are simply shot upwards. Since they decay into other particles it is unlikely that one of the decay products ends up in the cathode, however this cannot be completely excluded. `Mokka` takes energy losses into account which makes the simulation more realistic than the simplified simulation used for first tests of `Pathfinder` (see Chapter 7.6). The hits created with `Mokka` are located perfectly on the track. To make the events more realistic the processor `TPCDigiProcessor` [97] is used. It applies smearing to the hits and merges close hits. The parameters used for the `TPCDigiProcessor` are set to the default values and are listed in Table 8.1. The values are chosen such that they match the values needed to be achieved by the ILD TPC (see Table 4.2).
The simulated tracks are reconstructed with `Pathfinder` twice, once assuming a helix and once assuming a straight line. The track finding efficiencies are compared to those obtained with a different track finding algorithm called `Clupatra`, which follows a different track finding approach. Only the hits in the TPC are used as input for `Pathfinder` and `Clupatra`. Those hits simulated in other parts of the detector are not taken into account. The steering parameters for `Pathfinder` are summarized in Table 8.2. If not stated otherwise these parameters are used to reconstruct the simulated tracks for muons and tau decays.

[2]For straight lines number of bins in Ω-direction is not needed.

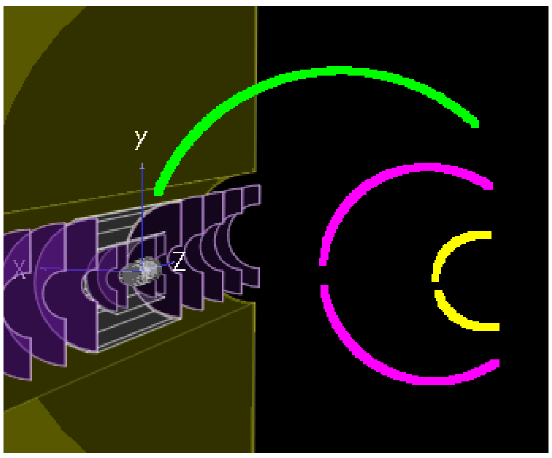


Figure 8.1: Event display of a reconstructed muon with $p_T = 1$ GeV in the ILD detector. Each color represents one found track. Only hits in the TPC are shown.

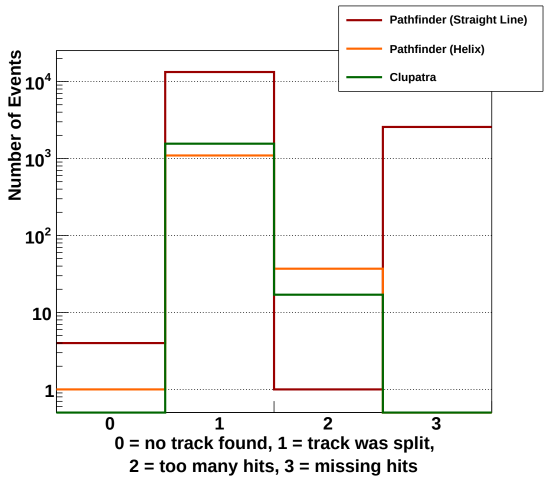


Figure 8.2: The reason why tracks are not found correctly for `Pathfinder` assuming Helix and Straight line and for `Clupatra`.

8.3 Efficiency Studies for Particles in the ILD

8.3.1 Single Muons in the ILD

Single muons are simulated in the ILD detector with transverse momenta in a range between about 1 and 50 GeV. Since it is a rather realistic simulation also secondary particles like electrons are simulated. In some events these additional electrons can be seen as curlers in the TPC.

An example event can be seen in Figure 8.1 where an event display of a muon with a transverse momentum of 1 GeV in the ILD detector is shown. The muon is started in the center of the detector and then enters the TPC. It travels through the TPC and leaves it. Then it deposits energy in the electromagnetic calorimeter (calorimeter hits are not shown) and is then curling back into the TPC with less energy. This even happens a second time. `Pathfinder` finds three individual tracks which would count as not found correctly. The `TruthTracker`, which finds the maximum possible amount of tracks in the detector based on the Monte Carlo information, does not find it as one single track either. Thus all three tracks are counted as found correctly, and thus the track finding efficiency is high in this p_T region (see Figure 8.3).

As a next step it is investigated in more detail why tracks are not found correctly. This is shown in Figure 8.2 for `Pathfinder` (track models straight line and helix) and `Clupatra`. Here tracks are defined to be found correctly if all hits are assigned correctly to the found track thus all hits on the found track have to be created by the same particle and all hits having been created by that particle have to be on the found track. For `Pathfinder` assuming straight lines mostly the reason for wrongly found tracks is, that the track is split in several parts or hits are not assigned correctly to the track. This is understandable for tracks with a significant curvature.

In such cases the tracks cannot be described by a straight line and the track is thus split into small parts which are consistent with a straight line. It can also happen that a few hits remain which cannot be assigned to a track and are not enough to build another track.
`Pathfinder` assuming a helix shows a significant amount of split tracks which can be explained by energy losses in the TPC or the scattering of particles. Tracks are in this case no proper helices anymore and are split. There is also a large fraction of tracks to which more hits are assigned than expected. This can be explained by the fact that the true tracks are created with the `TruthTracker` which does not create curlers correctly while `Pathfinder` can cope with them if the energy losses are not too large. Furthermore hits created by secondary particles can be added to the track of the primary particle. The results for `Clupatra` look similar to those of `Pathfinder` assuming a Helix which means that the two algorithms need to face similar challenges.
The plot in Figure 8.3 shows the track finding efficiency for different transverse momenta of single muons in the ILD. A track is defined to be found correctly if at least

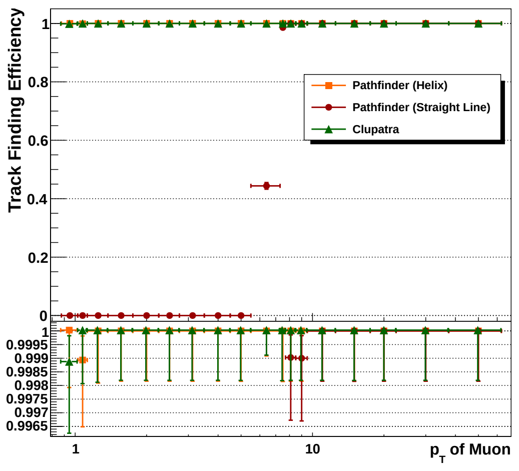


Figure 8.3: Track finding efficiencies for single muons simulated with Mokka in the ILD detector for various p_T. The reconstruction is done with `Pathfinder` (assuming a straight line and a helix) and with a track finding method based on cluster finding and a Kalman Filter (`Clupatra`).

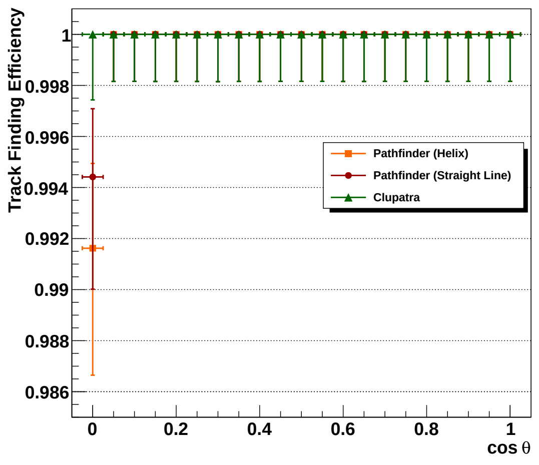
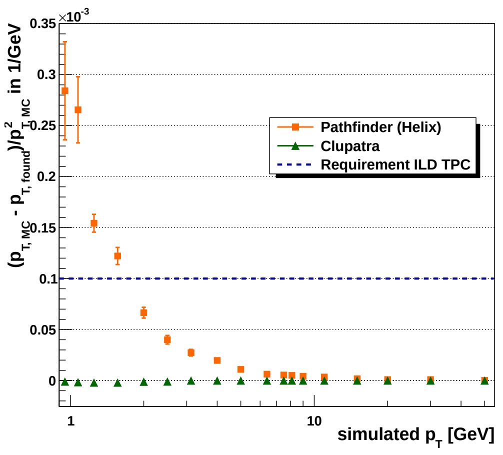


Figure 8.4: Track finding efficiencies for single muons simulated with Mokka in the ILD detector with a momentum of 50 GeV and various angles θ. The reconstruction is done with `Pathfinder` (assuming a straight line and a helix) and with a track finding method based on cluster finding and a Kalman Filter (`Clupatra`).

75 % of all hits on the found track are produced by the same particle. The number of tracks which in principle could be found is determined by the `TruthTracker`, which uses the Monte Carlo information and the positions of the simulated hits to build tracks. The events are reconstructed three times. First they are reconstructed with `Pathfinder` using a helix as track hypothesis, then they are reconstructed again with `Pathfinder` using a straight line. Lastly the events are reconstructed with `Clupatra`, which follows a local approach to find tracks, for comparison. In case

for `Pathfinder` using helices and for `Clupatra` the track finding efficiencies drop slightly for transverse momenta of about 1 GeV. The reason for this is that for low momenta energy losses have a much bigger effect on the curvature of the track so that tracks cannot be found correctly anymore, however the effect is small (about 1 out of 1000 tracks).
For `Pathfinder` using straight lines the track finding efficiency drops for transverse momenta smaller than about 10 GeV. Those tracks have a curvature too large to match straight lines. However these results also strongly depend on the steering parameter used for `Pathfinder` and on the definition of what a correctly found track is. Comparing `Pathfinder` and `Clupatra` shows that the two algorithms deliver the same results for the track finding efficiency for single muons.
In Figure 8.4 the track finding efficiency for single muons with a momentum of 50 GeV crossing the TPC with different angles is shown. θ is the angle between the z-axis and the initial momentum of the particle. Thus $\cos\theta$ is zero if the particle moves perpendicular to the z-axis. Since the muons are started in the center of the TPC in this case they end up in the cathode of the TPC where the material budget is higher than in the other regions of the TPC. The muons interact with the detector material and lose more energy than in the drift volume of the TPC. For this reason it is more difficult to reconstruct the tracks correctly which explains the low track finding efficiency in this region. For $\cos\theta$ larger than about 0.77 not the full track can be measured because the muon leaves the TPC through the anode of the TPC. For $\cos\theta$ larger than about 0.97 less than 30 hits can be measured in the TPC per track at maximum. For this region during track finding with `Pathfinder` the requirement of having at least 30 hits per track is changed to having at least 5 hits per track. In general `Pathfinder` and `Clupatra` show good agreement. For comparison: in $t\bar{t} \rightarrow 6$ jets an average efficiency for momenta larger than 1 GeV is expected to be 99.7%[3] [27]. However,the case shown here is still rather simple. In the next sections more complicated events are investigated. In the following the single muon events are investigated further in order to get a better understanding on how the algorithm behaves.
In the ILD detector it is essential to reconstruct the momentum of a particle precisely. This is important for the particle identification and the particle flow concept followed in the ILD detector. Thus the reconstructed transverse momentum of the single muon events is compared to the simulated one. The result is shown in Figures 8.5 (momentum resolution for different transverse momenta) and 8.6 (momentum resolution for different angle θ) for `Pathfinder` searching for a helix and for `Clupatra`. In an ideal case the simulated and reconstructed transverse momentum are the same. This would mean that the measurement points in Figures 8.5 and 8.6 would be at zero. For `Pathfinder` the reconstructed transverse momenta deviate more from the simulated ones than for `Clupatra`. The reconstructed transverse momentum is found to be smaller than the simulated one because the particles lose energy in the detector. The deviations are largest for small simulated transverse momenta and large $\cos\theta$. For small transverse momenta tracks are more difficult to find and energy losses influence the result. For this reason the deviations are largest in this region. For large $\cos\theta$ the momentum resolution gets worse as well.

[3]combined tracking systems



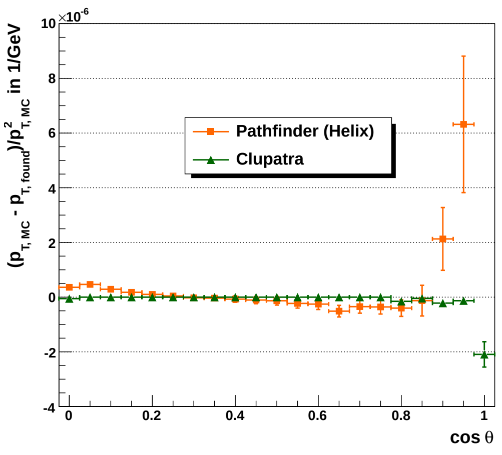


Figure 8.5: Difference of the reconstructed and simulated transverse momentum over simulated transverse momentum.

Figure 8.6: Difference of the reconstructed and simulated transverse momentum for different angles with which the muons cross the TPC.

This can be explained by the fact that not all space points can be measured and the lever arm is shorter because the particle leaves the TPC through the readout plane (see equation 4.10). This decreases the momentum resolution. For large transverse momenta and small $\cos\theta$ both algorithms can reconstruct the transverse momentum reliably. For transverse momenta larger than about 1.5 GeV the resolution goal for the ILD TPC is met even with the rough estimate of track parameters coming from the Hough transformation.
One has to keep in mind that `Clupatra` uses a Kalman Filter which delivers fitted tracks, the track parameters coming from `Pathfinder` are only a rough estimate and the tracks still need to be fitted. Furthermore `Clupatra` includes a track matching algorithm in which tracks with similar track parameters are merged to one track. Applying both, fitting and track merging, to the tracks found with `Pathfinder` is expected to improve the results especially in cases where energy losses play a significant role.

8.3.2 Single Muons in the ILD with Detector Inefficiencies

In order to test how well the track finding algorithm works for detector inefficiencies (hits missing on the track) single muons with a transverse momentum of 10 GeV are simulated with Mokka in the ILD detector. As shown before (see Figure 8.3), these tracks can be found reliably with `Pathfinder` assuming a straight line and a helix and with `Clupatra` if no hits are missing. Events with one full track (which means there are no hits missing on the track after simulation) are chosen. Then different fractions of hits are removed randomly from the tracks and the track finding is done again with `Pathfinder` and `Clupatra`. The track finding efficiencies for different fractions of missing hits are shown in Figure 8.7. The TruthTracker is used to determine how many tracks at maximum can be found. A track is again defined to

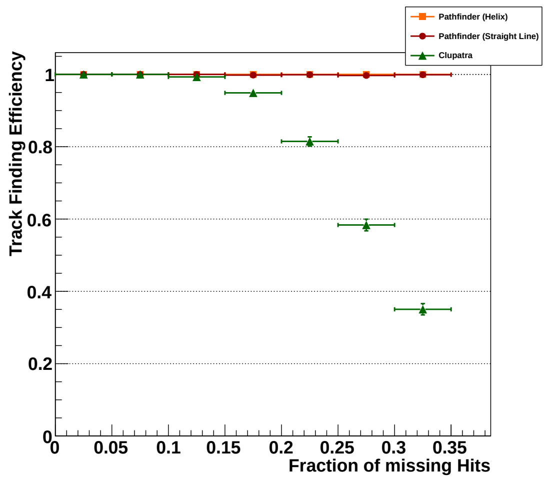


Figure 8.7: Track finding efficiency for different amount of missing hits.

be found correctly if 75% of the hits are assigned correctly to the track. `Pathfinder`, using a global method, has no significant problems finding the tracks with missing hits no matter if a straight line or a helix is assumed. For `Clupatra` on the other hand the track finding efficiency starts breaking down at about 15 % of missing hits. This effect increases if more hits are missing.
The low efficiency for `Clupatra` is due to the fact that `Clupatra` uses a local track finding algorithm which adds hits row by row. If too many measurements are missing there is too little information to extrapolate the track properly. If the prediction of the track is wrong hits cannot be found and added anymore. The track search is stopped before the full information can be used. This leads to a reduction of the track finding efficiency. `Pathfinder` on the other hand uses a global method where all hits enter at the same time. The whole information is used to find the track and the algorithm does not perform worse when hits are missing. This is a clear advantage of global track finding methods like the Hough transformation over local methods.

8.3.3 Muon Pairs in the ILD and Track Separation

In Section 8.3.1 it is shown that single muons in the ILD can be reconstructed reliably using `Pathfinder`. In this section it is investigated how well two tracks being close to each other can be separated and how far they need to be apart. Two muons with a transverse momentum of 50 GeV are simulated in the center of ILD with Mokka. They are shot into the TPC with an angle of about 10 degrees to ensure that the muons do not end up in the cathode of the TPC. A high momentum is chosen so that the tracks are almost straight. As a measure for the distance between the tracks the angle between their initial momenta is used. The simulation is done for various angles between the initial momenta in the xy and yz-plane separately. Additionally in the xy-plane muons with same charge and with opposite charge are simulated. Tracks created by two muons with opposite charge are expected to be easier to separate because they bend into different directions in the magnetic field. How well the track separation works strongly depends on the steering parameters used for the Hough transformation. The smaller the allowed distance between hit and track is chosen the better tracks can be separated. However, if the value is chosen too small, tracks cannot be found properly anymore because many hits cannot be assigned to the track correctly. In the end the limit corresponds to the value of the double hit resolution used during digitization with the `TPCDigiProcessor`. Due to this hits of the two tracks are merged if they are very close to each other. This makes it impossible to separate for example two same sign muons with an angle of zero degrees between them.

Up to now the efficiency was based on the number of correctly found tracks. This definition is not useful in this case since the separation of two tracks is under investigation. Thus it is required that both tracks in the event are found correctly. The enumerator of the efficiency is thus not the number of correctly found tracks but the number of events in which both tracks are reconstructed correctly and the denominator is the total number of events instead of the total number of tracks. The definition of a correctly found track is the same as before (no more than 25 % of wrong hits on the track).

The result for same sign muons separated in the xy-plane is shown in Figure 8.8. The efficiency breaks down for small angles and the edge is shifted for different allowed distances between hit and track. The larger the allowed distance between hit and track gets, the worse the two tracks can be separated.

The same plot but for muons with opposite sign is shown in Figure 8.9. The two

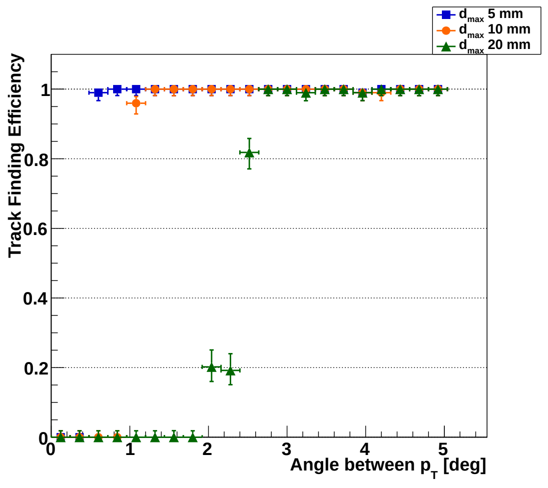


Figure 8.8: Track finding efficiencies for two-muon-events in xy where both muons have the same sign. The muons are simulated with different angles between them. Track finding is done on the same sample with different allowed distances between hits and tracks.

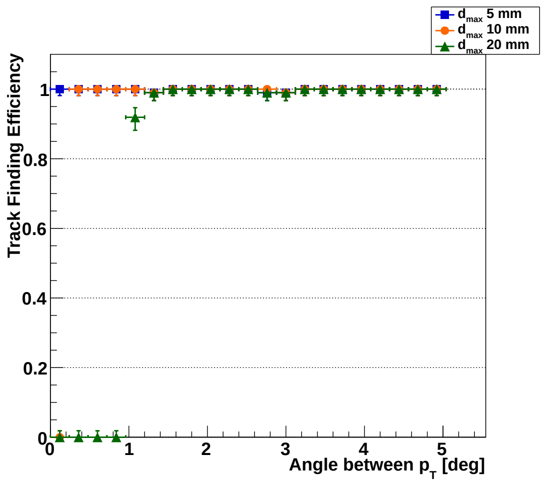


Figure 8.9: Track finding efficiencies for two-muon-events in xy where the muons have opposite sign. The muons are simulated with different angles between them. Track finding is done on the same sample with different allowed distances between hits and tracks.

tracks curve into opposite directions. Thus the tracks are much further separated in large regions than for same sign muons. For this reason the track finding efficiency is close to one for almost all angles and all allowed distances between hits and tracks

under investigation. The only exception is the case for an allowed distance of 20 mm which is rather large. The number of misassigned hits is getting large in this case because especially in the inner part the algorithm assigns hits to the wrong track.
Figure 8.10 shows the track finding efficiency for same sign muons separated in z. The muons have the same charge and no angle between them in the xy-projection.

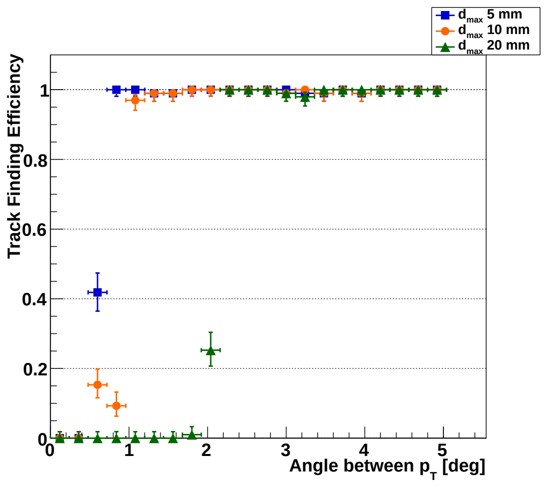


Figure 8.10: Track finding efficiencies for two-muon-events in sz where the muons have the same sign and are not separated in xy. The muons are simulated with different angles between them. Track finding is done on the same sample with different allowed distances between hits and tracks.

The muons cannot be separated in the xy-plane. Any separation of tracks is thus done in the sz-projection of the search. For small angles the track finding efficiency rises in a similar way and for similar values as for same sign muons separated in xy. This behavior is expected from the similarity of the tasks.

8.3.4 Tau Decays in the ILD

The next level of realism are events with more than two tracks. To study this tau decays are simulated in the ILD. In most cases one charged particle is in the tau decay (1-prong events). In these events the track is expected to be found correctly and the results should look similar to those from single muons. More interesting are events with three charged particles in the event (3-prong events). About 10 % of the events are 3-prong events. The different decay modes are summarized in Table 8.3. For the analysis 20000 events with one tau with a momentum of 30 GeV each are simulated with Mokka. The taus are started in the center of the ILD detector. The most important results of the analysis are given here.
During track finding a vertex constraint is used which assumes that the particles seen in the TPC are produced at the vertex, which is the origin of the coordinate system, where the taus are produced in the simulation. However the tau does not decay immediately but travels a certain distance before decaying. It has a mean life time of $(290.6 \pm 1.0) \times 10^{-15}$ s [4]. In this time a tau with a momentum of 30 GeV travels 1.470 ± 0.006 mm, assuming a mass of 1.77682 ± 0.00010 GeV [4]. The tracks measured in the TPC are thus not coming from the origin. To prove that the mean

Decay Mode	Total Fraction
1-prong:	
$\tau \rightarrow \pi^-\pi^0\nu_\tau$	$(25.52 \pm 0.09)\%$
$\tau \rightarrow e^-\overline{\nu}_e\nu_\tau$	$(17.83 \pm 0.04)\%$
$\tau \rightarrow \mu^-\overline{\nu}_\mu\nu_\tau$	$(17.41 \pm 0.04)\%$
$\tau \rightarrow \pi^-\nu_\tau$	$(10.83 \pm 0.06)\%$
$\tau \rightarrow \pi^-\pi^0\pi^0\nu_\tau$	$(9.30 \pm 0.11)\%$
3-prong:	
$\tau \rightarrow \pi^-\pi^+\pi^-\nu_\tau$	$(9.31 \pm 0.06)\%$

Table 8.3: The most important decay modes of the tau [4].

distance the taus travel is small enough to be consistent with the vertex constraint the distances of the starting point of all particles measured in the TPC are plotted, which is shown in Figure 8.11. The assumption is made that all particles in the TPC are produced by the tau decay and in 3-prong events the tau is not counted once but three times. An exponential function of the form of the decay law

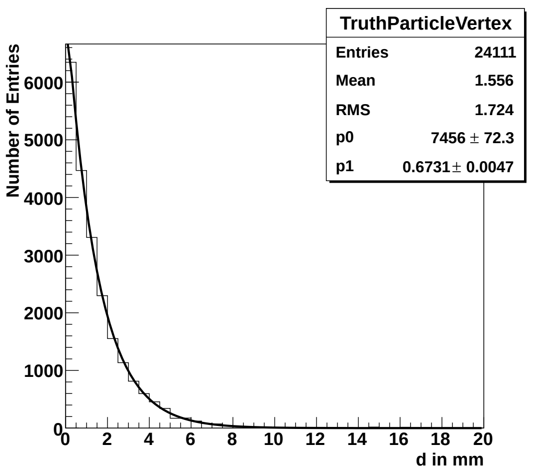


Figure 8.11: Distance of the starting point of the particles measured in the TPC to the vertex where the taus are created.

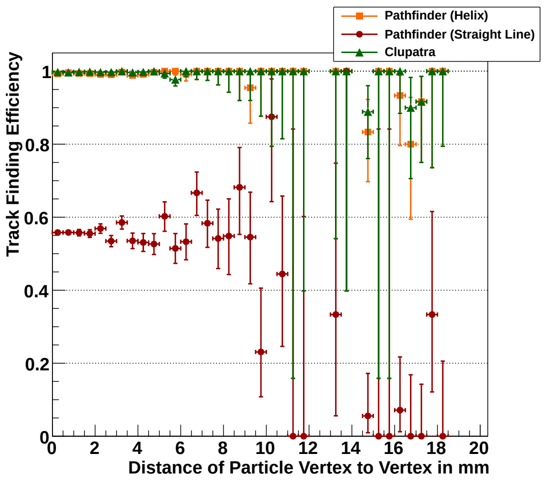


Figure 8.12: Track finding efficiencies for tau decays for distances between track start point and vertex.

$$f(x) = N_0 e^{-\lambda x} \tag{8.5}$$

is fitted to the distribution. The mean length the taus travel before decaying is given by $1/\lambda = 1.486 \pm 0.009$ mm which is consistent with the expectation given above.

To prove that the approximation done by the vertex constraint is valid can be seen in Figure 8.12 which shows the track finding efficiencies for different production points of the particles producing the track. The track finding efficiency is one over almost the full region for `Pathfinder` assuming a helix and for `Clupatra`. Thus the approximation that the track is coming from the vertex is valid. The track

finding efficiency for `Pathfinder` assuming a straight line is only about 0.5. The reason for this is that tracks produced by particles with a small momentum cannot be described by a straight line. For large distances the errors on the efficiencies get very large which is due to the small statistics available in this region (see Figure 8.11).
In Figure 8.13 the track finding efficiency is shown for all tau decays (including all decay modes). The track finding efficiencies are almost 1 in most regions. However,

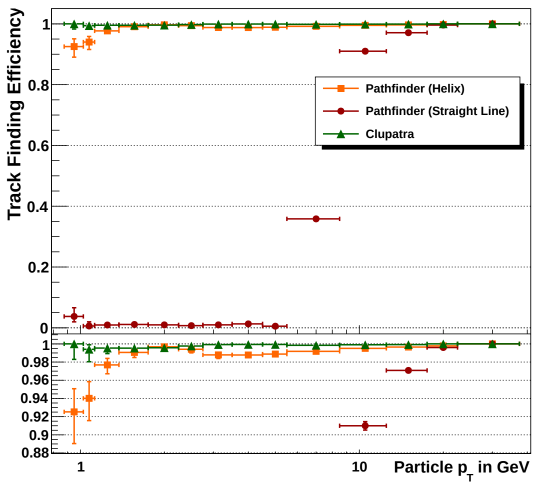


Figure 8.13: Track finding efficiencies for tau decays including all decay modes.

for small transverse momenta the efficiencies drop significantly. The effects of energy losses have a much stronger effect on particles with low p_T, this is especially true for electrons which deposit more energy in the TPC than the other particles. Thus it is expected that the inefficiencies for low transverse momenta appear stronger for taus decaying into electrons than for taus decaying into muons. The efficiencies are shown in Figures 8.14 for electrons and 8.15 for muons, which are mostly 1-prong events. Comparing these two plots shows that indeed for electrons the inefficiencies in the low p_T region up to about 5 GeV are larger than for muons. The result for the muons look very similar to those from the single muon events shown in Figure 8.3. This gives a consistent picture.
To study the performance of `Pathfinder` in more complex events 3-prong events are selected. The particles crossing the TPC are in this case mostly pions. The efficiencies for different transverse momenta in 3-prong events are shown in Figure 8.16. The track finding efficiencies are reasonably high but do drop between 4 and 15 GeV for `Pathfinder` assuming a helix while they are almost 1 for `Clupatra`. 3-prong events are more complex events than 1-prong events, which can be an explanation for this effect. However, some further analysis is done to understand the inefficiencies better.
To get an idea why tracks are not found well in such a big p_T range the fake rate and the split rate are plotted for 3-prong events. For a track to be marked as split a purity of at least 0.75 is required[4]. This value is chosen by filling the purities of all

[4]This means that 75% of the hits on the track are created by the same particle.

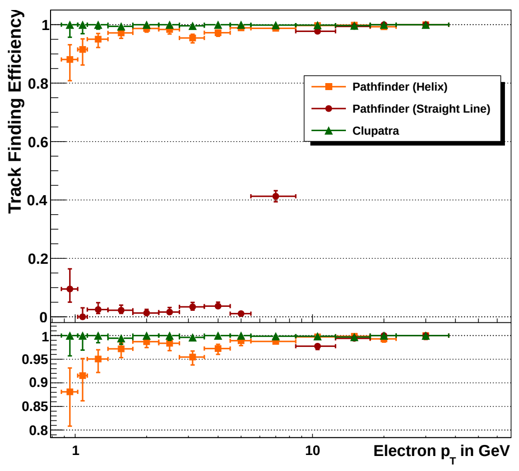


Figure 8.14: Track finding efficiencies for taus decaying to an electron.

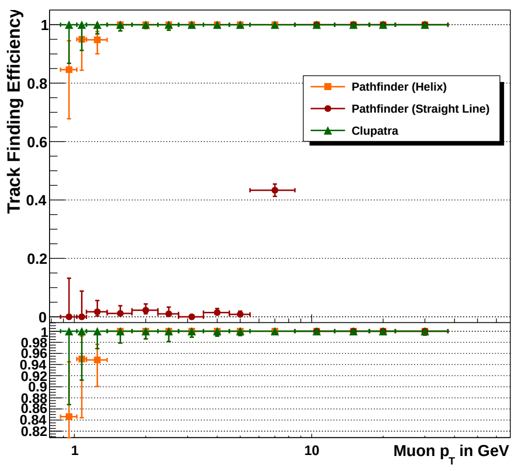


Figure 8.15: Track finding efficiencies for taus decaying to a muon.

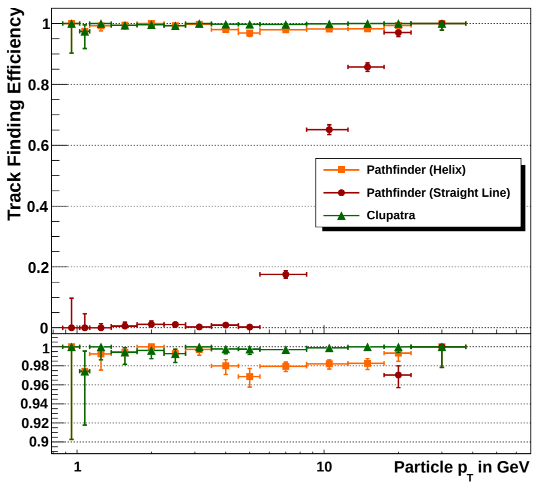


Figure 8.16: Track finding efficiencies for tau decays (3-prong events).

tracks in 3-prong events which are marked as not found correctly into a histogram (see Figure 8.17). In this plot an increase of entries can be seen which starts roughly at 0.75 and is going up to 1. Most of the tracks in this part are expected to be split tracks, all other ones most likely do not correspond to any simulated particle and are thus fake tracks. The fake rate and the split rate are shown in Figures 8.18 and 8.19, respectively. The fake rate is very low in all regions while the split rate is higher in the region between 3 and 20 GeV. Thus the tracks producing the low efficiency obviously are split into two or more parts.
One explanation for the increased split rate could be that two tracks are too close to each other to be separated properly. As shown before, the track finding efficiencies drop for close tracks (see Figures 8.8, 8.9 and 8.10). To investigate this further the track finding efficiencies for different angles between the tracks closest to each other are plotted in Figure 8.20. The efficiency for `Pathfinder` assuming a helix is almost

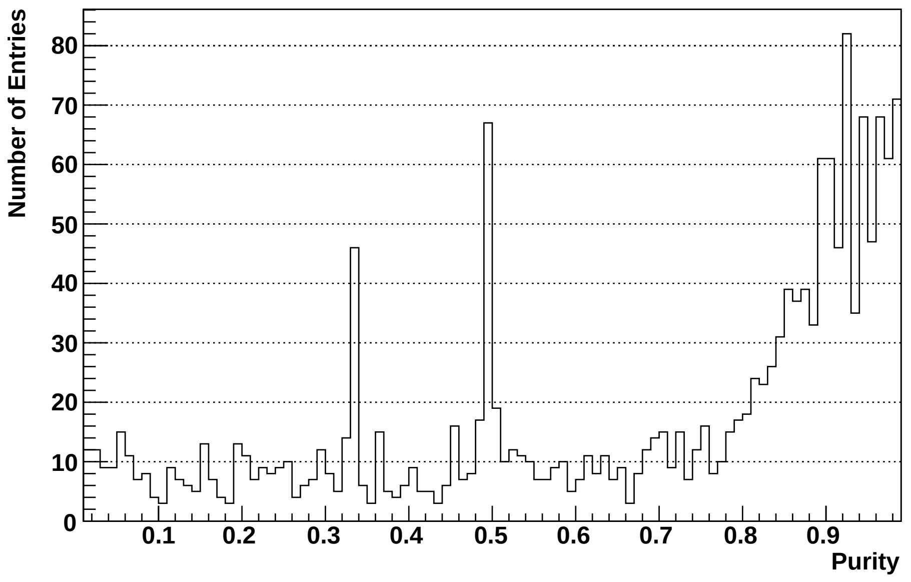


Figure 8.17: Purity of tracks in 3-prong events which are marked as not found correctly.

constant at about 90-95% over the full range. For angles between eight and nine degrees the statistics is getting low and thus the uncertainties on the efficiencies are high. So tracks being too close to each other cannot be the explanation for the high split rate in the 3-prong events because no significant drop appears in any region.
Another explanation could be that one of the particles by chance ends up in the cathode in the center of the ILD TPC. To prove this hypothesis the efficiencies for different angle between the initial momentum of the particle and the xy plane is plotted (Figure 8.21). But also here no significant dependence of the track finding efficiency on the angle can be found for `Pathfinder` assuming a helix. Also here the statistics gets low for large angles. That is why the uncertainties are large for the efficiencies in this region. So particles ending in the cathode of the TPC cannot be the explanation for the high split rate either.
Despite the great effort and plotting various quantities no explanation for the low efficiencies (and corresponding to this high split rates) in the p_T region between 4 and 15 GeV in 3-prong events could be found. It was shown that the tracks are split into several parts but no obvious reason could be found why this happens only in 3-prong events and not in 1-prong events and at rather high transverse momenta where the losses of energy do not change the curvature significantly.

8.4 Summary

In this chapter it was shown that with `Pathfinder` tracks created by a Monte Carlo simulation based on geant4 can be reconstructed reliably. This was tested on single muon events, events containing two muons and tau decays. In most cases the track finding efficiencies are 100%. Most of the exceptions are understood and are caused by either energy losses or the tracks were too close to each other to be recognized

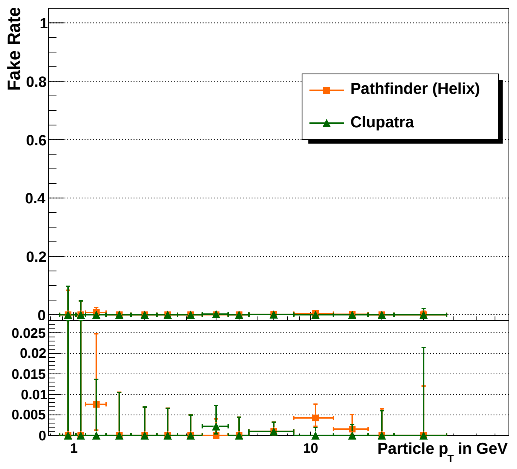


Figure 8.18: Fake rate for 3-prong events.

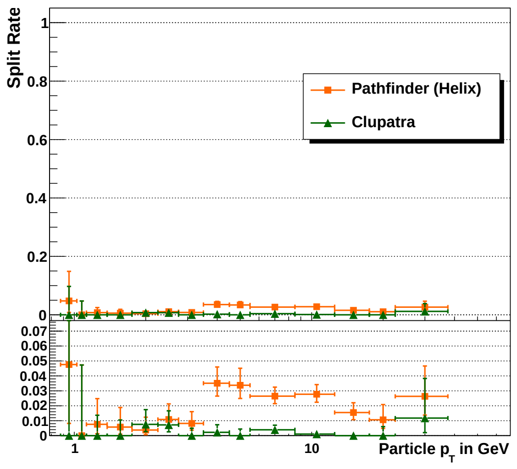


Figure 8.19: Split rate for 3-prong events.

as two individual tracks. Only in 3-prong tau decays the track finding efficiency is rather low for transverse momenta between 4 and 15 GeV, about 98%, which is caused by a large split rate. This can be resolved by applying a track matching algorithm which merges tracks that are found in two or more segments. All in all the track finding efficiencies in these simple events are better than those planned for $t\bar{t}$ events in the ILD of 99.7% [27].

Although `Pathfinder` gives back a rough estimate of track parameters, the momentum resolution achieved matches the one required for the ILD Detector of 1×10^{-4} GeV^{-1}. Only exceptions are tracks with small momenta, where losses of energy have a big influence, and for small θ, where not the full track can be measured because the particle leaves the TPC through the anode of the TPC.

The results show that tracks can be found in realistic events, however the situations studied are not very complex. To run `Pathfinder` to reconstruct tracks in more complex events such as $t\bar{t}$ events (several tens of tracks expected), however, is not feasible in the current implementation because computing time increases rapidly with increasing number of hits in the event.

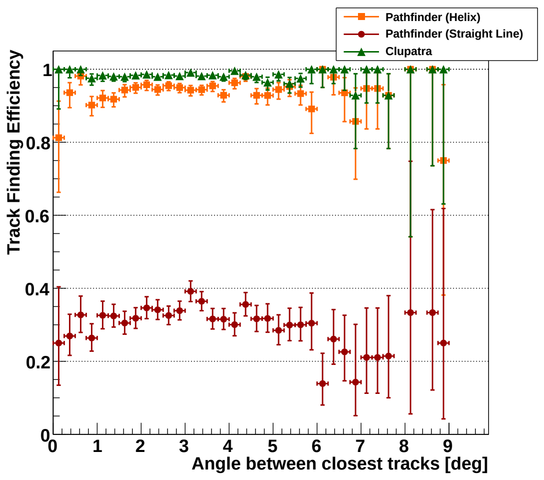


Figure 8.20: Track finding efficiencies for 3-prong tau decays for different angle to the closest track.

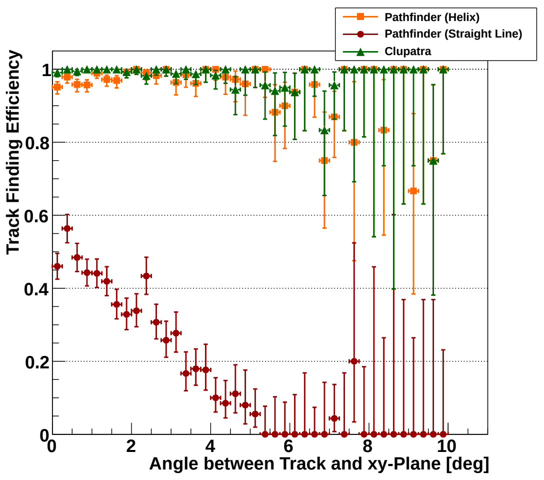


Figure 8.21: Track finding efficiencies for 3-prong tau decays for different angle between track and xy-plane.

Chapter 9

Resolution Measurement with Testbeam Data

For the ILD TPC good momentum resolution is required, which was motivated in Chapter 3.3. Since the achievable momentum resolution depends on the achievable single point resolution, readout structures are needed which can deliver the required values for this quantity.
In order to prove that a TPC with the required resolution can be built a large prototype TPC was built in which readout modules as they could be used in the ILD TPC are tested. In this chapter the analysis of testbeam data taken with a readout module using GEM amplification and pad readout is presented which aims at determining the single point resolution which can be achieved with this module.

9.1 The Setup

9.1.1 The DESY Testbeam

At DESY there are three testbeam lines available. How the beams are created is sketched in Figure 9.1 and described in [98]. The electron-positron synchrotron DESY II creates bremsstrahlung. Bremsstrahlung beams are created by carbon fibers in three different places as can be seen in Figure 9.2. The photons are converted to electron-positron pairs using a metal plate. After that the beam is spread by a dipole magnet. The final beams are cut out by collimators and are sent to the testbeam areas (T22, T23, T24 and T24/1, T24 and T24/1 share the same beam line). Beam energies between 1 and 6 GeV are available.
The data analyzed here were taken at the testbeam area T24/1 with a beam energy of 5 GeV.

9.1.2 The Large Prototype

The Large Prototype [88] is a TPC with a diameter of 77 cm and a length of 61 cm. Its endplate is designed in such a way, that it can hold up to seven modules which have the same shape as the modules currently foreseen for a future TPC for the International Large Detector. The Large Prototype is designed to fit into a

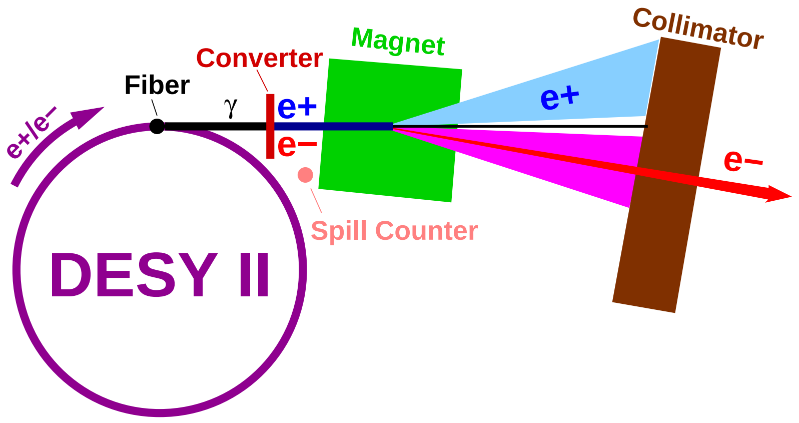


Figure 9.1: Layout of the DESY Testbeam. Picture taken from [99].

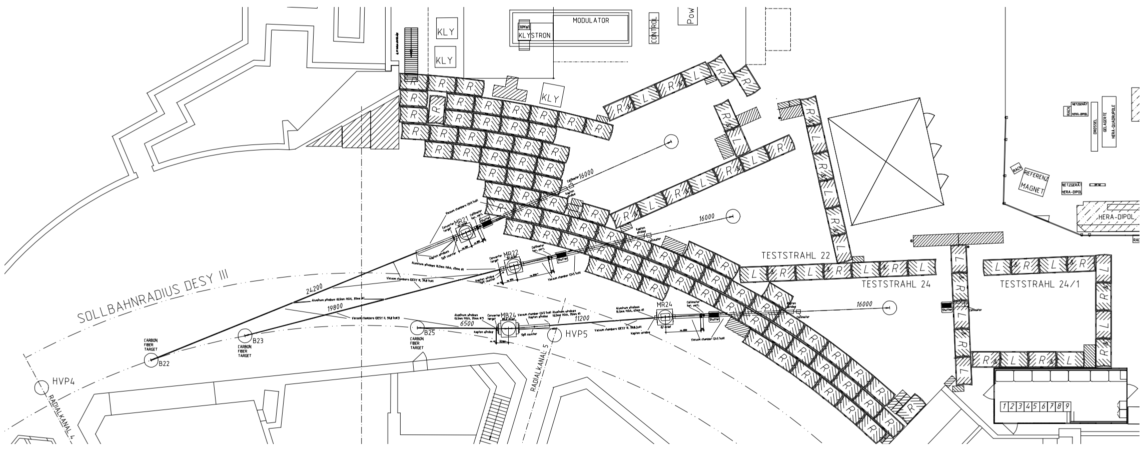


Figure 9.2: DESY Testbeam Areas. Drawing was done by Norbert Meyners and taken from [99].

superconducting magnet called PCMAG (see Section 9.1.3 for more details).
In the TPC T2K gas was used. This gas mixture consists of 95% Argon, 3% CF_4 and 2% Isobuthane (iC_4H_{10}). A drift field of 220 V/cm was chosen for all runs (excluding the drift field scans). As can be seen in Figure 9.3 this is the drift field for which the transverse diffusion is minimal. The gas mixture was chosen because the drift velocity is relatively high.

9.1.3 The Magnet

The PCMAG is a superconducting magnet which can provide a field of 1T. Figure 9.4 shows a drawing of the cross section of the magnet and its magnetic field. The blue rectangle represents the Large Prototype TPC. During data taking it is positioned centrally in the magnet where the field is most homogeneous.

9.1.4 The Module

The data are taken with a module [100] consisting of a stack of three GEMs which are supported by 1 mm thick grids made from ceramics which at the same time

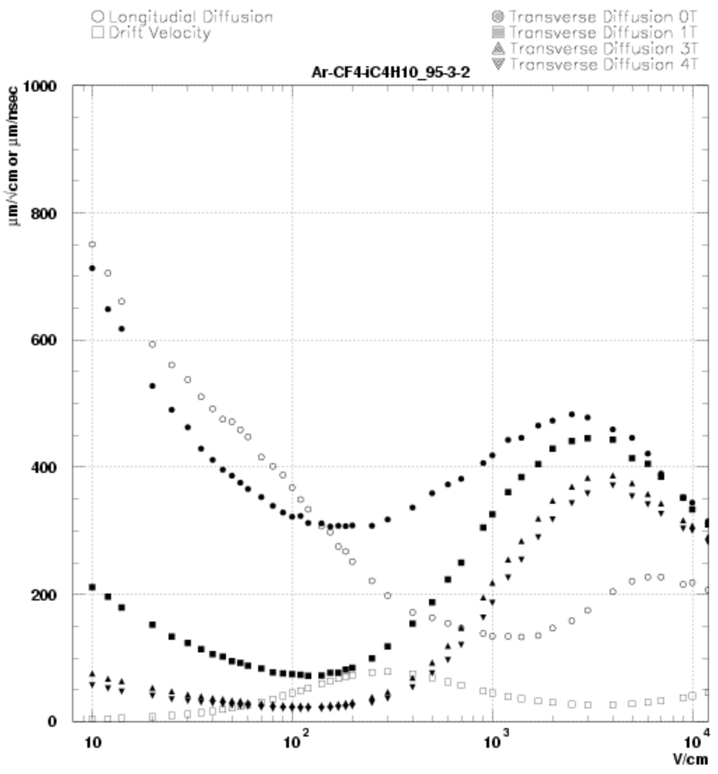


Figure 9.3: Gas properties in dependence of the drift field strength simulated with `MAGBOLTZ` for T2K gas [71].

serve as spacers. The module is divided into quadrants (so called sectors). An exploded view of the module is shown in Figure 9.5. The GEM facing the drift volume is called the drift GEM, the GEM closest to the pad board is referred to as anode GEM. The third GEM is the middle GEM. Each GEM has two sides which is referred to as anode side and cathode side depending on whether they face the anode or the cathode of the TPC. The gap between two GEMs (transfer gap) is 2 mm wide while the gap between the anode GEM and the pad board (induction gap) is 3 mm wide. The voltage settings are given in Table 9.1. They are chosen such that for each GEM the potential between cathode side and anode side is the same. The drift field in the transfer gaps is 1500 V/cm each and in the induction gap it is 3000 V/cm. In the past, GEM stacks with this configuration proved to work

GEM, Side	**Voltage [V]**
Anode GEM, anode side	900
Anode GEM, cathode side	1150
Middle GEM, anode side	1450
Middle GEM, cathode side	1700
Drift GEM, anode side	2000
Drift GEM, cathode side	2250

Table 9.1: Voltages applied to the GEMs.

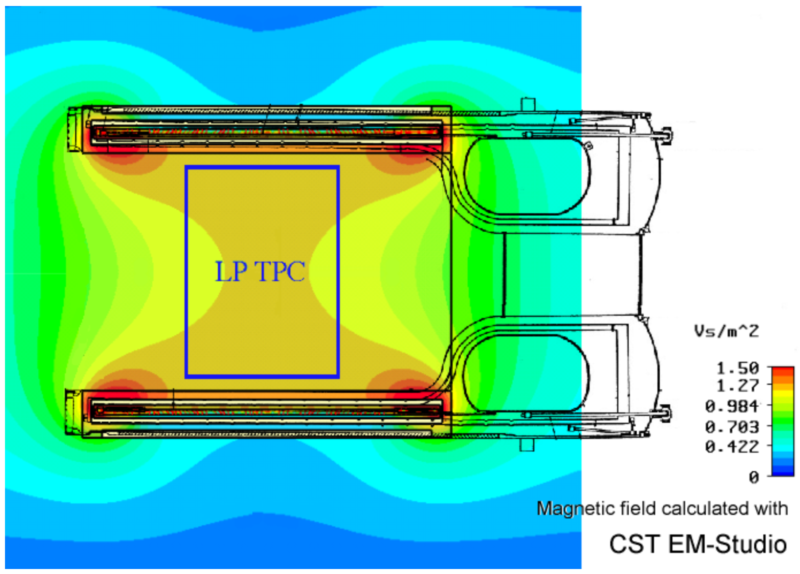


Figure 9.4: A drawing of the PCMAG and the magnetic field [35].

Property	Value
Kapton layer	50 µm
copper layer	5 µm
hole pattern	hexagonal
hole diameter	70 µm
pitch	140 µm

Table 9.2: Properties of standard CERN GEMs [50].

reliably. The induction field (between anode GEM and pad board) is chosen such that the diffusion is maximal. Between the GEMs (transfer fields) the field is chosen such that the extraction and collection efficiency (see Chapter 4.2) is maximal. The GEMs used for the module have, apart from the size of the foils, the same properties as standard CERN GEMs [50]. The properties are summarized in Table 9.2.
The pad board has a fine segmented region in the center of the board while it is covered with larger pads in all other regions. A photography showing the different pad sizes is shown in Figure 9.6. Only the small pads, which are $1.2 \times 5.7\text{mm}^2$ large, are read out. In total there are about 1000 such pads in 28 rows. The module is inserted into the Large Prototype which is positioned in such a way that the electron beam is perpendicular to the pad rows and that the charge is deposited on the small pads in the center of the module.
The module is read out with the Altro electronics [101].

9.2 The Data

In order to investigate the dependence of the resolution on the drift distance several runs were taken at different z positions. In particular a number of runs were taken

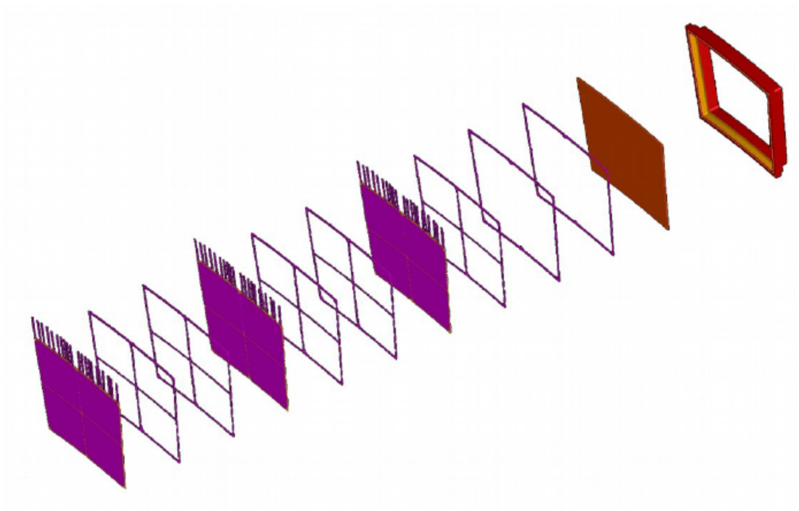


Figure 9.5: Exploded view of the DESY GEM module [100].

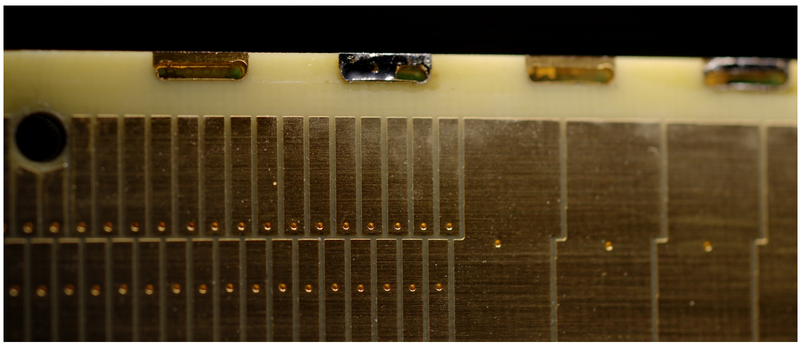


Figure 9.6: Photography of the pad board of the readout module. The center part is equipped with small pads, the outer parts are covered with larger pads which are not read out [100].

close to the anode and close to the cathode to find their exact position. However, to avoid shooting the beam into the amplification structure the distance of the beam to the anode was not smaller than about 1 cm, while the beam was shot directly into the cathode. Those runs close to the anode and close to the cathode are not used for the resolution studies but only for finding the cathode and anode position. All these runs were taken without magnetic field.
A number of runs were also taken with magnetic field but since parts of the module broke during data taking only one of the four sectors of the module was available and only very few data are available. As shown in Section 9.4.6 those data still look reasonable. A z position scan for resolution studies and a drift field scan are available with magnetic field.
A list of all runs can be found in Appendix E.

9.3 The Reconstruction

The raw data taken with the setup described above consist of charge spectra for each pad which means that for each pad the charge collected over time is recorded. In these spectra so called pulses need to be found. These are regions in the charge spectrum where charges exceed a given threshold are collected. These regions are called pulses. Pulses with a minimum height (8 ADC counts) and minimum length (5 time bins) are selected to reduce the amount of noise signal. Pulses measured on neighboring pads in a row need to be combined to so called hits, which are points in 3D-space. At least two pulses must be in the hit to accept it. Finally tracks need to be found. A more detailed description of the reconstruction algorithms can be found in Chapter 5.1.
The reconstruction of the Large Prototype data is done with `MarlinTPC`. The steering parameters for the processors used for the final reconstruction are given in Table 9.3. Some of the values need to be determined from the data (drift velocity, time shift and time cuts). Therefore the reconstruction is first done with default values for the drift velocity (80 mm/µs) and no time cut or time shift up to hit level. This first reconstruction is used to calculate the needed values. More details on the procedure are given in sections 9.4.1 and 9.4.3.
From the drift length, the drift time and the readout frequency the number of time bins in the TPC is calculated. As start offset the time bin corresponding to the uppermost GEM in the TPC is chosen. The value for the end offset is chosen according to the shaping time of the electronics (120 ns). Every measurement being outside the TPC is cut off. After that the time is shifted by 10 bins such that the time for the uppermost GEM is zero. On the corrected raw data the pulse finding and hit finding is done. From the charges collected on each pad points in 3D-space are reconstructed.
The last step in the reconstruction is the track finding which runs on hits. `Pathfinder` (see Chapter 7) was used for this part of the reconstruction. The hits in some rows are excluded from the track finding. The rows at the borders of the module (rows 0,1,24-27) are cut out because almost no charge was collected there due to field distortions [102]. Two rows in the center of the module (rows 13,14) are cut out because of low measured charges as well. This is caused by the ceramic grid supporting the GEMs at this position. Finally rows 6 and 10 are cut out. In these rows pads did not work properly which lead to a wrong hit reconstruction. Before the track finding is done all hits are shifted closer to the origin[1]. After track finding the hits are shifted back to their original positions and the reference point of the track is changed accordingly. Figure 9.7 shows why this is necessary. This is needed because the algorithm suffers from numerical uncertainties. The track found without shifting the hits deviates strongly from the positions of the hits. If shifting the hits first the track agrees with the hit positions.

[1]The origin of the coordinate system is shifted such that it is positioned in the center of the readout module.

Steering Parameter	**Value**
Cut on raw data outside TPC:	
Drift Length	569.37 mm
Drift Velocity	75.24 mm/µs
Readout Frequency	20 MHz
Start Offset	10 bins
End Offset	5 bin
Long pulses are cut off at the end of the TPC.	
Time Correction:	
Time Shift	10 bins
Pulse Finder:	
Minimum pulse height	8 ADC counts
Minimum pulse length	5 time bins
Pulse start threshold	6 ADC counts
Pulse end threshold	6 ADC counts
Number of bins saved before start	3
Number of bins saved after end	7
Remove unwanted pulses:	
Charge variation	6 ADC counts
Hit Finder:	
Maximum number of empty consecutive pads	1
Maximum time between pulses in hit	500 ns
Minimum number of pads	2
Minimum height of maximum pulse in hit	12 ADC counts
Drift velocity	75.24 mm/µs
Rows cut out for track finding: 0, 1, 6, 10, 13, 14, 24, 25, 26, 27	
Track Finder (with cuts):	
Track type	Straight line
Number of bins	1000 (all directions)
Maximum distance of the hits to the track in xy	1.5 mm
Maximum distance of the hits to the track in sz	1 mm
Minimum number of hits on track	5
Maximum range of Hough space in xy	4000 mm
Maximum range of Hough space in sz	4000 mm
Track Finder (without cuts):	
Track type	Straight line
Number of bins	1000 (all directions)
Maximum distance of the hits to the track in xy	1.5 mm
Maximum distance of the hits to the track in sz	1 mm
Minimum number of hits on track	5
Maximum range of Hough space in xy	4000 mm
Maximum range of Hough space in sz	4000 mm

Table 9.3: Steering parameters used for the reconstruction of test beam data.

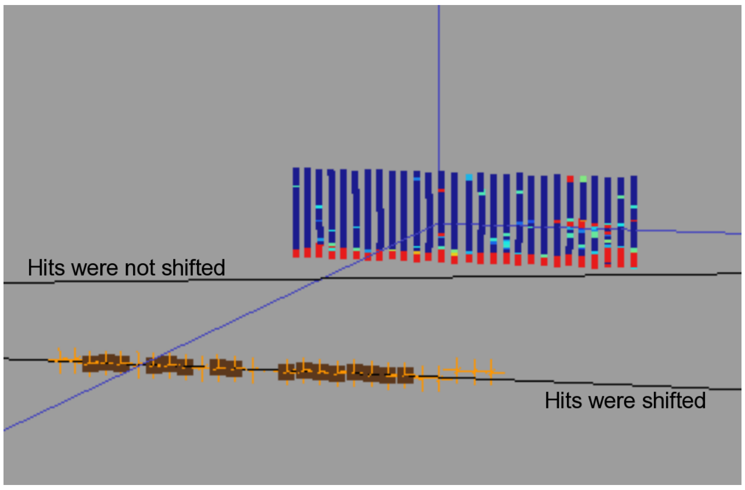


Figure 9.7: The event display shows the hits reconstructed in one testbeam event (orange crosses) and those hits having been assigned to the track. Additionally, as solid black lines two tracks are shown. One was reconstructed without shifting the hits closer to the origin, one was found after having shifted the hits. The first one does not match the hit positions while the latter one does.

9.4 Analysis

9.4.1 Anode and Cathode Position

Before any measurements can be done the endpoints of the time projection chamber have to be calculated. A number of runs are taken close to the anode and close to the cathode. The reconstruction is done up to pulse level. The pulse times for these runs are plotted in the histogram shown in Figure 9.8. In an ideal case one would expect the runs at the anode to have pulse times at zero because the electrons do not have to drift. However in reality not only the drift time in the drift volume is measured but also the time the signal needs between the GEMs, the time between the anode GEM and the pad board and the time the signal needs in the cables and electronics. Run 17705 and runs 17713 through 17717 are taken close to the cathode of the TPC. Run 17713 is rather far away from the cathode. The pulse time distribution shows a Gaussian shape because the particles did not interfere with the cathode. For all other runs the pulse time distribution is asymmetric. They have a sharp edge at the position of the cathode, which is used to calculate the time corresponding to the cathode position. For the cathode runs a lot of pulses are measured far away from the cathode in the drift volume of the TPC. This is caused by secondary particles produced by the beam in the cathode. The pulse time distributions for the anode runs (run 17708, 17709, 17718 and 17719) do not show any such strong asymmetric behavior. If at all it is only visible for the run closest to the anode (run 17709). The reason for this is that no data closer than about 10 mm from the drift GEM are taken in order to protect the readout structure (the beam width is about 1 cm). In the following the anode and cathode runs are analyzed in more detail.
To determine the time corresponding to the cathode position only the runs 17714 through 17717 are taken into account because run 17713 is not close to the cathode

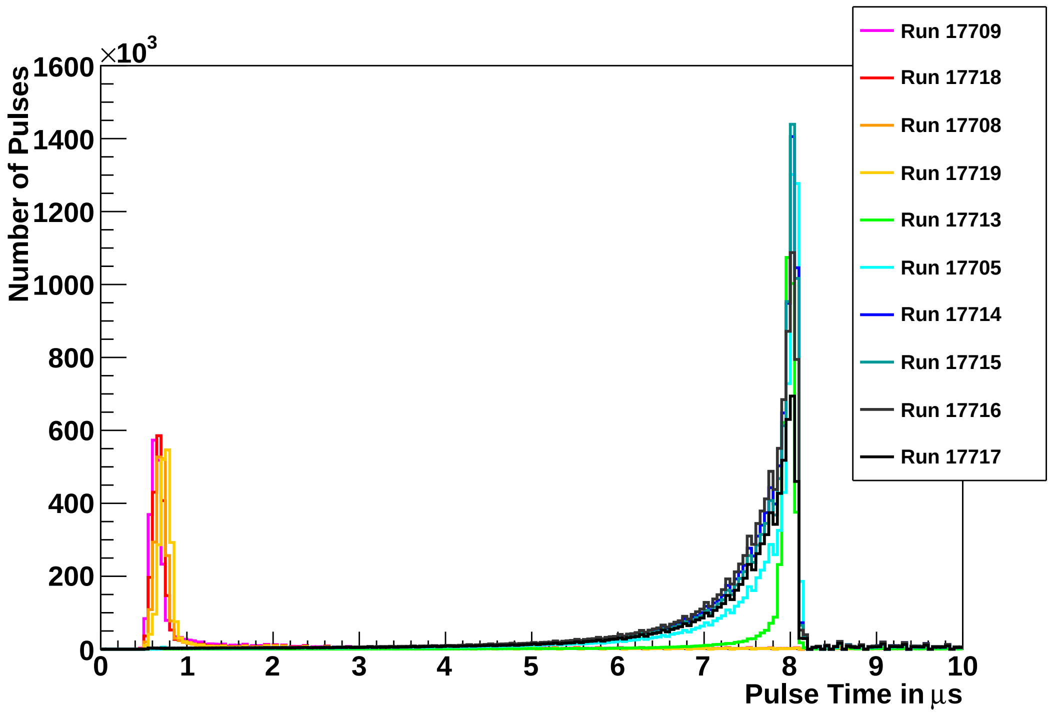


Figure 9.8: Pulse times of runs close to the anode (runs 17708, 17709, 17718, 17719) and close to the cathode (run 17705 and runs 17713 - 17717).

and run 17705 only hit it slightly. The pulse times of the runs are plotted in one histogram and the cathode time is calculated from the mean of the falling edge as described in [69] by using the derivative of the pulse time distribution as shown in Figure 9.9. The result obtained is

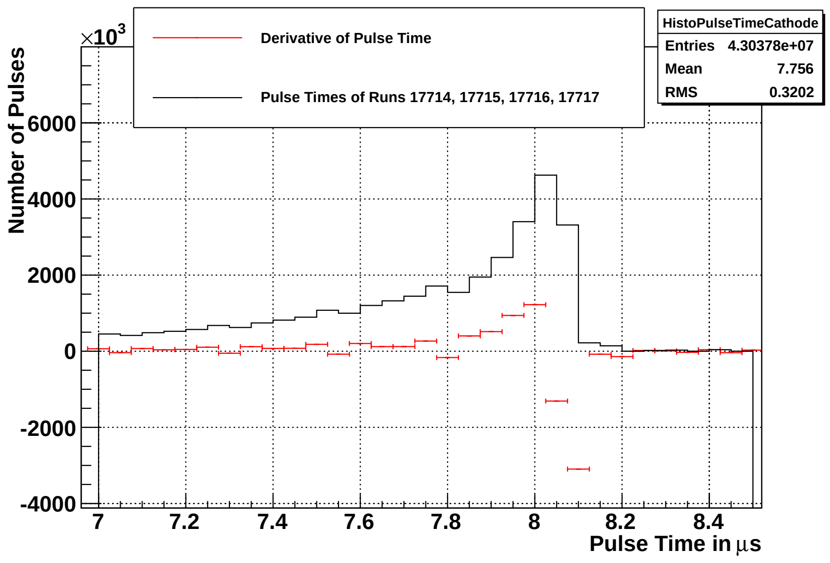


Figure 9.9: Pulse times for runs 17714 through 17717 and the derivative of the pulse time distribution.

$$t_{\text{Cathode}} = 8.09 \pm 0.202 \text{ (stat) µs}$$

Calculating the anode time is more challenging because there is no sharp edge as for the cathode. In the following only run 17709 is taken into account because that is the one closest to the anode. Assuming that some of the particles hit the drift GEM or crossed between the drift GEM and the middle GEM one would expect pulses with rather low charges because one amplification stage is missing. The pulse reconstruction is done again with steering parameters allowing detection of such small pulses. This is shown in Figure 9.10. The pulse start and end threshold

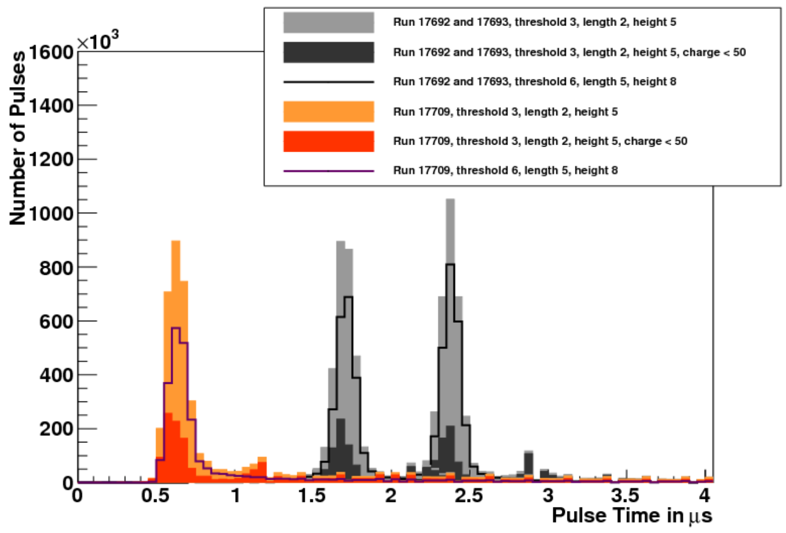


Figure 9.10: Pulse times for run 17709 (anode run) and runs 17692 and 17693 for different parameter used for pulse finding.

(3 instead of 6 ADC counts) as well as the minimum length (2 instead on 5 time bins) and minimum height (5 instead of 8 ADC counts) of the pulses are set to lower values. Additionally a cut on the pulse charge is applied. Only pulses with a charge less than 50 ADC counts are plotted. One can then see the asymmetric shape of the pulse time distribution. For comparison the same selection is done for two runs (17692 and 17693) far in the drift volume where the shape of the pulse time distribution is symmetric after all steps of the selection.
However this only gives a rough idea which time corresponds to the anode position. To get a better estimate on the anode time one can have a look at the cathode runs again. Run 17705 slightly touches the cathode. This run is taken at a z-Position of 276.1 mm (table position). The length of the drift volume of the Large Prototype is measured to be 569.4 mm. From this one can calculate that the anode must be at a table position of -293.2 mm. Run 17709 is taken at a table position of -282.6 mm. This gives about 10.6 mm which are not scanned. From a `MAGBOLTZ` simulation for T2K gas and a drift field of 220 V/cm a drift velocity of 72.89 mm/µs is obtained. A Gaussian fit to the pulse time distribution for run 17709 gives a mean pulse time of $t_{17709} = 0.64576 \pm 0.00005$ µs. From the simulated drift velocity and the mean pulse time for run 17709 on can then estimate that the anode time is roughly 0.49951 ± 0.00005 µs.

9.4.2 Abnormal Pulse Shapes

A closer look at the reconstructed pulses showes that there are a number of pulses which have extremely high charges. Such pulses start off as normal pulses but do not drop under the pulse end threshold. Rather the charges stay on a constant level as sketched in Figure 9.11. The long tail is split into many small pulses by the

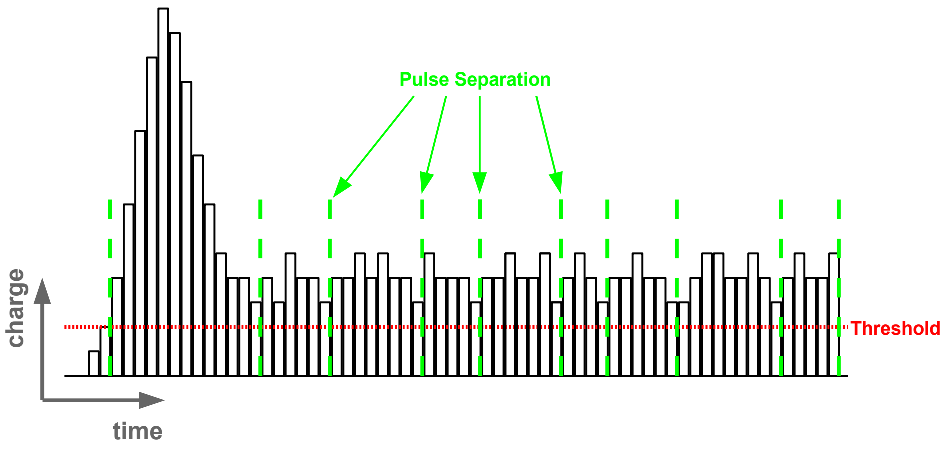


Figure 9.11: A sketch on how the charge develops for high pulses (not to scale). The pulse starts like a normal pulse but charges do not drop under the pulse end threshold (dotted line). The pulse finding algorithm splits the tail of the distribution into many small pulses (dashed lines).

pulse finding. Since the tail reaches beyond the end of the TPC many pulses are reconstructed outside the TPC. Most of the pulses are cut out by time cuts, but still few of these pulses (which were reconstructed inside the TPC) remain and need to be removed.

Pulses being split from a large pulse show similar shapes which deviates from the normal pulse shape. The charge in the bins does not vary much. Plotting this variation (difference between maximum and minimum charge value in the pulse) shows a rise in number of pulses with low variations with a maximum at about 2 ADC counts, see 9.12, while there is a second broader maximum at about 17 ADC counts. The pulses in the lower peak are expected to be produced by pulses being split from a large pulse while the pulses in the broad peak are expected to be normal pulses. Thus to discriminate between the two types of pulses a cut is introduced. Only pulses with a charge variation larger than 6 ADC counts are accepted for the next reconstruction step. The minimum height of a pulse is set to be at least eight ADC counts which gives (assuming the charges starts from zero ADC counts and drops to zero ADC counts) a charge variation of at least eight ADC counts. It might also happen that the pulse starts and ends at one ADC count, which still gives a charge variation of seven ADC counts. Therefore requiring the charge variation of being larger than 6 ADC counts is a reasonable value.

To check if this selection gives the desired result the time of the pulses (with cut and without cut) are compared for different pulse qualities. Pulse quality 1 stands for normal pulses. Pulse quality 17 means that the pulse was split from another and

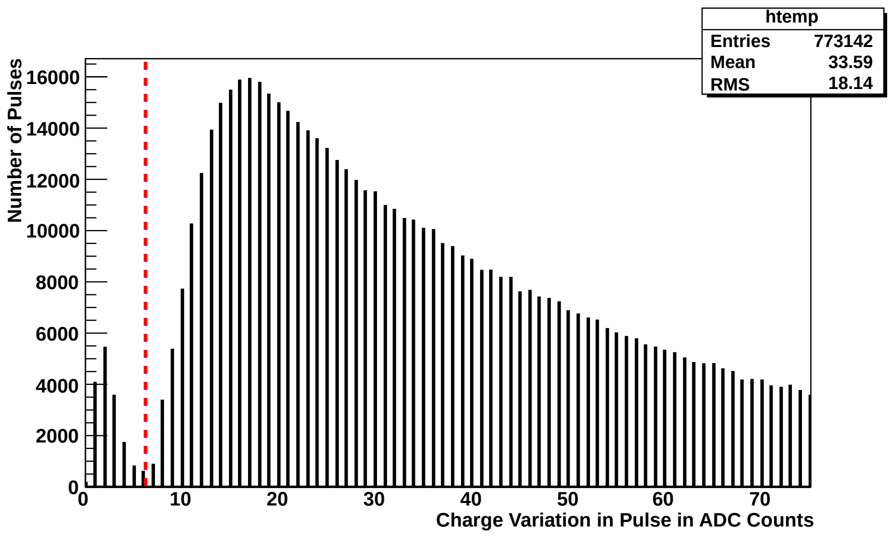


Figure 9.12: The charge variation (difference between maximum and minimum charge in pulse) within pulses. The first peak below six ADC counts is produced by pulses being split of from a very high pulse. Those pulses are cut out by requiring a charge variation of at least seven ADC counts (dashed line).

pulse quality 21 corresponds to very high pulses. In the left plot of Figure 9.13 the pulse times for pulses having one of these three qualities are plotted before the cut on the charge variation. The right plot of Figure 9.13 shows the pulse time for these

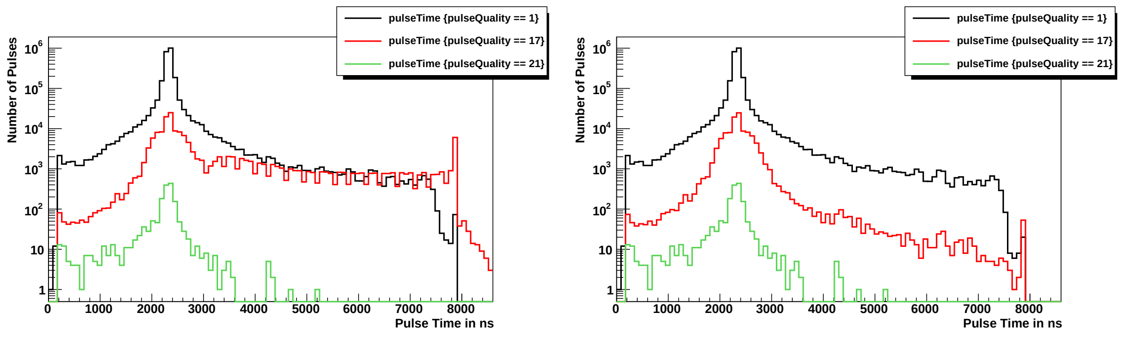


Figure 9.13: The histograms show the pulse time distributions for pulses with different quality flags before (left) and after (right) removal of pulses with low charge variation. Pulse quality 1 is a normal pulse, pulse quality 17 means the pulse is split from anther pulse and pulses with pulse quality 21 are pulses with high charge.

pulses after the cut on the charge variation. The distributions for normal pulses and pulses with large charge do not change significantly. For pulses being split from another the distribution changes significantly, as expected. Before the cut there are many split pulses with large times. The distribution is almost uniform. The number of these pulses is significantly reduced by the cut on the charge variation. Thus the cut on the charge variation is a good choice to remove the unwanted pulses.

9.4.3 Drift Velocity

To do a proper analysis it is essential to know the correct drift velocity because it is used to calculate the z-position of the hits which are than used for track finding. There are different ways how to calculate the drift velocity. They will be presented in the following. All methods have in common that the reconstruction must be done up to hit level assuming a drift velocity (here 80 mm/µs). The z-axis for the hit positions is defined such that z is zero at the cathode and is positive along the drift direction. With a correct drift velocity no negative values for z are possible. However, hits having been reconstructed with a drift velocity of 80 mm/µs do show negative values see Figure 9.15. This means the hits have traveled a larger distance than they could have. This corresponds to an overestimation of the drift velocity.
The first possibility to measure the drift velocity is to use runs taken at different z-positions. Each hit has a time (coming from the time of the maximum pulse in the hit). The times of all hits for runs at different z-positions are filled into histograms which are fitted with a Gaussian. Most of the hits are expected to be created by the electron beam. The z-position of the beam in the TPC is known from the measurement of the table position. One can now plot the true z-positions over the mean hit time (see Figure 9.14). This is a straight line and the slope is the drift

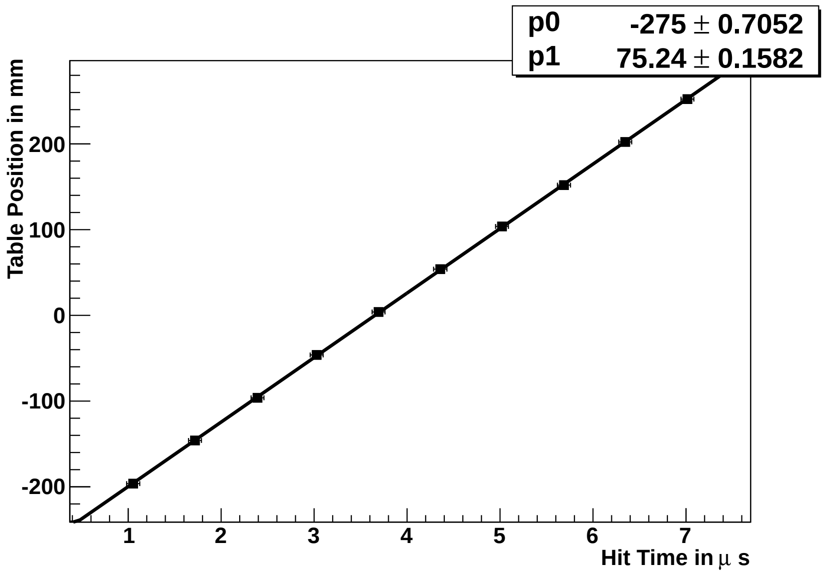


Figure 9.14: Drift velocity measurement. On the x-axis the mean time of the hits in each run is shown, on the y-axis the position at which the run was taken is plotted. Runs close to the anode and close to the cathode were not used. A straight line was fitted to the data. Its slope gives the drift velocity.

velocity. The result of a straight line fit gives a drift velocity of about 75.24 ± 0.16 mm/µs which is, as expected, a smaller value than 80 mm/µs.
All other methods do not use the time of hits but the time of pulses which in the end is not a big difference because the hit time is taken from the time of the largest pulse in the hit. The anode time and the cathode time calculated in the previous section are used. As seen in the previous section run 17705 touches the cathode slightly and run 17709 is the run closest to the anode. The cathode time is

Field [V/cm]	**Drift Velocity in mm/µs**
1500	35.23
3000	26.48
220	75.84

Table 9.4: Simulation of drift velocities for different fields in T2K gas done with `MAGBOLTZ` by Ralf Diener [103].

calculated to be $t_{\text{Cathode}} = (8.09 \pm 0.20)$ µs and the mean pulse time for run 17709 is $t_{17709} = (0.64576 \pm 0.00005)$ µs. The difference of the table positions for the two runs is $l_{\text{scanned}} = 558.7$ mm. From this one can calculate a drift velocity of (75.06 ± 0.20) mm/µs which is in agreement with the previous measurement.
Instead of using the mean pulse time of run 17709 on can use the anode time calculated previously $t_{\text{Anode}} = 0.49951 \pm 0.00005$ µs and the drift length of the TPC $l_{\text{drift}} = 569.37$ mm. This give a drift velocity of 75.01 ± 0.20 mm/µs. This method assumes that the time corresponding to the anode position is correct.
The last method only uses the cathode time. The measured time consists of the time needed in the drift volume, between the GEMs (induction gap), between anode GEM and pad board (transfer gap) and in the electronics:

$$t_{\text{meas}} = t_{\text{cable}} + t_{\text{ind}} + 2 \cdot t_{\text{trans}} + t_{\text{drift}}. \tag{9.1}$$

The time needed in the cables can be estimated. There is a total of 40 m of copper cable via which the signal has to travel. Each meter of cable has a delay of 5 ns. The total delay in the copper cables is 0.2 µs. The time needed in the electronics is still not taken into account, so this time is larger in reality. The time the signal needs inside the TPC can now be calculated

$$t_{\text{TPC}} = t_{\text{meas}} - t_{\text{cable}}. \tag{9.2}$$

The drift velocities in the transfer gap between the GEMs and in the induction gap between the anode GEM and the pad board are different from the drift velocity in the sensitive volume of the TPC because different drift fields are present. To take this into account the effective length of the TPC is calculated using simulated drift velocities for the different gaps [70].

$$\begin{aligned}
t_{\text{TPC}} &= t_{\text{ind}} + 2 \cdot t_{\text{trans}} + t_{\text{drift}} && (9.3)\\
\frac{l_{\text{eff}}}{v_{\text{drift}}} &= \frac{l_{\text{ind}}}{v_{\text{ind}}} + 2 \cdot \frac{l_{\text{trans}}}{v_{\text{trans}}} + \frac{l_{\text{drift}}}{v_{\text{drift}}} && (9.4)\\
l_{\text{eff}} &= \frac{v_{\text{drift}}}{v_{\text{ind}}} l_{\text{ind}} + 2 \cdot \frac{v_{\text{drift}}}{v_{\text{trans}}} l_{\text{trans}} + l_{\text{drift}} && (9.5)
\end{aligned}$$

The idea behind this is that even if the real drift velocity is not the same as the simulated one (due to different water and oxygen contents or different voltages) the ratios between the different drift velocities are approximately the same. The drift velocities are simulated with `MAGBOLTZ` and are given in Table 9.4: The effective length is $l_{\text{eff}} = 585.904$ mm. This gives, together with the estimated time needed in the cables and the cathode time, a drift velocity of 74.259 ± 0.188 mm/µs. This

value is lower than the ones obtained with the previous methods which means that the time needed in the cables and electronics is underestimated.
In the following the drift velocity from the first method (straight line fit) is used which is expected to be the most reliable one because it is based on many measurements. Figure 9.15 shows how the z-positions of the hits change when using different values for the drift velocity.

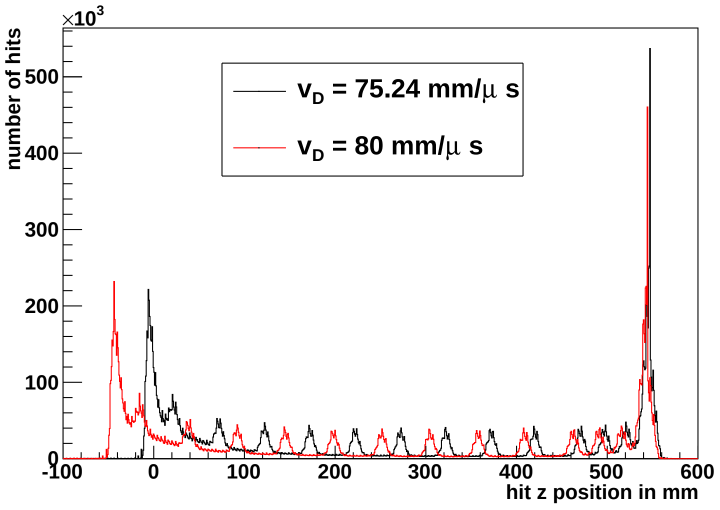


Figure 9.15: The plot shows the reconstructed hit positions for two different drift velocities integrated over all all runs taken without magnetic field. The rest of the reconstruction is done exactly the same way. The small regular peaks are produced by the runs taken at different z-positions.

9.4.4 Correction for Distortions

The distance between the hits and the found track is expected to be centered around zero for each row. However, due to field distortions which are strongest at the borders of the module this is not the case in the testbeam data analyzed here. The distortions broaden the distributions of the distances between hit and track and thus worsen the resolution. For this reason it is necessary to correct for them. In general this is done by shifting the hits towards the track. Then the track finding is done again. To do this the amount by which the hits need to be shifted has to be determined from the data. There are two approaches how this could be done. One method is to determine the distortions integrated over all runs. For all runs the distances[2] between hit and track are filled into a histogram for each row. The amount the hits need to be shifted is the mean of the histograms. This method only works if the distortions have similar shapes in all runs. This is true for the distortions in the xy-plane for the data analyzed here, but the distortions in z show a dependence on the drift length, as is shown in Figure 9.16.

[2]In this case in the xy-projection the distance between hit and track along the rows is used. However in the setup used there is no significance between the distance along the row and the distance in y or the shortest distance in the xy-plane.

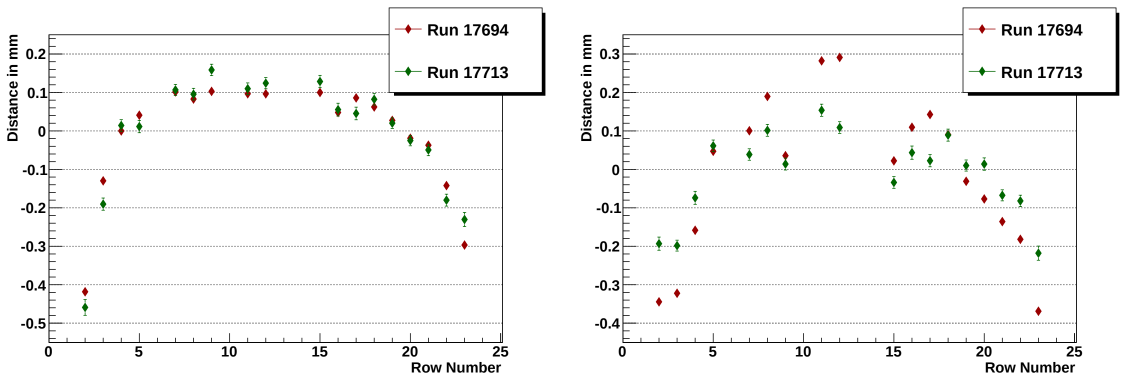


Figure 9.16: Distortions (distance between hit and track) in xy (left) and z (right). One run close to the anode (Run 17694) and one run close to the cathode (Run 17713) are shown. While in xy there are no large differences between the runs, there are huge differences in z.

Another possibility is to determine the distortions for each run individually by filling the distances between hit and track of the first 1000 events into a histogram of each row and determining the correction from this. In the further analysis (like single point resolution) the first 1000 events must then be excluded in order to not introduce any bias.
Both methods give the same result for the corrections in the xy-plane. Thus the two methods are equivalent. Since the distortions in z do depend on the drift length the second method is used in the following where the distortions are determined for each run individually.

9.4.5 Resolution Studies without Magnetic Field

9.4.5.1 Theory

In TPCs with pad readout a charge distribution along the pad rows is measured. The width of the charge distribution in the readout plane mainly depends on the transverse diffusion, which again depends on the gas used in the TPC and the electric field. Contributions from the readout system also affect the resolution (the hole pattern of the GEM and diffusion between the GEMs, pad layout, the thresholds in the reconstruction). In z-direction the width of the signal depends on the longitudinal diffusion and the shaping time used in the electronics [104]. The amount of diffusion is larger for larger drift distances hence a drift distance scan is performed to calculate the single point resolution across the full drift volume. From this charge distribution in each row hits are reconstructed. The single point resolution is a measure for the spread of the measured hits around the true track [105].
To calculate the single point resolution one would determine the distance between hits and true track and take the width of the distribution as the single point resolution. However the problem is, that it is unknown where the true track passed the detector because no external reference is available. Thus it is necessary to use a track fitted to the reconstructed hits instead of the real track. However, each

hit pulls the track towards itself, thus the single point resolution obtained using distances between hit and track is too small. To overcome this effect a hit can be removed from the track and a new track fit is performed with the remaining hits. The distance between the excluded hit and the new track is calculated (this will be called residual in the following). The hits still included in the track pull the track away from the excluded hit. Thus the calculated distance is too large. In the Appendix of [105] it is shown that for straight lines the true resolution is

$$\sigma_{\text{resolution}} = \sqrt{\sigma_{\text{distance}} \cdot \sigma_{\text{residual}}}, \tag{9.6}$$

where σ_{distance} is the width of the distribution of the distances and σ_{residual} is the width of the distribution of the residuals.

9.4.5.2 Event Selection

For the resolution study 1-track events are selected. An additional requirement is that each track has at maximum 1 hit per row. Additionally only such tracks are selected which are almost parallel to the readout plane which corresponds to cutting out all tracks with $|\tan\lambda| > 0.1$. This cut is done to ensure that the drift length is the same over the full length of the track.

9.4.5.3 Resolution Analysis in xy

The resolution in y-direction (along the pad rows) is calculated according to equation 9.6 for each run. The resolution for different drift length is shown in Figure 9.17 with and without corrections for distortions. The correction improves the resolution significantly. The function [105]

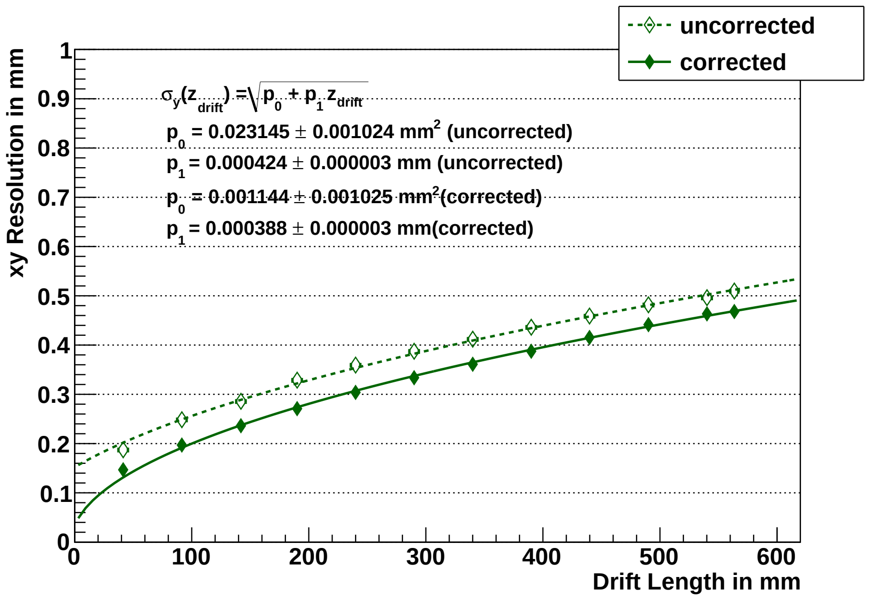


Figure 9.17: Resolution in xy for different drift length with and without corrections for distortions.

$$\sigma_{xy}(z_{\text{drift}}) = \sqrt{p_0 + p_1 \cdot z_{\text{drift}}} \tag{9.7}$$

is fitted to the data, where

$$p_0 = \sigma_0^2 \tag{9.8}$$

and

$$p_1 = \frac{D_T^2}{N_{\text{eff}}}. \tag{9.9}$$

N_{eff} is proportional to the number of electrons produced along one pad row. $\frac{D_T^2}{N_{\text{eff}}}$ describes how the resolution rises with increasing z. D_T is the transverse diffusion coefficient and can be obtained from a `MAGBOLTZ` simulation to be $D_T = 304.363 \pm 3.547 \frac{\mu\text{m}}{\sqrt{\text{cm}}}$ [103], so that N_{eff} can be calculated. σ_0 shifts the function to match the measurement. In principle it gives the point resolution at zero drift length and thus describes effects during amplification. The two parameters in the fit are, however, correlated as can be seen in Figure 9.18. The two variables thus loose their original meaning since they can have a wide range of values all describing the measurements. The minimum of the χ^2 is rather broad and in the case without corrections the minimum actually found (represented as a black dot) is not at the place were it would be expected. But still, the fitted function describes the data well. To investigate the stability of the fit, σ_{xy}^2 was plotted for each drift length (not

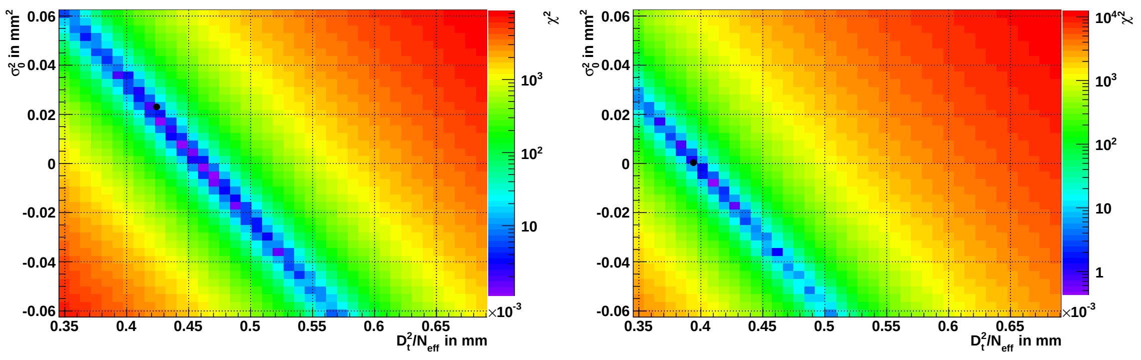


Figure 9.18: Plots showing the correlation of the parameters in the resolution fit. Left: no correction for distortions. Right: With correction for distortions.

shown here) and a straight line was fitted. The result and the χ^2 distribution does not change much.
The fit results and the values for N_{eff} and σ_0 calculated from that are summarized in Table 9.5. The two values for N_{eff} are in agreement while the values for σ_0 show a large deviation. Although the two variables are strongly correlated on can assume that the N_{eff} gives a good value for the number of electrons produced along each pad row. This is expected because the number of produced electrons does not change by correcting distortions.

	p_0 [mm^2]	p_1 [µm]	σ_0 [mm]	N_{eff}
no correction	0.023 ± 0.001	0.424 ± 0.003	0.152 ± 0.003	21.848 ± 0.532
with correction	0.0011 ± 0.0010	0.388 ± 0.003	0.034 ± 0.015	23.875 ± 0.586

Table 9.5: xy-resolution fit results.

9.4.5.4 Resolution Analysis in z

The resolution in z is calculated analog to the y-resolution for different drift length and is shown in Figure 9.19. The dependence of the drift length is less strong

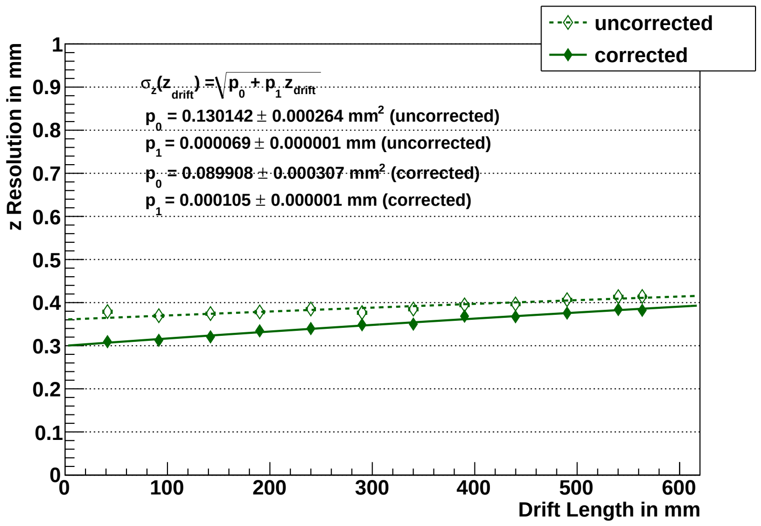


Figure 9.19: Resolution in z for different drift length with and without corrections for distortions.

compared to the y-resolution. This is due to the lower longitudinal diffusion of $D_L = 233.17 \pm 4.22 \frac{\mu\text{m}}{\sqrt{\text{cm}}}$ [103]. The function

$$\sigma_z(z_{\text{drift}}) = \sqrt{p_0 + p_1 \cdot z_{\text{drift}}} \tag{9.10}$$

is fitted to the data with

$$p_0 = \sigma_0^2 \tag{9.11}$$

and

$$p_1 = \frac{D_L^2}{N}. \tag{9.12}$$

which is the same shape as for the xy-resolution. In z direction N consists of the number of electrons N_{eff} produced along one pad row but it contains mainly effects from the shaping time of the readout electronics. The electronics broadens the signal artificially. The measured charge is more broadened by the shaping of the electronics than by the diffusion. The values obtained by the fit are summarized in Table 9.6.

	p_0 [mm^2]	p_1 [µm]	σ_0 [mm]	N
no correction	0.1301 ± 0.0003	0.069 ± 0.001	0.3608 ± 0.0004	78.795 ± 3.072
with correction	0.0899 ± 0.0003	0.105 ± 0.001	0.2998 ± 0.0005	51.779 ± 1.938

Table 9.6: z-resolution fit results.

9.4.6 Resolution Studies with Magnetic Field

A number of measurement runs were taken in a magnetic field of 1T. Since large parts of the module were destroyed due to trips, only one of the four sectors is used for the analysis. The beam is positioned over the remaining sensitive region of 14 rows. Large field distortions are expected, but nevertheless the resolution is investigated. In the following the z-scan performed with magnetic field is investigated. The methods used are the same as in the analysis of the 0T data.
First two runs, one with and one without magnetic field are compared. Runs in the center of the TPC are chosen: 17699 (without magnetic field) and 17767 (with magnetic field). The comparison is done after each reconstruction step. Only few significant differences are observed which can be explained by the presence of the magnetic field. Figure 9.20 shows the average number of pulses in a hit per event. Without magnetic field there are more pulses in a hit, which is due to the higher

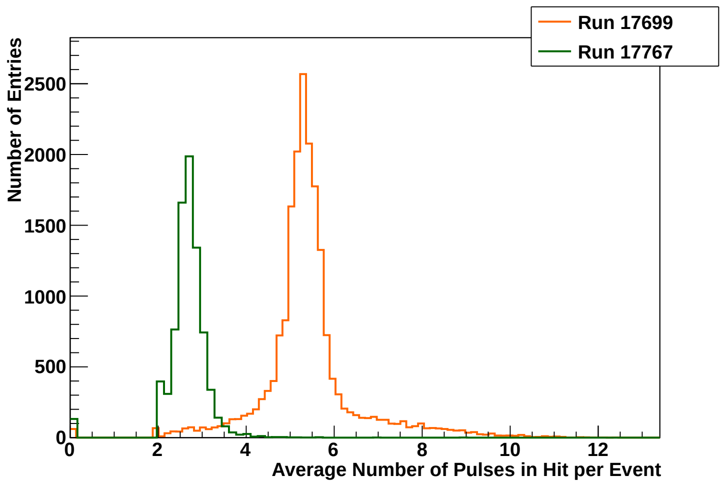


Figure 9.20: The average number of pulses in a hit per event for a run taken with (Run 17767) and one taken without (Run 17699) magnetic field. The two runs are taken at about the same drift length.

transverse diffusion. The charge is spread over more pads in the absence of a magnetic field.
First of all the drift velocity is measured. For this measurement the table position and the mean hit time for each run is used. There is a linear dependence between these two numbers, as can be seen in Figure 9.21. The slope of the straight line gives the drift velocity. The drift velocity is $v_{\text{drift}} = 75.46 \pm 0.02$ mm/µs. This value is in agreement with the one measured previously for the 0T data.
There are no runs taken close to the anode and close to the cathode, so the determination of the anode and cathode position cannot be done and thus no time shift

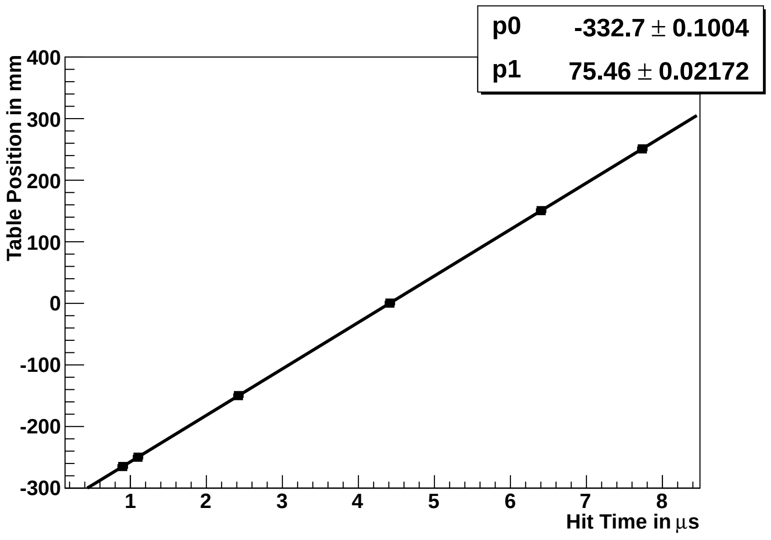


Figure 9.21: Drift velocity measurement with 1T data. On the x-axis the mean time of the hits in each run is plotted, the y-axis shows the table position.

can be calculated to correct the data. However, the electronics parameters and the setup was not changed. Thus the same corrections on the (raw) data time as for the 0T data are applied. Apart from the drift velocity the same parameters for the reconstruction are used as for the 0T data.
A distortion analysis is done as described before. The plots in Figure 9.22 show the rowwise distortions for two runs, one close to the anode, one close to the cathode. The distortions are corrected by shifting the hits accordingly and refitting the track. Since the tracks are not very long a straight line is assumed in the track reconstruction. Rows 0 to 13 are excluded because the module was broken in those regions. Furthermore rows 25 to 27 are excluded because they show a low hit efficiency. The

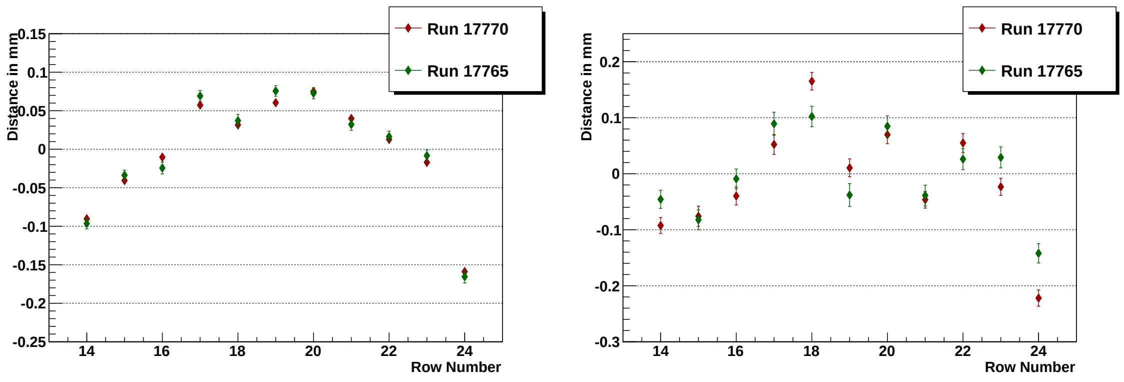


Figure 9.22: Distortions (distance between hit and track) in y (left) and z (right) for 1T data for one run close to the anode (Run 17770) and one run close to the cathode (Run 17765). Rows that were excluded or did not work are not shown.

resolution is calculated according to equation 9.6 for each run (runs are taken at different drift length). The result is shown in Figure 9.23. The resolution in xy is, as

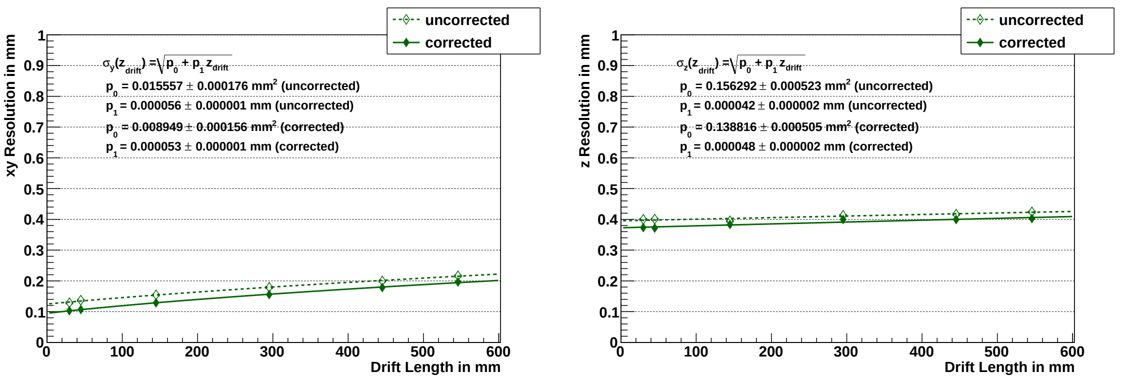


Figure 9.23: Resolution in xy (left) and z (right) for different drift length with and without corrections for distortions.

expected, much better with magnetic field than without. The transverse diffusion is less due to the magnetic field. The resolution in z looks almost the same with and without magnetic field. The reason for this is that the longitudinal diffusion does not depend on the magnetic field if it has the same direction as the electric field.
From the fit to the xy-resolution N_{eff} and σ_0 are calculated where $D_T = 90.576 \pm 0.998 \frac{\mu\text{m}}{\sqrt{\text{cm}}}$ [103] from a `MAGBOLTZ` simulation is used. The fit results are summarized in Tabl e9.7. The two values for N_{eff} are in good agreement. However, as already

	p_0 [mm^2]	p_1 [µm]	σ_0 [mm]	N_{eff}
no correction	0.0156 ± 0.0002	0.056 ± 0.001	0.1247 ± 0.0007	14.650 ± 0.416
with correction	0.0089 ± 0.0002	0.053 ± 0.001	0.0946 ± 0.0008	15.479 ± 0.449

Table 9.7: xy-resolution fit results of 1T data.

mentioned in Section 9.4.5.3 the fit is not very reliable because the minimum of the χ^2 is rather broad. This is even more pronounced in this case because only few data points are available.
The longitudinal diffusion in a 1T magnetic field is found to be $D_L = 233.16 \pm 4.22 \frac{\mu\text{m}}{\sqrt{\text{cm}}}$ [103] (`MAGBOLTZ` simulation). In z the same function is fitted as before, however the shaping of the electronics does have a big effect and thus N_{eff} cannot be calculated. The values obtained by the fit are summarized in Table 9.8.

	p_0 [mm^2]	p_1 [µm]	σ_0 [mm]	N
no correction	0.1563 ± 0.0005	0.042 ± 0.002	0.3953 ± 0.0006	129.437 ± 7.742
with correction	0.1388 ± 0.0005	0.048 ± 0.002	0.3726 ± 0.0006	113.257 ± 6.251

Table 9.8: z-resolution fit results of 1T data.

9.4.7 Drift Velocity for different Drift Fields

During the testbeam campaign a number of runs were taken with different drift fields inside the TPC. Those data were taken in the presence of a magnetic field of

1T. For each drift field two runs are available with different drift length. From this the drift velocity for different drift fields is calculated using the mean hit time and the table position of the two runs

$$v_{\text{drift}} = \frac{z_{\text{table},1} - z_{\text{table},2}}{t_{\text{hit},1} - t_{\text{hit},2}}. \tag{9.13}$$

This equation is nothing else but the slope of a straight line between the two measurements. The uncertainties are calculated by error propagation using the width of the hit time distribution as uncertainty on the hit time. The table position is regarded as perfectly measured. The result is shown in Figure 9.24. The measure-

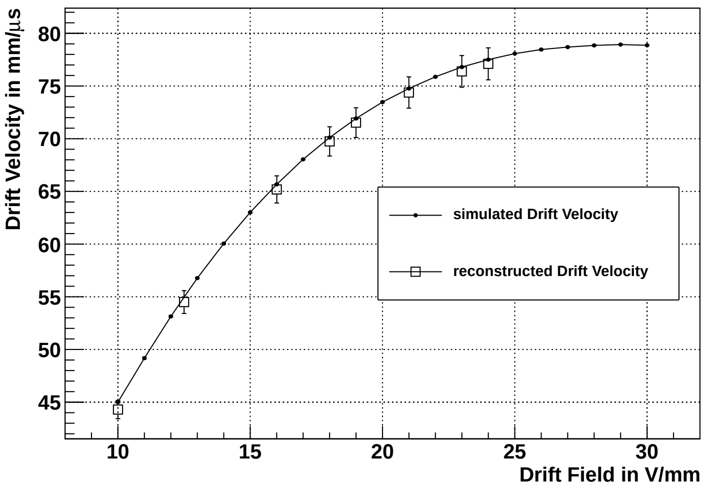


Figure 9.24: Drift velocities calculated from testbeam data for different drift fields compared with `MAGBOLTZ` simulation by Klaus Zenker [106].

ments are in good agreement with the `MAGBOLTZ` simulation which is shown as a black line.

9.5 Three Module Data

In Figure 9.25 an event display is shown of a typical event taken in the testbeam campaign in September 2012. The data were taken with a new version of the readout module described in Section 9.1.4 [107]. Three modules were mounted in the Large Prototype and a magnetic field of 1T was used.
The track finding was done with `Pathfinder` using a helix as track model. Most of the reconstructed hits (orange crosses) are assigned to the track (black line) and the track matches the measurements. In the event several measurements are missing. A large part of one module did not work during data taking. Nevertheless hits measured on all three modules are assigned to the track. This proves that the Hough transformation implemented in `Pathfinder` can find tracks in real data where many measurements are missing.

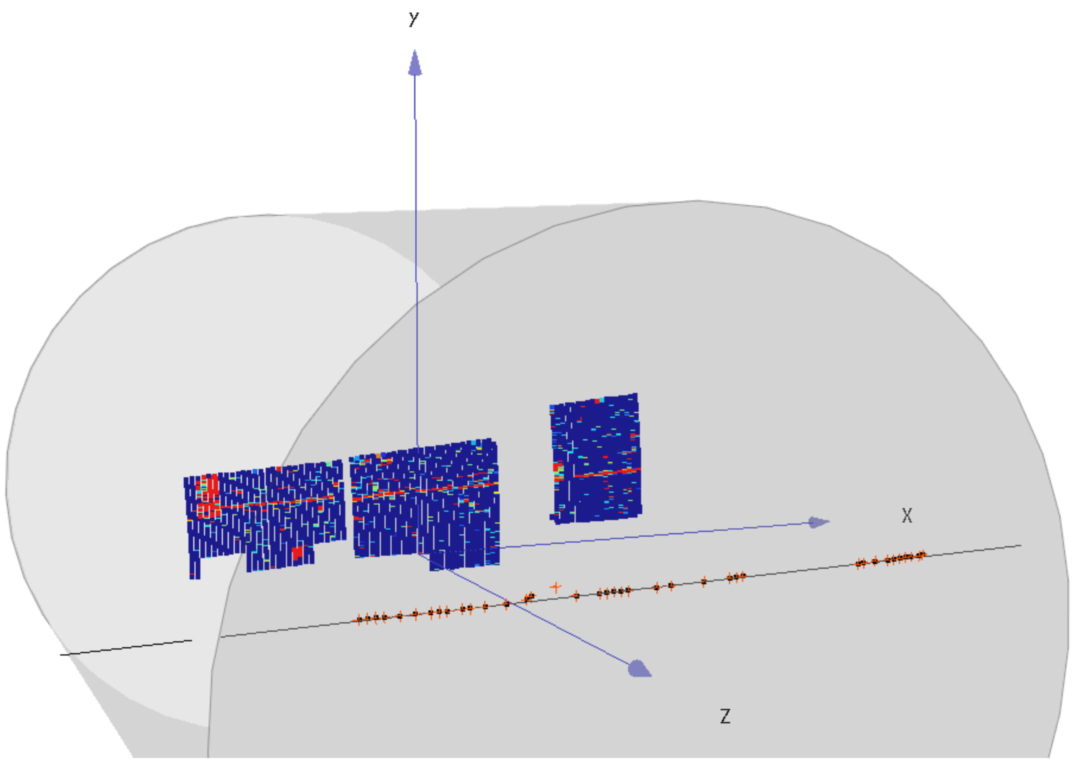


Figure 9.25: Event display of an event taken with the Large Prototype and three readout modules in September 2012 with a magnetic field of 1T. The track found with `Pathfinder` is shown as a black line and the hits assigned to the track are represented by black dots. All hits having been reconstructed in the event are shown as orange crosses. The areas with colors ranging from blue to red represent the amount of charge that was collected on each pad (red: much charge, blue: little charge).

9.6 Summary

The most important results of the measurements presented in this chapter are summarized in Table 9.9. The values obtained for the drift velocity are similar with and without magnetic field, which is expected because the drift velocity depends on the electric field only. Same is true for the single point resolution in z direction. The

	0T	**1T**
v_{drift}	75.24 ± 0.16 mm/µs	75.46 ± 0.02 mm/µs
Resolution over full drift length of the Large Prototype (567.6 mm):		
σ_{xy}	≈ 460 µm	≈ 200 µm
σ_z	≈ 380 µm	≈ 410 µm
Resolution over full drift length of the ILD TPC (2350 mm):		
σ_{xy}	≈ 950 µm	≈ 360 µm
σ_z	≈ 580 µm	≈ 500 µm

Table 9.9: The most important results of the testbeam data analysis.

single point resolution in xy is significantly smaller in the presence of a magnetic field, which is also expected because the magnetic field forces the electrons having been produced during ionization onto curved trajectories. Thus the transverse diffusion is less and the single point resolution is higher.
The maximum drift length in the Large Prototype (567.6 mm) is smaller that the one in the ILD TPC (2350 mm, see Table 4.2). The functions fitted to the resolution measurements in Sections 9.4.5.3 to 9.4.6 are extrapolated to the ILD TPC drift length. This is shown in Figures 9.26. The values obtained for σ_z do meet

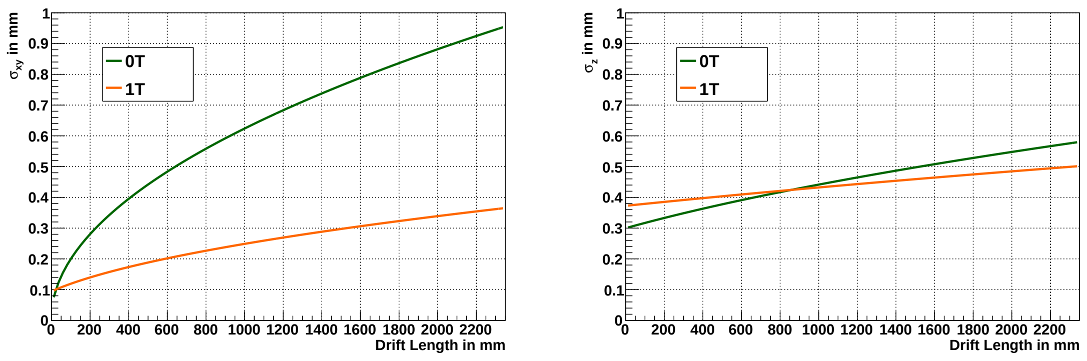


Figure 9.26: Extrapolation of point resolution to ILD TPC drift length for xy (left) and z (right).

the requirements for the ILD TPC ($<$ 1.4 mm over full drift length, 0.4 mm at zero drift length, see Table 4.2). The point resolution in xy strongly depends on the magnetic field. For the ILD a magnetic field of 3.5T is foreseen, thus the single point resolution is expected to be smaller than the values obtained in the analysis presented here. The data for 1T magnetic field were taken with a module which was not fully operational and only very few data are available. Furthermore large field distortions appeared at the borders of the module. Nevertheless, the results look promising so that, with further improvements of the module and with a higher magnetic field, it will be possible to achieve the required single point resolution for the ILD TPC.

Chapter 10

Summary and Outlook

In this thesis the development of a software package for track finding based on a Hough transformation was presented. The package is called `Pathfinder` and was mainly developed for the reconstruction of tracks in testbeam data. It was used on different kinds of data: a simplified simulation, a Monte Carlo simulation based on `geant4` and testbeam data.
First systematic tests were performed with a simplified simulation for straight lines and helices to understand the limitations of the algorithm. These turned out to be that tracks along a pad row and tracks parallel to the drift direction cannot be found. In the first case only one hit would be reconstructed along the whole pad row. A different algorithm is needed to deal with such a situation. In the latter case the track would end up on one single pad and thus also only one single hit would be reconstructed. It is, however, expected that such tracks usually do not come from physics events. Apart from these limitations also the computing time sets limits to which events can be reconstructed in reasonable time. This mainly depends on how many hits there are in the event and on the binning chosen for the Hough space. All in all it turned out that straight lines can be found better, be it in multi track events, in noisy events or with smearing applied. In general the more tracks and noise hits are present and the more smearing was applied to the hits the lower the track finding efficiency is, which is expected because the events get more and more complex and therefore hits cannot always be assigned to the correct track. For straight lines in all tested cases the track finding efficiency is larger than 99 %. For helices in most cases it is larger than 97 %. The only exception is if smearing is larger than 0.5 mm. However, this is a rather large value. In the ILD TPC a single point resolution in xy of no more than 0.1 mm is expected. Since in most of these studies a correctly found track was defined via the difference between the simulated and reconstructed track parameters[1], the results can be improved by applying a track fit to determine the parameters precisely.
In order to answer the question if `Pathfinder` can be applied on ILD-like tracks a simulation of events (single muon events, two muon events and tau decays) in the ILD detector, based on `geant4` was done. `Pathfinder` was run on these data and the results were compared to those of a different track finding algorithm called `Clupatra` which is based on a local pattern recognition algorithm (cluster finding and Kalman

[1]`Pathfinder` gives a rough estimate on the track parameters only.

Filter) and was developed specifically to find tracks in the ILD detector. Both algorithms show similar track finding efficiencies for tracks with transverse momenta between 1 and 50 GeV and for $\cos\theta$ between 0 and 1 of almost 100 %. However, there are some differences, which are understood and options for solutions/improvments for those cases in which `Pathfinder` performs worse exist. The most significant differences are:

- In 3-prong tau decays it turned out that the split rate for `Pathfinder` is larger than for `Clupatra` which leads to a lower track finding efficiency for `Pathfinder`. However, `Clupatra` uses a track matching algorithm. Applying such an algorithm to the tracks found by `Pathfinder` is expected to improve the result.

- The reconstruction of the transverse momentum is worse with `Pathfinder` than with `Clupatra`. However, the Kalman Filter used by `Clupatra` delivers fitted tracks while the track parameters delivered by `Pathfinder` are only a rough estimate. Applying a proper track fit to the tracks found by `Pathfinder` is expected to improve the transverse momentum reconstruction.

- `Pathfinder` can deal with detector inefficiencies (missing measurements) much better than `Clupatra`. For `Clupatra` the track finding efficiency breaks down at about 15 % of missing hits while `Pathfinder` can find tracks realiably even if 35 % or more of the measurements are missing.

- `Pathfinder` is much slower than `Clupatra` which is a clear drawback. For this reason `Pathfinder` was only tested on rather simple event with low track multiplicities. For testbeam data this is sufficient. There is a number of options how `Pathfinder` can be improved in this respect. They will be discussed further below in detail.

To summarize, in the current implementation it is not feasible to use `Pathfinder` in complex ILD-like events because the algorithm is too slow to run it in a reasonable amount of time.
As mentioned before, `Pathfinder` was mainly written to find tracks in testbeam data. A measurement of the single point resolution of such data was presented in the last chapter of this thesis. Tracks are needed for this analysis because the single point resolution is a measure for how much the hits deviate from the track. Before reconstructing tracks in the testbeam data a number of other analysis steps are required: The determination of the end points of the TPC, the measurement of the drift velocity and the analysis of distortions. For all these steps software was written which can be (and is) used for other testbeam data. With the tracks found in the data an analysis of the single point resolution was done with a magnetic field of 1T and without magnetic field. Although a large part (three fourth) of the readout module was broken when the data with magnetic field were taken it could be shown, that the single point resolution in the readout plane (transverse resolution) is smaller than in the data without magnetic field while the longitudinal resolution does not change much, which is expected since the magnetic field is parallel to the electric field and thus only influences the electrons perpendicular to the drift

direction. Extrapolating the measurements to the drift length of the ILD-TPC gives a single point resolution about 0.95 mm (0.36 mm) with a magnetic field of 0T (1T) in the readout plane and about 0.58 mm (0.50 mm) with a magnetic field of 0T (1T) in z. The point resolution in the readout plane will further decrease with a higher magnetic field (for the ILD a magnetic field of 3.5T is foreseen), thus it is possible to reach the requirement for the ILD-TPC (point resolution < 0.1 mm over full drift length). In z the requirements are already met (single point resolution < 1.4 mm). All in all the results look promising, especially if one keeps in mind that no proper track fit was applied to the track (the estimate of the track parameters delivered by `Pathfinder` were used) and for 1T data not the full module was operational.
`Pathfinder` proved to work on simulated data in the ILD detector as well as on testbeam data (even if large parts of a readout module are broken). However, there is still room for improvements. A lot of effort was put into decreasing the computing time the Hough transformation needs and the result is sufficient. However, further improvements are possible. The part of the algorithm which takes longest is the calculation of the Hough space, thus this is the place were the computing time can be decreased most easily. Several options exist how this can be achieved:

- In the current implementation a map is used to hold the Hough space. The entries of a map are sorted when a new entry is added. Thus using maps is comparably slow. A possible solution is to use a container to hold the Hough space which is not sorted, for example a vector.
- Instead of calculating the full Hough space it is possible to calculate the intersection of two functions analytically and fill only these into the Hough space. No loop over the bins is needed for which the functions need to be calculated and the container holding the Hough space stays much smaller, but on the other hand one additional loop over the hits is required.
- The current implementation of `Pathfinder` follows an iterative approach. In each iteration one track is found. This iteration can be avoided by finding all intersections in Hough space in one go, thus the Hough space needs to be calculated only once, and not once for each track.
- The implementation of an adaptive Hough transformation is another option to improve the computing time. In this method the Hough space is calculated with a coarse binning, the region which most likely contains the intersection is identified which is recalculated with a finer binning.

The gain in speed is expected to be largest for the adaptive Hough transformation which can be up to a factor of 50 (rough estimation). But the algorithm is very complex and the interesting regions must be identified reliably, which is expected to be the most difficult part. However, in combination with the second point mentioned above this could be easier. It is also not clear how well the algorithm performs for high track multiplicities. The initial binning needs to be chosen fine enough so that it represents the overall structure of the Hough space. If that binning has to be chosen very fine already, one cannot gain anything with an adaptive Hough transformation [86].

Finding all intersections in Hough space in one go is expected to be fast because the Hough space needs to be calculated only once. However, one has to ensure that all intersections and only those that are real intersections are found. With this method the fake rate (which is almost zero in the current implementation) could increase dramatically if too many fake intersections are found or the efficiency could drop significantly if to few real intersections are found.
Changing the used container can be done relatively easily but it is difficult to estimate how much this will improve the computing time. Since the computing time increases with number of hits (depending on the track type to be found) quadratically or with the third power it is likely that one cannot gain much by this.
The method of filling the intersections into the Hough space can be implemented in the current version of `Pathfinder` with comparably small effort and a factor of about 10 could be gained (very rough estimate). It is expected that no new limitations are introduced by this method. Instead of calculating all intersections analytically one could regard only those for hits that are likely to be on the same track (i.e. which are close to each other). It needs to be shown if and by how much the computing time can be reduced in multiple track events because the number of hit combinations increases with number of tracks per event.

Appendix

Appendix A

Derivation of Equations used in `Pathfinder`

A.1 Slope and Offset of a Straight Line and d_0, ϕ_0

Let m be the slope and b the offset of a straight line. Then a straight line is described by

$$f(x) = m \cdot x + b. \tag{A.1}$$

Using the LCIO track parameter d_0 and ϕ_0 [60] the function reads

$$f(x) = \tan(\phi_0) \cdot x + \frac{d_0}{\cos(\phi_0)}. \tag{A.2}$$

A.2 Calculating the Point of Closest Approach

The point of closest approach (pca) is the point on a track which is closest to the point of reference (here: the origin of the coordinate system). A sketch is shown in Figure A.1) and the equations to calculate the pca are calculated in the following. The equations

$$x_{\text{pca}} = d_0 \cdot \sin(\phi_0^{'}) \tag{A.3}$$
$$y_{\text{pca}} = d_0 \cdot \cos(\phi_0^{'}) \tag{A.4}$$

are valid. The task is to convert $\phi_0^{'}$ to the LCIO track parameter ϕ_0 [60]. In this case this is done via

$$\phi_0^{'} = \pi - \phi_0 \tag{A.5}$$

Inserting this into the equations for the pca gives

$$\begin{aligned}
x_{\text{pca}} &= d_0 \cdot \sin(\pi - \phi_0) \\
&= d_0 \cdot (\sin(\pi)\cos(\phi_0) - \cos(\pi)\sin(\phi_0)) \\
&= d_0 \cdot \sin(\phi_0) \qquad (A.6) \\
y_{\text{pca}} &= d_0 \cdot \cos(\pi - \phi_0) \\
&= d_0 \cdot (\cos(\pi)\cos(\phi_0) + \sin(\pi)\sin(\phi_0)) \\
&= -d_0 \cdot \cos(\phi_0). \qquad (A.7)
\end{aligned}$$

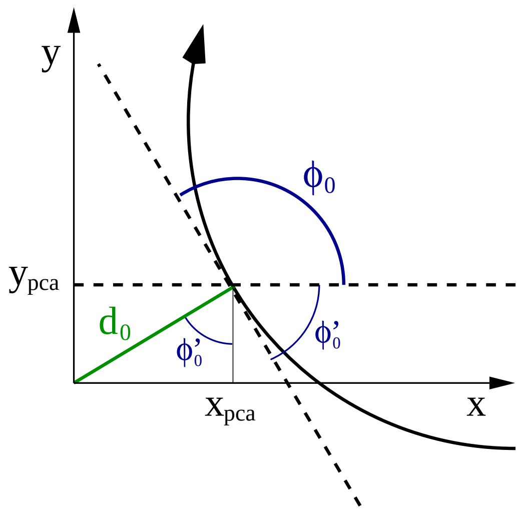


Figure A.1: Calculating point of closest approach.

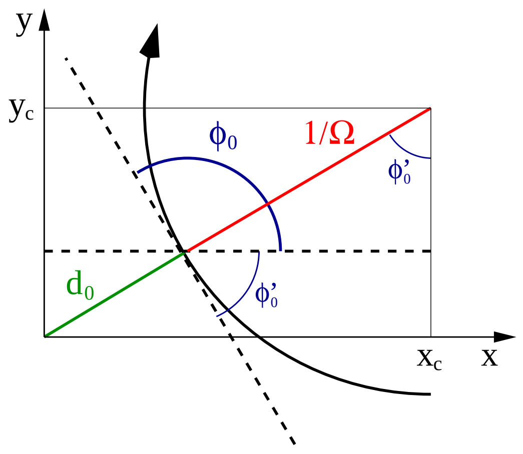


Figure A.2: Calculating center of circle.

Since in the shown case $d_0 < 0$ these expressions are equivalent to

$$\boxed{\begin{aligned} x_{\text{pca}} &= -d_0 \cdot \sin(\phi_0) \quad (A.8)\\ y_{\text{pca}} &= d_0 \cdot \cos(\phi_0). \quad (A.9)\end{aligned}}$$

A.3 Calculating the Center of a Circle

Calculating the center of a circle using LCIO track parameters [60] is similar to calculating the point of closest approach. In Figure A.2 the parametrization is shown. The center of the circle is calculated by

$$x_c = \left(\frac{1}{|\Omega|} + |d_0|\right) \cdot \sin(\phi_0^{'}) \tag{A.10}$$

$$y_c = \left(\frac{1}{|\Omega|} + |d_0|\right) \cdot \cos(\phi_0^{'}). \tag{A.11}$$

The conversion of $\phi_0^{'}$ to the LCIO track parameter ϕ_0 [60] is given by

$$\phi_0^{'} = \pi - \phi_0. \tag{A.12}$$

Thus for the center of the circle one gets

$$\begin{aligned} x_c &= \left(\frac{1}{|\Omega|} + |d_0|\right) \cdot \sin(\pi - \phi_0) \\ &= \left(\frac{1}{|\Omega|} + |d_0|\right) \cdot (\sin(\pi)\cos(\phi_0) - \cos(\pi)\sin(\phi_0)) \\ &= \left(\frac{1}{|\Omega|} + |d_0|\right) \cdot \sin(\phi_0) \qquad \text{(A.13)} \\ y_c &= \left(\frac{1}{|\Omega|} + |d_0|\right) \cdot \cos(\pi - \phi_0) \\ &= \left(\frac{1}{|\Omega|} + |d_0|\right) \cdot (\cos(\pi)\cos(\phi_0) + \sin(\pi)\sin(\phi_0)) \\ &= -\left(\frac{1}{|\Omega|} + |d_0|\right) \cdot \cos(\phi_0). \qquad \text{(A.14)} \end{aligned}$$

Since in this case $d_0 < 0$ and $\Omega > 0$ one can write

$$\boxed{\begin{aligned} x_{\text{center}} &= \left(\frac{1}{\Omega} - d_0\right) \cdot \sin(\phi_0) \qquad \text{(A.15)} \\ y_{\text{center}} &= -\left(\frac{1}{\Omega} - d_0\right) \cdot \cos(\phi_0). \qquad \text{(A.16)} \end{aligned}}$$

A.4 Calculating the Arc Length

A.4.1 Straight Lines

For straight lines the arc length s is the shortest distance between the hit and the point of closest approach. Let $(x_{\text{hit}}, y_{\text{hit}})$ be the position of the hit and $(x_{\text{pca}}, y_{\text{pca}})$ the position of the point of closest approach. Then the arc length is given by

$$\boxed{s = \pm\sqrt{(x_{\text{hit}} - x_{\text{pca}})^2 + (y_{\text{hit}} - y_{\text{pca}})^2}.} \tag{A.17}$$

s is positive if the particle crossed the point of closest approach first and it is negative if the particle crossed the hit position first. x_{pca} and y_{pca} are calculated according to Equations A.8 and A.9.

A.4.2 Circles

To calculate the arc length s for circles the curvature of the circle must be taken into account (see Figure A.3). In general the arc length is calculated via

$$s = \frac{1}{|\Omega|} \cdot \omega, \tag{A.18}$$

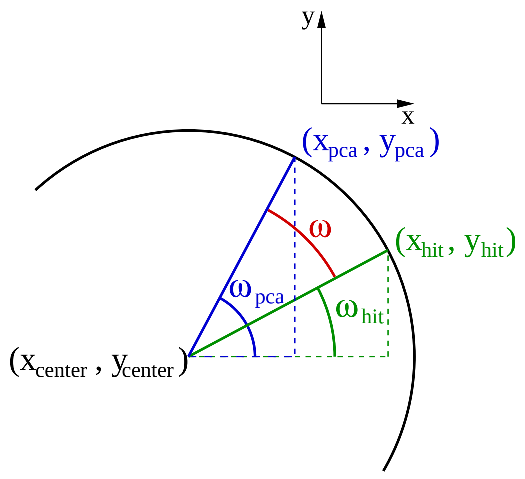


Figure A.3: Calculating arc length s for circles.

where ω is the difference between ω_{pca} and ω_{hit}. Thus one can write

$$s = \frac{1}{|\Omega|} \cdot (\omega_{\text{pca}} - \omega_{\text{hit}}) , \tag{A.19}$$

where ω_{pca} is given by

$$\omega_{\text{pca}} = \arctan\left(\frac{y_{\text{pca}} - y_{\text{center}}}{x_{\text{pca}} - x_{\text{center}}}\right) \tag{A.20}$$

and analogously

$$\omega_{\text{hit}} = \arctan\left(\frac{y_{\text{hit}} - y_{\text{center}}}{x_{\text{hit}} - x_{\text{center}}}\right) . \tag{A.21}$$

The arc length s for circles can thus be calculated by

$$\boxed{s = \frac{1}{|\Omega|} \cdot \left(\arctan\left(\frac{y_{\text{pca}} - y_{\text{center}}}{x_{\text{pca}} - x_{\text{center}}}\right) - \arctan\left(\frac{y_{\text{hit}} - y_{\text{center}}}{x_{\text{hit}} - x_{\text{center}}}\right)\right) .} \tag{A.22}$$

A.5 Calculating the Distance between Hit and Track

A.5.1 Straight Lines

How the distance between a hit and a straight line d can be calculated is shown in Figure A.5. The distance is given by

$$d = (y_{\text{hit}} - y_{\text{sl}}) \cdot \cos(\phi_0) . \tag{A.23}$$

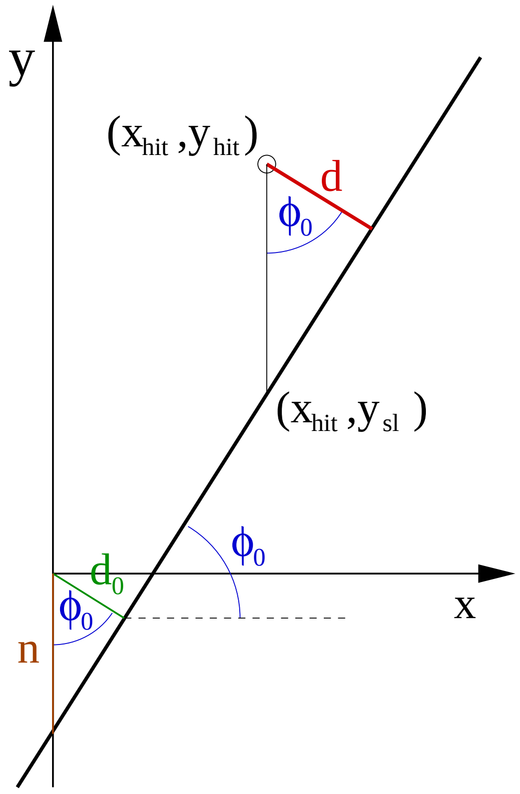


Figure A.4: Calculating the distance between a hit and a straight line.

y_{sl} is the y position on the track at the x position of the hit:

$$\begin{aligned} y_{\mathrm{sl}} &= y(x_{\mathrm{hit}}) && \text{(A.24)} \\ &= m \cdot x_{\mathrm{hit}} + n && \text{(A.25)} \\ &= \tan(\phi_0) \cdot x_{\mathrm{hit}} + \frac{d_0}{\cos(\phi_0)} && \text{(A.26)} \\ &= \frac{\sin(\phi_0)}{\cos(\phi_0)} \cdot x_{\mathrm{hit}} + \frac{d_0}{\cos(\phi_0)} && \text{(A.27)} \end{aligned}$$

where m and n are the slope and the offset of the straight line, respectively. Inserting Equation A.27 in Equation A.23 gives

$$\begin{aligned} d &= \left(y_{\mathrm{hit}} - \frac{\sin(\phi_0)}{\cos(\phi_0)} \cdot x_{\mathrm{hit}} - \frac{d_0}{\cos(\phi_0)} \right) \cdot \cos(\phi_0) && \text{(A.28)} \\ &= y_{\mathrm{hit}} \cdot \cos(\phi_0) - x_{\mathrm{hit}} \cdot \sin(\phi_0) - d_0 && \text{(A.29)} \\ &= -1 \cdot (x_{\mathrm{hit}} \cdot \sin(\phi_0) - y_{\mathrm{hit}} \cdot \cos(\phi_0) + d_0)\,. && \text{(A.30)} \end{aligned}$$

Since only the value of d is of interest, not the sign one can thus calculate the distance d between a hit and a straight line with

$$\boxed{d = |x_{\mathrm{hit}} \cdot \sin(\phi_0) - y_{\mathrm{hit}} \cdot \cos(\phi_0) + d_0|} \qquad \text{(A.31)}$$

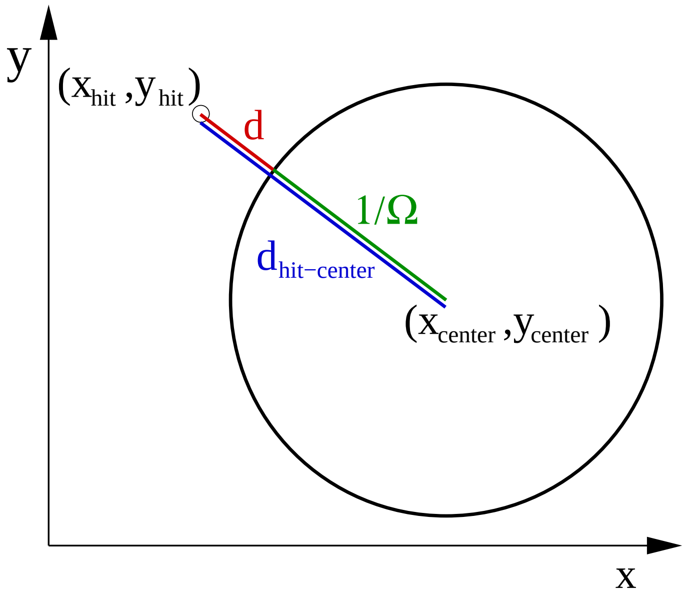


Figure A.5: Calculating the distance between a hit and a circle.

A.5.2 Circles

The shortest distance between a hit and a circle d is calculated by first calculating the distance between the hit and the center of the circle $d_{\text{hit-center}}$ and then subtracting the radius $R = \frac{1}{|\Omega|}$:

$$d \quad = \quad \left| d_{\text{hit-center}} - \frac{1}{|\Omega|} \right| . \tag{A.32}$$

The distance between the hit and the center of the circle is given by

$$d_{\text{hit-center}} \quad = \quad \pm\sqrt{(x_{\text{hit}} - x_{\text{center}})^2 + (y_{\text{hit}} - y_{\text{center}})^2}. \tag{A.33}$$

The center of the circle is calculated according to Equations A.15 and A.16. This equations has two solutions. Since the minimum distance between hit and circle is wanted, the smaller solution for the distance d is chosen. The distance is thus calculated by

$$\boxed{d = \min\left(\left|\pm\sqrt{(x_{\text{hit}} - x_{\text{center}})^2 + (y_{\text{hit}} - y_{\text{center}})^2} - \frac{1}{|\Omega|}\right|\right) .} \tag{A.34}$$

Appendix B

Usage of Pathfinder: Example Code

B.1 Track Finding

B.1.1 Steering Parameters

```
//FinderParameter* myFinderParameter= new FinderParameter(true, false) //straight line
FinderParameter* myFinderParameter= new FinderParameter(false, true) //helix
myFinderParameter -> setVertex(0.,0.);
myFinderParameter -> setFindCurler(false);
myFinderParameter -> setMaxXYDistance(5.);
myFinderParameter -> setMaxSZDistance(5.);
myFinderParameter -> setMaxXYDistanceFit(3.);
myFinderParameter -> setMaxSZDistanceFit(3.);
myFinderParameter -> setMinimumHitNumber(5);
myFinderParameter -> setNumberXYThetaBins(1000);
myFinderParameter -> setNumberXYDzeroBins(1000);
myFinderParameter -> setNumberXYOmegaBins(1000);
myFinderParameter -> setNumberSZThetaBins(1000);
myFinderParameter -> setNumberSZDzeroBins(1000);
myFinderParameter -> setMaxDxy(10.);//for helix
//myFinderParameter -> setMaxDxy(3200.);//for straight line
myFinderParameter -> setMaxDsz(3200.);
myFinderParameter -> setSearchNeighborhood(false);
myFinderParameter -> setSaveRootFile(false);
```

B.1.2 Creating Hits

```
basicHit myHit(xpos, ypos, zpos);
myBasicHitVector.push_back(myHit);
```

B.1.3 Perform Track Finding

```
HoughTrafoTrackFinder myTrackFinder;
myTrackFinder.setFinderParameter(*myFinderParameter);
myTrackFinder.setInitialHits(myBasicHitVector);
bool track_found = myTrackFinder.find();
```

B.1.4 Getting Result

```
vector<TrackFinderTrack> foundTracks = myTrackFinder.getTracks();
//get track parameters of first track in vector
TrackParameterFull recoTrackParam = foundTracks[0].getTrackParameter();

double d0 = recoTrackParam.getDZero();
double phi = recoTrackParam.getPhi();
double omega = recoTrackParam.getOmega();
double z0 = recoTrackParam.getZZero();
double tanl = recoTrackParam.getTanLambda();

double d0Error = recoTrackParam.getDZeroError();
double phiError = recoTrackParam.getPhiError();
double omegaError = recoTrackParam.getOmegaError();
double z0Error = recoTrackParam.getZZeroError();
double tanlError = recoTrackParam.getTanLambdaError();

//get hits on track
vector<basicHit> hitsOnTrack = foundTracks[0].getHitsOnTrack();
```

B.2 Track Generation

B.2.1 Define Event Type

```
  unsigned int nhitsontrack = 50;
  unsigned int nevents = 10;
  unsigned int ntracks = 1;
  unsigned int nnoise = 0;
  double smearing = 0.;
  TGEventType EventType(nhitsontrack,
                        ntracks,
                        nnoise,
                        smearing,
                        nevents);
```

B.2.2 Define Detector Type

```
  double padplanexmin = -1000.;
  double padplanexmax = 1000.;
  double padplaneymin = -1400.;
  double padplaneymax = 1400.;
  double padplanezmin = -1000.;
  double padplanezmax = 1000.;
  double padsizey = 7.;
  TGDetectorType DetectorType(padplanexmin, padplanexmax,
                              padplaneymin, padplaneymax,
                              padplanezmin, padplanezmax,
                              padsizey);
```

B.2.3 Set Parameter Limits

```
  double phimin = -3.14159;
  double phimax = 3.14159;
  double d0min = 0.;
  double d0max = 1000.;
  double rmin = -1000.;
  double rmax = 1000.;
  double tanlmin = -1.;
  double tanlmax = 1.;
  double z0min = 0.;
  double z0max = 1000.;
  TGTrackParameterLimits TrackParameterLimits(phimin, phimax,
                                              d0min, d0max,
                                              rmin, rmax,
                                              tanlmin, tanlmax,
                                              z0min, z0max);
```

B.2.4 Perform Track Generation

```
unsigned int track_type = 2;//0 = straight line, 1 = helix segment, 2 = curler

TrackGenerator newTrackGen(track_type, TrackParameterLimits, EventType,
    Detectortrype);

for(unsigned int event = 0; event<nevents; event++)
{
 newTrackgen.generateTracks(event);
 vector<TrackFinderTrack*> simtracks = newTrackGen.getGeneratorTracks();
 vector<basicHit> allhits = newTrackGen.getHits();
 vector<basicHit> noisehits = newTrackGen.getNoiseHits();
 for(unsigned int i = 0; i<simtracks.size(); i++)
 {
  TrackParameterFull currentTrackParameters = simtracks[i]->getTrackParameter();
  vector<basicHit> currentTrackHits = simtracks[i]->getHitsOnTrack();
  double d0 = currentTrackParameters.getDZero();
  double phi = currentTrackParameters.getPhi();
  double omega = currentTrackParameters.getOmega();
  double tanl = currentTrackParameters.getTanLambda();
  double z0 = currentTrackParameters.getZZero();
  for(unsigned int j =0; j<currentTrackHits.size(); j++)
  {
   double pos[3];
   pos[0] = currentTrackHits[j].getX();
   pos[1] = currentTrackHits[j].getY();
   pos[2] = currentTrackHits[j].getZ();
  }
 }
}
```

Appendix C

UML Diagrams

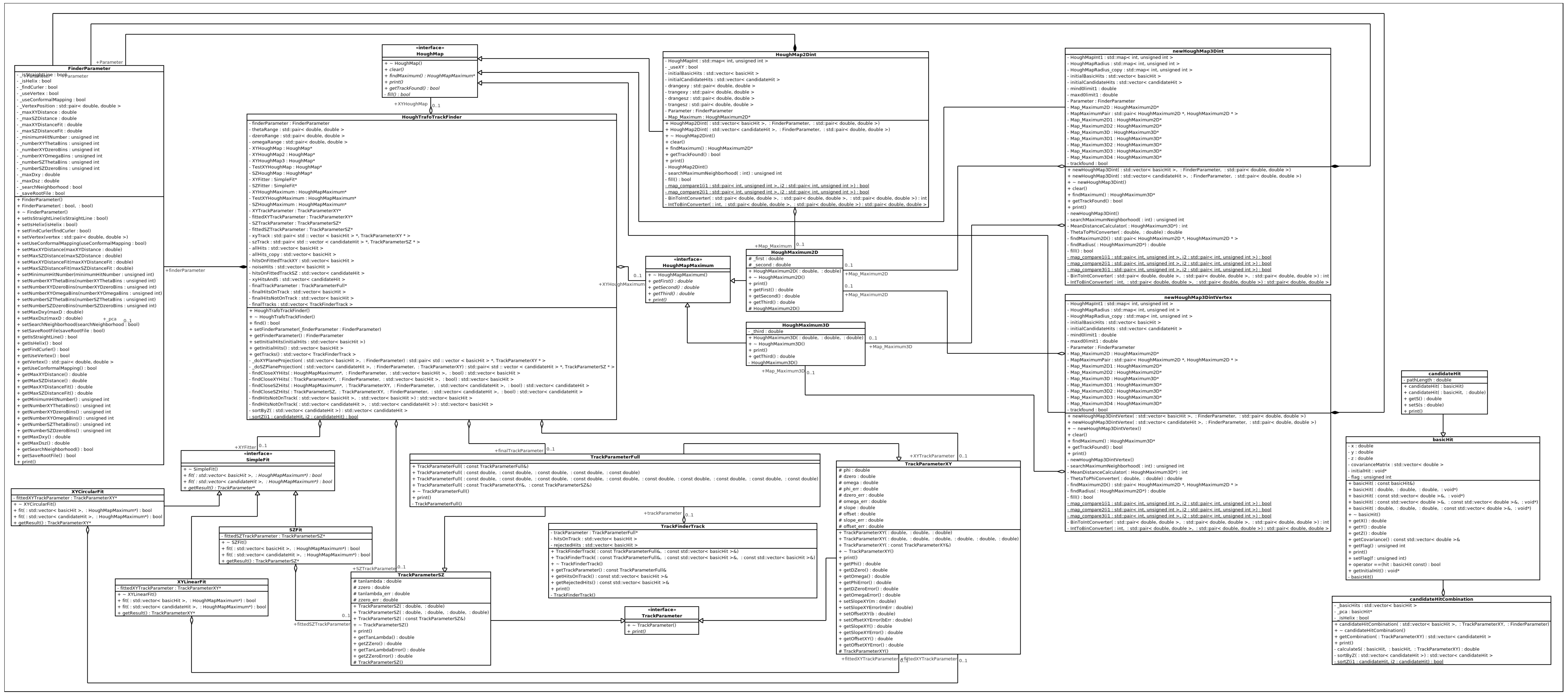
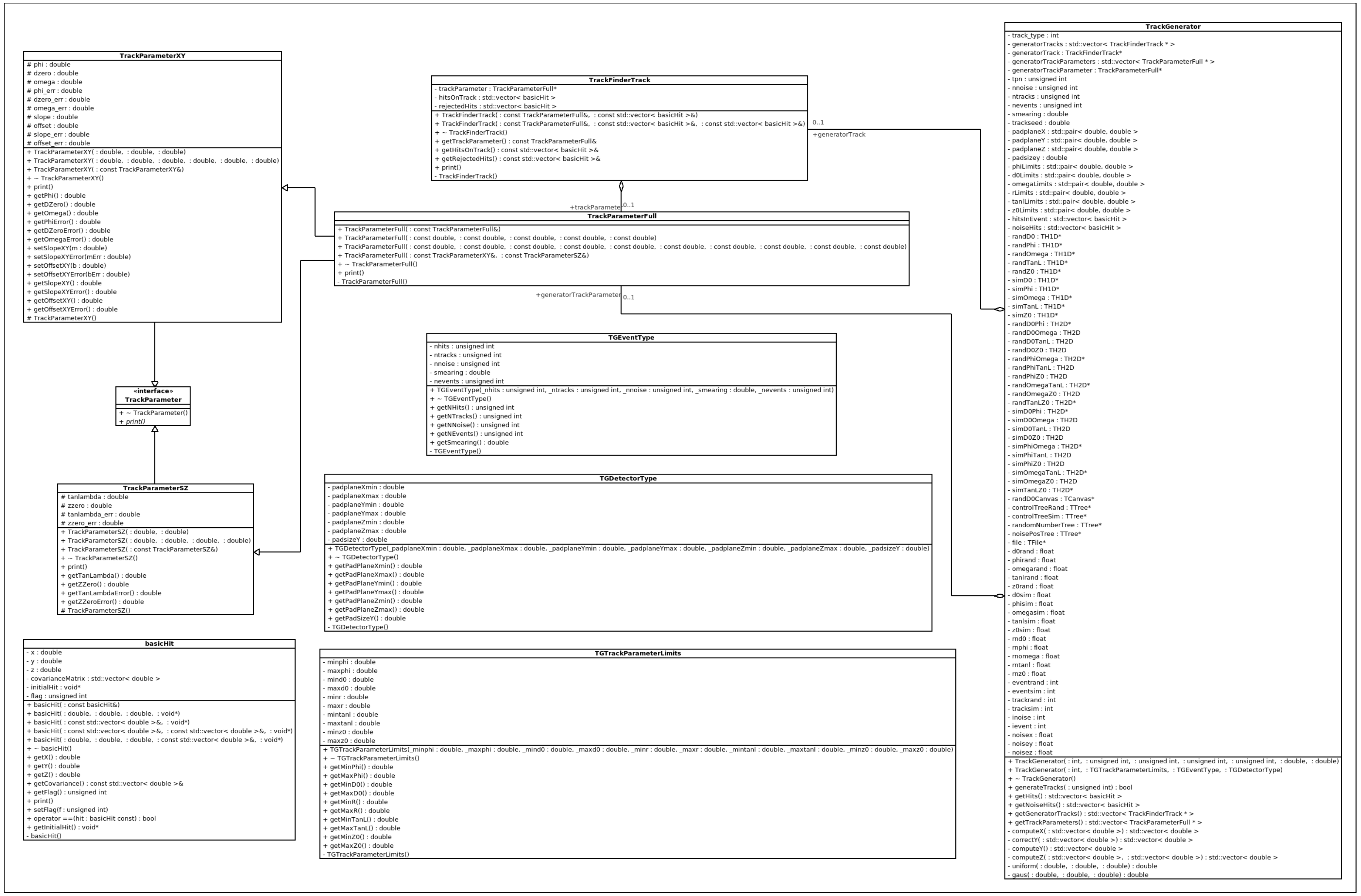


Figure C.1: Class diagram of Pathfinder (track finding).



```
TrackParameterXY
# phi : double
# dzero : double
# omega : double
# phi_err : double
# dzero_err : double
# omega_err : double
# slope : double
# offset : double
# slope_err : double
# offset_err : double
+ TrackParameterXY( : double,  : double,  : double)
+ TrackParameterXY( : double,  : double,  : double,  : double,  : double,  : double)
+ TrackParameterXY( : const TrackParameterXY&)
+ ~ TrackParameterXY()
+ print()
+ getPhi() : double
+ getDZero() : double
+ getOmega() : double
+ getPhiError() : double
+ getDZeroError() : double
+ getOmegaError() : double
+ setSlopeXY(m : double)
+ setSlopeXYError(mErr : double)
+ setOffsetXY(b : double)
+ setOffsetXYError(bErr : double)
+ getSlopeXY() : double
+ getSlopeXYError() : double
+ getOffsetXY() : double
+ getOffsetXYError() : double
# TrackParameterXY()
```

```
«interface»
TrackParameter
+ ~ TrackParameter()
+ print()
```

```
TrackParameterSZ
# tanlambda : double
# zzero : double
# tanlambda_err : double
# zzero_err : double
+ TrackParameterSZ( : double,  : double)
+ TrackParameterSZ( : double,  : double,  : double,  : double)
+ TrackParameterSZ( : const TrackParameterSZ&)
+ ~ TrackParameterSZ()
+ print()
+ getTanLambda() : double
+ getZZero() : double
+ getTanLambdaError() : double
+ getZZeroError() : double
# TrackParameterSZ()
```

```
basicHit
- x : double
- y : double
- z : double
- covarianceMatrix : std::vector< double >
- initialHit : void*
- flag : unsigned int
+ basicHit( : const basicHit&)
+ basicHit( : double,  : double,  : double,  : void*)
+ basicHit( : const std::vector< double >&,  : void*)
+ basicHit( : const std::vector< double >&,  : const std::vector< double >&,  : void*)
+ basicHit( : double,  : double,  : double,  : const std::vector< double >&,  : void*)
+ ~ basicHit()
+ getX() : double
+ getY() : double
+ getZ() : double
+ getCovariance() : const std::vector< double >&
+ getFlag() : unsigned int
+ print()
+ setFlag(f : unsigned int)
+ operator ==(hit : basicHit const) : bool
+ getInitialHit() : void*
- basicHit()
```

```
TrackFinderTrack
- trackParameter : TrackParameterFull*
- hitsOnTrack : std::vector< basicHit >
- rejectedHits : std::vector< basicHit >
+ TrackFinderTrack( : const TrackParameterFull&,  : const std::vector< basicHit >&)
+ TrackFinderTrack( : const TrackParameterFull&,  : const std::vector< basicHit >&,  : const std::vector< basicHit >&)
+ ~ TrackFinderTrack()
+ getTrackParameter() : const TrackParameterFull&
+ getHitsOnTrack() : const std::vector< basicHit >&
+ getRejectedHits() : const std::vector< basicHit >&
+ print()
- TrackFinderTrack()
```

```
TrackParameterFull
+ TrackParameterFull( : const TrackParameterFull&)
+ TrackParameterFull( : const double,  : const double,  : const double,  : const double,  : const double)
+ TrackParameterFull( : const double,  : const double,  : const double,  : const double,  : const double,  : const double,  : const double,  : const double,  : const double,  : const double)
+ TrackParameterFull( : const TrackParameterXY&,  : const TrackParameterSZ&)
+ ~ TrackParameterFull()
+ print()
- TrackParameterFull()
```

```
TGEventType
- nhits : unsigned int
- ntracks : unsigned int
- nnoise : unsigned int
- smearing : double
- nevents : unsigned int
+ TGEventType(_nhits : unsigned int, _ntracks : unsigned int, _nnoise : unsigned int, _smearing : double, _nevents : unsigned int)
+ ~ TGEventType()
+ getNHits() : unsigned int
+ getNTracks() : unsigned int
+ getNNoise() : unsigned int
+ getNEvents() : unsigned int
+ getSmearing() : double
- TGEventType()
```

```
TGDetectorType
- padplaneXmin : double
- padplaneXmax : double
- padplaneYmin : double
- padplaneYmax : double
- padplaneZmin : double
- padplaneZmax : double
- padsizeY : double
+ TGDetectorType(_padplaneXmin : double, _padplaneXmax : double, _padplaneYmin : double, _padplaneYmax : double, _padplaneZmin : double, _padplaneZmax : double, _padsizeY : double)
+ ~ TGDetectorType()
+ getPadPlaneXmin() : double
+ getPadPlaneXmax() : double
+ getPadPlaneYmin() : double
+ getPadPlaneYmax() : double
+ getPadPlaneZmin() : double
+ getPadPlaneZmax() : double
+ getPadSizeY() : double
- TGDetectorType()
```

```
TGTrackParameterLimits
- minphi : double
- maxphi : double
- mind0 : double
- maxd0 : double
- minr : double
- maxr : double
- mintanl : double
- maxtanl : double
- minz0 : double
- maxz0 : double
+ TGTrackParameterLimits(_minphi : double, _maxphi : double, _mind0 : double, _maxd0 : double, _minr : double, _maxr : double, _mintanl : double, _maxtanl : double, _minz0 : double, _maxz0 : double)
+ ~ TGTrackParameterLimits()
+ getMinPhi() : double
+ getMaxPhi() : double
+ getMinD0() : double
+ getMaxD0() : double
+ getMinR() : double
+ getMaxR() : double
+ getMinTanL() : double
+ getMaxTanL() : double
+ getMinZ0() : double
+ getMaxZ0() : double
- TGTrackParameterLimits()
```

```
TrackGenerator
- track_type : int
- generatorTracks : std::vector< TrackFinderTrack * >
- generatorTrack : TrackFinderTrack*
- generatorTrackParameters : std::vector< TrackParameterFull * >
- generatorTrackParameter : TrackParameterFull*
- tpn : unsigned int
- nnoise : unsigned int
- ntracks : unsigned int
- nevents : unsigned int
- smearing : double
- trackseed : double
- padplaneX : std::pair< double, double >
- padplaneY : std::pair< double, double >
- padplaneZ : std::pair< double, double >
- padsizey : double
- phiLimits : std::pair< double, double >
- d0Limits : std::pair< double, double >
- omegaLimits : std::pair< double, double >
- rLimits : std::pair< double, double >
- tanlLimits : std::pair< double, double >
- z0Limits : std::pair< double, double >
- hitsInEvent : std::vector< basicHit >
- noiseHits : std::vector< basicHit >
- randD0 : TH1D*
- randPhi : TH1D*
- randOmega : TH1D*
- randTanL : TH1D*
- randZ0 : TH1D*
- simD0 : TH1D*
- simPhi : TH1D*
- simOmega : TH1D*
- simTanL : TH1D*
- simZ0 : TH1D*
- randD0Phi : TH2D*
- randD0Omega : TH2D
- randD0TanL : TH2D
- randD0Z0 : TH2D
- randPhiOmega : TH2D*
- randPhiTanL : TH2D
- randPhiZ0 : TH2D
- randOmegaTanL : TH2D*
- randOmegaZ0 : TH2D
- randTanLZ0 : TH2D*
- simD0Phi : TH2D*
- simD0Omega : TH2D
- simD0TanL : TH2D
- simD0Z0 : TH2D
- simPhiOmega : TH2D*
- simPhiTanL : TH2D
- simPhiZ0 : TH2D
- simOmegaTanL : TH2D*
- simOmegaZ0 : TH2D
- simTanLZ0 : TH2D*
- randD0Canvas : TCanvas*
- controlTreeRand : TTree*
- controlTreeSim : TTree*
- randomNumberTree : TTree*
- noisePosTree : TTree*
- file : TFile*
- d0rand : float
- phirand : float
- omegarand : float
- tanlrand : float
- z0rand : float
- d0sim : float
- phisim : float
- omegasim : float
- tanlsim : float
- z0sim : float
- rnd0 : float
- rnphi : float
- rnomega : float
- rntanl : float
- rnz0 : float
- eventrand : int
- eventsim : int
- trackrand : int
- tracksim : int
- inoise : int
- ievent : int
- noisex : float
- noisey : float
- noisez : float
+ TrackGenerator( : int,  : unsigned int,  : unsigned int,  : unsigned int,  : unsigned int,  : double,  : double)
+ TrackGenerator( : int,  : TGTrackParameterLimits,  : TGEventType,  : TGDetectorType)
+ ~ TrackGenerator()
+ generateTracks( : unsigned int) : bool
+ getHits() : std::vector< basicHit >
+ getNoiseHits() : std::vector< basicHit >
+ getGeneratorTracks() : std::vector< TrackFinderTrack * >
+ getTrackParameters() : std::vector< TrackParameterFull * >
- computeX( : std::vector< double >) : std::vector< double >
- correctY( : std::vector< double >) : std::vector< double >
- computeY() : std::vector< double >
- computeZ( : std::vector< double >,  : std::vector< double >) : std::vector< double >
- uniform( : double,  : double,  : double) : double
- gaus( : double,  : double,  : double) : double
```

Figure C.2: Class diagram of Pathfinder (track generation).

Appendix D

Efficiency Plots

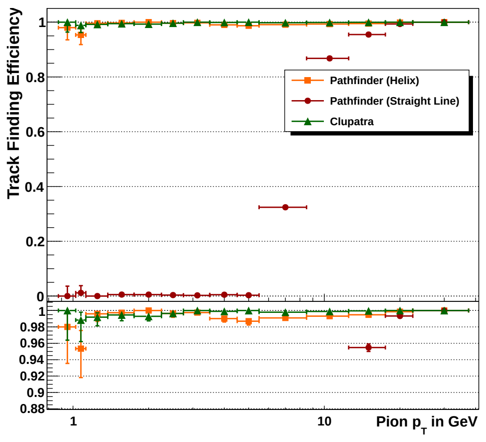


Figure D.1: Track finding efficiencies for taus decaying to pions.

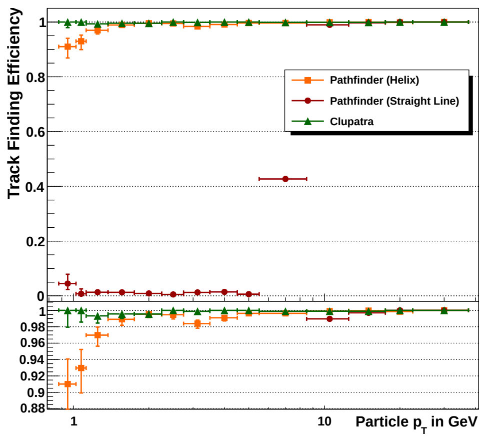


Figure D.2: Track finding efficiencies for 1-prong tau decays.

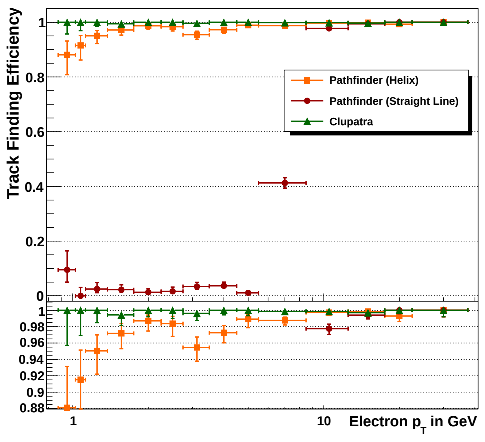


Figure D.3: Track finding efficiencies for 1-prong decays (taus decaying to an electron).

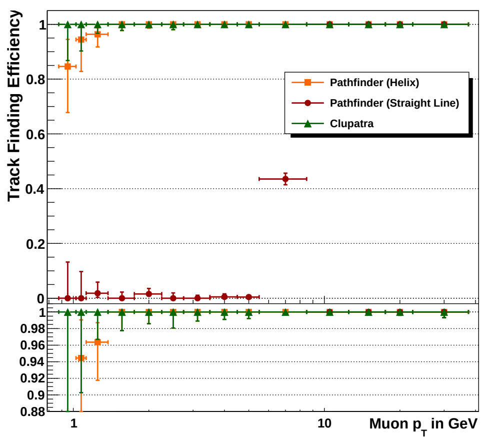


Figure D.4: Track finding efficiencies for 1-prong decays (taus decaying to a muon).

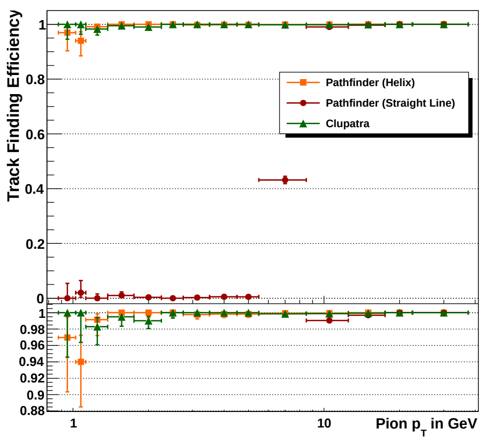


Figure D.5: Track finding efficiencies for 1-prong decays (taus decaying to pions).

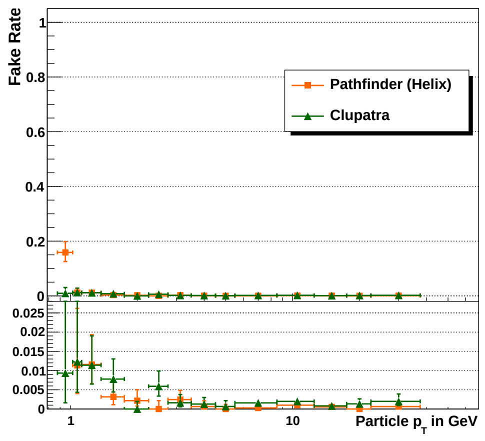


Figure D.6: Fake rate for all decay modes.

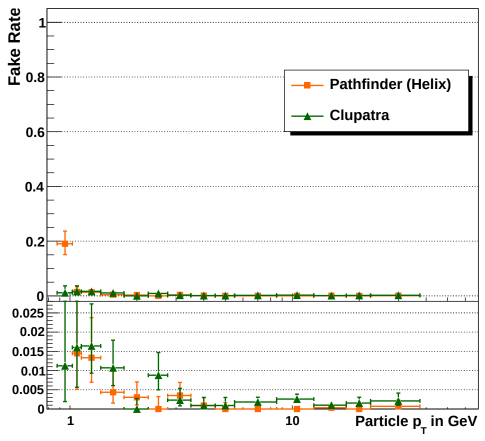


Figure D.7: Fake rate for 1-prong decays.

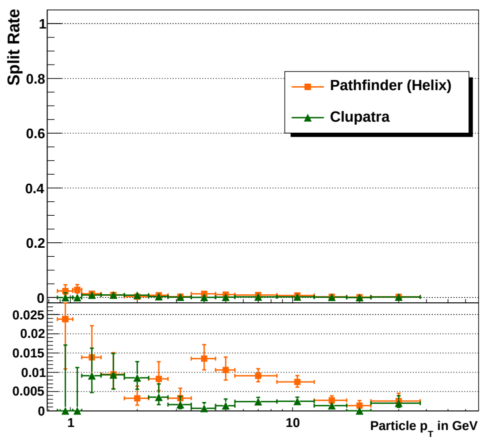


Figure D.8: Split rate for all decay modes.

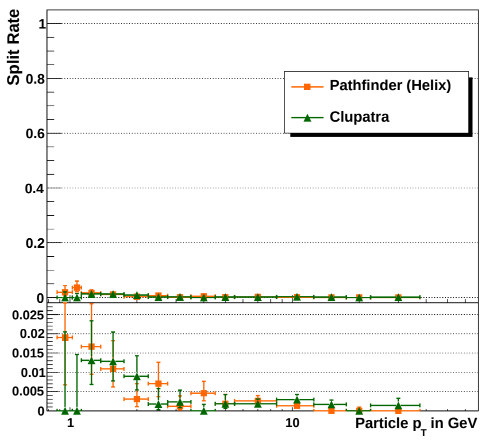


Figure D.9: Split rate for 1-prong decays.

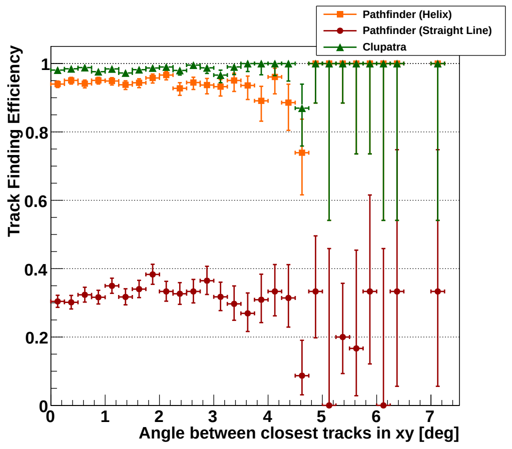


Figure D.10: Track finding efficiencies for taus decays (3-prong events) for different angles in xy.

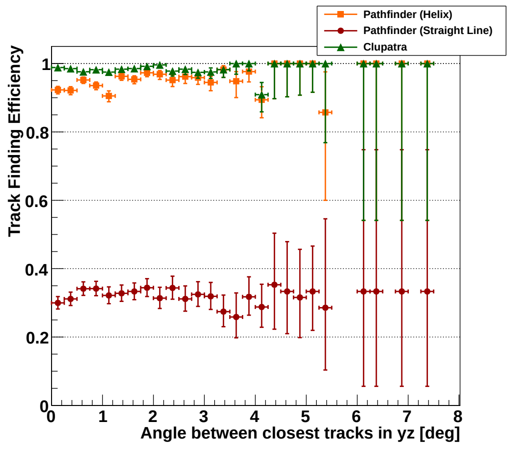


Figure D.11: Track finding efficiencies for taus decays (3-prong events) for different angles in yz.

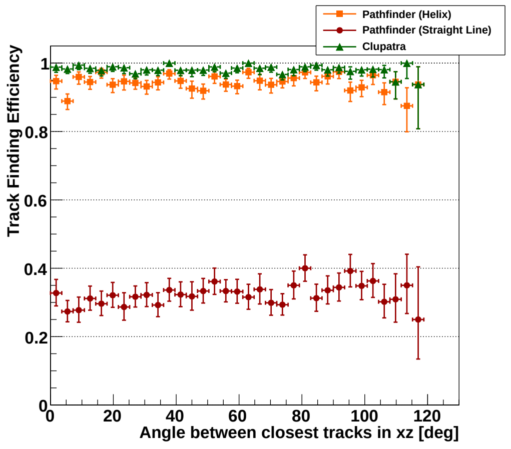


Figure D.12: Track finding efficiencies for taus decays (3-prong events) for different angles in xz.

Appendix E

Testbeam Runs

Table E.1: List of runs taken during the testbeam in June 2011 without magnetic field. All runs contain 20000 events and all runs were taken at a vertical table position of -7.493 mm and a table rotation of 0 rad.

Run	**Date, Start Time**	**Table Position horizontal**	**Measurement**
17692	July 4, 2011 16:40	-151.944 mm	resolution drift velocity
17693	July 4, 2011 17:31	-202.285 mm	resolution drift velocity
17694	July 4, 2011 17:31	-252.279 mm	resolution drift velocity
17697	July 4, 2011 19:23	-103.727 mm	resolution drift velocity
17698	July 4, 2011 19:42	-53.908 mm	resolution drift velocity
17699	July 4, 2011 20:08	-3.935 mm	resolution drift velocity
17700	July 4, 2011 20:36	46.169 mm	resolution drift velocity
17701	July 4, 2011 20:59	96.180 mm	resolution drift velocity
17702	July 4, 2011 21:25	145.952 mm	resolution drift velocity
17703	July 4, 2011 21:49	196.293 mm	resolution drift velocity
17704	July 4, 2011 22:11	246.254 mm	resolution
17705	July 4, 2011 22:31	276.148 mm	anode position
17708	July 5, 2011 09:13	-278.135 mm	resolution
17709	July 5, 2011 09:27	-282.594 mm	resolution
17711	July 5, 2011 10:24	-273.335 mm	resolution
17712	July 5, 2011 10:52	-228.869 mm	resolution

Run	**Date, Start Time**	**Table Position horizontal**	**Measurement**
17713	July 5, 2011 11:12	271.150 mm	cathode position resolution
17714	July 5, 2011 12:06	278.161 mm	cathode position
17715	July 5, 2011 12:21	277.696 mm	cathode position
17716	July 5, 2011 12:38	281.181 mm	cathode position
17717	July 5, 2011 13:02	283.770 mm	cathode position
17718	July 5, 2011 13:30	-280.040 mm	anode position
17719	July 5, 2011 13:42	-274.150 mm	anode position

Table E.2: List of runs taken during the testbeam in June 2011 with a 1T magnetic field performing a z scan. All runs contain 20000 events and all runs were taken at a vertical table position of 7.72 mm and a table rotation of 0 rad. Only one of the four sectors of the module was available.

Run	**Date, Start Time**	**Table Position horizontal**	**Measurement**
17765	July 8, 2011 15:36	250.87 mm	resolution drift velocity
17766	July 8, 2011 15:48	150.57 mm	resolution drift velocity
17767	July 8, 2011 16:04	0.42 mm	resolution drift velocity
17768	July 8, 2011 16:18	-149.81 mm	resolution drift velocity
17769	July 8, 2011 16:31	-249.85 mm	resolution drift velocity
17770	July 8, 2011 16:48	-264.95 mm	resolution drift velocity

Table E.3: List of runs taken during the testbeam in June 2011 with a 1T magnetic field performing a drift field scan. All runs contain 20000 events and all runs were taken at a vertical table position of 7.72 mm and a table rotation of 0 rad. Only one of the four sectors of the module was available.

Run	**Date, Start Time**	**Table Position horizontal**	**Drift Field**	**Measurement**
17772	July 8, 2011 17:06	-153.540 mm	23.998 V/mm	drift velocity
17773	July 8, 2011 17:27	146.770 mm	23.998 V/mm	drift velocity
17774	July 8, 2011 17:48	-153.165 mm	22.996 V/mm	drift velocity
17775	July 8, 2011 18:04	146.660 mm	22.996 V/mm	drift velocity
17776	July 8, 2011 18:14	-153.220 mm	20.997 V/mm	drift velocity
17778	July 8, 2011 18:48	146.690 mm	20.997 V/mm	drift velocity
17779	July 8, 2011 18:56	-153.200 mm	16.004 V/mm	drift velocity
17780	July 8, 2011 19:13	146.680 mm	16.004 V/mm	drift velocity
17781	July 8, 2011 19:30	-153.200 mm	18.998 V/mm	drift velocity
17782	July 8, 2011 19:45	146.620 mm	18.998 V/mm	drift velocity
17783	July 8, 2011 20:08	-153.200 mm	18.002 V/mm	drift velocity
17784	July 8, 2011 20:21	146.620 mm	18.002 V/mm	drift velocity
17788	July 8, 2011 20:47	-153.170 mm	10.002 V/mm	drift velocity
17789	July 8, 2011 21:34	146.840 mm	10.002 V/mm	drift velocity
17792	July 9, 2011 10:43	-153.200 mm	12.500 V/mm	drift velocity
17795	July 9, 2011 11:19	146.640 mm	12.500 V/mm	drift velocity

Bibliography

[1] S. L. Glashow. Partial-symmetries of weak interactions. *Nuclear Physics*, 22:579, 1961.

[2] S. Weinberg. A Model of Leptons. *Physical Review Letters*, 19, 1967.

[3] A. Salam. *Elementary particle theory: relativistic groups and analyticity*, chapter Weak and electromagnetic interaction. Almqvist & Wiksell.

[4] J. Beringer et al. (Particle Data Group). Review of Particle Physics. *Physical Review D*, 86, 2012. `http://pdg.lbl.gov` Accessed: June 7, 2013.

[5] The ATLAS Collaboration. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Physics Letters B*, 716:1–29, 2012.

[6] The CMS Collaboration. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Physics Letters B*, 716:30, 2012.

[7] P. W. Higgs. Broken symmetries, massless particles and gauge fields. *Physics Letters*, 12:132, 1964.

[8] P. W. Higgs. Broken symmetries and the masses of gauge bosons. *Physical Review Letters*, 13:508, 1964.

[9] P. W. Higgs. Spontaneous symmetry breakdown without massless bosons. *Physical Review*, 145:1156, 1966.

[10] L. Evans and P. Bryant. LHC Machine. *Journal of Instrumentation*, 3(08), 2008.

[11] The CDF and DØ Collaboration. Evidence for a particle produced in association with weak bosons and decaying to a bottom-antibottom quark pair in Higgs boson searches at the Tevatron. *Physical Review Letters*, 109, 2012. arXiv:1207.6436.

[12] G. Bertone, D. Hooper and J. Silk. Particle Dark Matter: Evidence, Candidates and Constraints. *Physics Reports*, 405:279, 2005.

[13] D. N. Spergel, L. Verde, H. V. Peiris, E. Komatsu, M. R. Nolta, C. L. Bennett, M. Halpern, G. Hinshaw, N. Jarosik, A. Kogut, M. Limon, S. S. Meyer,

L. Page, G. S. Tucker, J. L. Weiland, E. Wollack, and E. L. Wright. First-Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Determination of Cosmological Parameters. *The Astrophysical Journal Supplement Series*, 148(1):175, 2003.

[14] Planck Collaboration. Planck 2013 results. XVI. Cosmological parameters. 2013. arXiv:1303.5076.

[15] H. Georgi and S. L. Glashow. Unity of All Elementary-Particle Forces. *Physical Review Letters*, 32:438–441, 1974.

[16] A.J. Buras, J.R. Ellis, M.K. Gaillard, D.V. Nanopoulos. Aspects of the Grand Unification of Strong, Weak and Electromagnetic Interactions. *Nuclear Physics B*, 135:66–92, 1977.

[17] U. Amaldi, W. de Boer, and H. Fürstenau. Comparison of grand unified theories with electroweak and strong coupling constants measured at LEP. *Physics Letters B*, 260(3–4):447–455, 1991.

[18] J. Wess and B. Zumino. A Lagrangian Model invariant under Supergauge Transformations. *Physics Letters B*, 49, 1974.

[19] A. Salam and J. Strathdee. Super-Symmetry and non-abelian Gauges. *Physics Letters B*, 51, 1974.

[20] A. Salam and J. Strathdee. Superfields and Fermi-Bose symmetry. *Physics Letters D*, 11:1521 – 1535, 1975.

[21] D. I. Kazakov. Beyond the Standard Model - in Search for Supersymmetry. 2000. arXiv:hep-ph/0012288.

[22] S. Myers, E. Picasso. The design, construction and comissioning of the CERN Large Electron-Positron collider. *Contemporary Physics*, 31(6):387–403, 1990.

[23] N. Phinney, N. Toge, and N. Walker. International Linear Collider Reference Design Report Volume 3 - Accelerator. 2007. arXiv:0712.2361.

[24] The International Linear Collider Technical Design Report - Volume 3 Part II: Accelerator Baseline Design, 2013. CERN-ATS-2013-037, DESY 13-062, ILC-REPORT-2013-040.

[25] LCC Homepage. `http://www.linearcollider.org` Accessed: May 6, 2013.

[26] ILD Concept Group. The International Large Detector - Letter of Intent. February 2010. DESY 2009-87.

[27] The International Linear Collider Technical Design Report - Volume 4: Detectors, 2013. CERN-ATS-2013-037, DESY 13-062, ILC-REPORT-2013-040.

[28] ILD Homepage. `http://ilcild.org/` Accessed: March 1, 2013.

[29] ACFA Working Group Webpage. `http://acfahep.kek.jp/` Accessed: May 14, 2013.

[30] The International Linear Collider Technical Design Report - Volume 2: Physics, 2013. CERN-ATS-2013-037, DESY 13-062, ILC-REPORT-2013-040.

[31] H. Li. Higgs Recoil Mass and Higgs-Strahlung Cross-Section Study for the ILD LOI. 2010. arXiv:1007.2999.

[32] G. Aarons et al. International Linear Collider Reference Design Report Volume 2: Physics at the ILC. 2007. arXiv:0709.1893.

[33] H. Baer and J. List. Post LHC7 SUSY benchmark points for ILC physics. 2012. arXiv:1205.6929.

[34] P. Huang X. Tata H. Baer, V. Barger. Natural Supersymmetry: LHC, dark matter and ILC searches. 2012. arXiv:1203.5539.

[35] P. Schade. Developement and Construction of a Large TPC Prototype for the ILC and Study of τ Polarisation in $\widetilde{\tau}$ Decays with the ILD Detector. DESY-THESIS-09-040.

[36] J.A. Aguilar-Saavedra et al. Supersymmetry Parameter Analysis: SPA Convention and Project. 2005. arXiv:hep-ph/0511344.

[37] D.R. Nygren and J.N. Marx. The time projection chamber. *Physics Today*, 31:46–53, 1978.

[38] O. Schäfer. Ein Monitorsystem für gasbasierte Detektoren am International Linear Collider (ILC). Master's thesis, Universität Rostock, Fachbereich Physik, 2005. `http://adweb.desy.de/~oschfer` Accessed: July 30, 2013.

[39] W. Blum and L. Rolandi. *Particle Detection with Drift Chambers*. Springer, Berlin, 1993.

[40] T. Lohse and W. Witzeling. The Time Projection Chamber. In F. Sauli, editor, *Advanced Series on Directions in High Energy Physics Volume 9: Instrumentation in High Energy Physics*, pages 81–155. World Scientific, 1992.

[41] P. Gros. Private communication.

[42] R. L. Glückstern. Uncertainties in Track Momentum and Direction, due to Multiple Scattering and Measurement Errors. *Nuclear Instruments and Methods*, 24:381–389, 1963.

[43] Ch. K. Bowdery. *The ALEPH handbook: 1995*. CERN, Geneva, 1995.

[44] G. Dellacasa, et al. ALICE Technical Design Report of the Time Projection Chamber. CERN-OPEN-2000-183.

[45] P.A. Aarnio, et al. The DELPHI detector at LEP. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 303, 1991.

[46] M. Anderson. The STAR time projection chamber: a unique tool for studying high multiplicity events at RHIC. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 499:659–678, 2003.

[47] D. Buskulic et al. Performance of the ALEPH detector at LEP. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 360:481–506, 1995.

[48] J. Alme et al. The ALICE TPC, a large 3-dimensional tracking device with fast readout for ultra-high multiplicity events. 2010. arXiv:1001.1950v1.

[49] F. Sauli. GEM: A new concept for electron amplification in gas detectors. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 386.

[50] Webpage of the CERN GDD group. `http://gdd.web.cern.ch/GDD/` Accessed: March 5, 2013.

[51] O. Schäfer. Private communication, simulation by Blanka Sobloher.

[52] K. Zenker. Simulation with Garfield. `http://www.desy.de/~zenker/garfieldpp.html` Accessed: June 7, 2013.

[53] S. Bachmann et al. Discharge studies and prevention in the gas electron multiplier (GEM). *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 479:294–308, 2002.

[54] R. Settles. LCTPC and the Magnetic Field for ILD: Update 2010. 2011. LC Note LC-DET-2011-002.

[55] S. Agostinelli et al. Geant4 - a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506:250–303, July 2003.

[56] J. Allison et al. Geant4 developments and applications. *IEEE Transactions on Nuclear Science*, 53(1):270–278, February 2006.

[57] F. Gaede, T. Behnke, N. Graf, and T. Johnson. LCIO - A persistancy framefork for linear collider simulation studies. 2003. LC Note LC-TOOL-2003-053, arXiv:physics/0306114v1.

[58] LCIO Homepage. `http://lcio.desy.de` Accessed: May 23, 2012.

[59] J. Alcaraz. Helicodial Tracks. Februar 18 1995. L3 Internal Note 1666.

[60] T. Krämer. Track Parameters in `LCIO`. 2006. LC_DET_2006-004.

[61] GEAR Homepage. `http://ilcsoft.desy.de/portal/software_packages/gear/` Accessed: February 28, 2013.

[62] Extensible Markup Language. `http://www.w3.org/XML/` Accessed: June 4, 2013.

[63] Mokka Homepage. `http://ilcsoft.desy.de/portal/software_packages/mokka/` Accessed: February 28, 2013.

[64] P. M. de Freitas. Detector Simulation with MOKKA and Geant4: Present and Future. 2003. LC Note LC-TOOL-2003-010.

[65] Marlin Homepage. `http://ilcsoft.desy.de/portal/software_packages/marlin` Accessed: May 23, 2012.

[66] J. Abernathy, K. Dehmelt, R. Diener, J. Hunt, M. E. Janssen, M. Killenberg, T. Krautscheid, A. Münnich, M. Ummenhofer, A. Vogel, and P. Wienemann. MarlinTPC: A Marlin based common TPC software framework for the LC-TPC collaboration. 2007. LC Note LC-TOOL-2007-001.

[67] MarlinTPC Homepage. `http://ilcsoft.desy.de/portal/software_packages/marlintpc` Accessed: May 23, 2012.

[68] MarlinTPC Wiki. `https://znwiki3.ifh.de/MarlinTPC` Accessed: May 23, 2013.

[69] M. E. Janssen. Auflösungsstudien an einer Zeit-Projectionskammer (TPC) mit GEM Gasverstärkungssystem. DESY-THESIS-2004-049.

[70] T. Lux. Studies for a Time Projection Chamber for the International Linear Collider and Measurement of Beauty Cross Sections in Deep Inelastic Scattering at HERA. DESY-THESIS-2005-019.

[71] Gas Properties Studies with Magboltz by Akimasa Ishikawa. `http://www-hep.phys.saga-u.ac.jp/ILC-TPC/gas/index.html` Accessed: October 28, 2012.

[72] P. V. C. Hough et al. Method and Means for Recognizing Complex Patterns. 1962. US Patent 3,069,654.

[73] J. Blom, C.T.A.M. de Laat, P.G. Kuijer, and W. Lourens. A fuzzy radon transform for track recognition. *Proc. Computing in High Energy Physics Converence, San Francisco.*

[74] J. J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Science of the United States of America*, 79:2554–2558, 1982.

[75] B. Denby. Neural Networks and Cellular Automata in Experimental High Energy Physics. *Computer Physics Communications*, 49:429–488, 1988.

[76] C. Peterson. Track Finding with Neural Networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 279:537–549, 1989.

[77] H. D. Schulz and H. J. Stuckenberg. A trigger processor for ARGUS. In *Topical Conference on the Application of Microprocessors to High-energy Physics Experiments*, pages 194–203, 1981.

[78] N. Koch et al. The ARGUS Vertex Trigger. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 373:387–405, 1996.

[79] M. Hansroulm, H. Jeremie, D. Savard. Fast Circle Fit with the Conformal Mapping Method. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, pages 498–501, 1988.

[80] R. Frühwirth. Application of Kalman filtering to track and vertex fitting. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 262:444–450, December 1987.

[81] R. Mankel. Pattern Recognition and Event Reconstruction in Patricle Physics Experiments. arXiv:physics/0402039v1.

[82] R. Mankel. A Concurrent Track Evolution Algorithm for Pattern Recognition in the HERA-B Main Tracking System. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 395, 1997.

[83] S. Burke et al. Track finding and fitting in the H1 Forward Track Detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, pages 227–260, 1996.

[84] A. Strandlie and R. Frühwirth. Track and vertex reconstruction: From classical to adaptive methods. *Review of Modern Physics*, 82:1419–1458, 2010.

[85] P. V. C. Hough. Machine Analysis of Bubble Chamber Pictures. In *Proceedings of the International Converence on High Energy Acceleration and Instrumentation*, pages 554–556, CERN, Geneva, 1959.

[86] J. Illingworth and J. Kittler. The Adaptive Hough Transformation. *IEEE Transactions on Pattern Analysis amd Machine Intelligence*, PAMI-9:690–698, September 1987.

[87] T. Behnke, S. Bertolucci, R. D. Heuer, R. Settles (editors). TESLA: The Superconducting Electron Positron Linear Collider with an Integrated X-Ray Laser Laboratory. Technical Design Report, Pt IV: A Detector for TESLA. 2001. DESY 01-011, DESY-TESLA-2001-23, DESY-TESLA-FEL-2001-05, ECFA-2001-209.

[88] P. Schade and J. Kaminski on behalf of the LCTPC collaboration. A large TPC prototype for a linear collider detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 628:128–132, 2011.

[89] V. Karimäki. Effective circle fitting for particle trajectories. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 305:187–191, 1991.

[90] R. Brun and F. Rademakers. ROOT - An Object Oriented Data Analysis Framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389:81–86, 1997. Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, see also `http://root.cern.ch` (Accessed: 16/11/2012).

[91] cplusplus.com - The C++ Resources Network. `http://www.cplusplus.com/` Accessed: May 21, 2013.

[92] A. Vogel. Beam-Induced Backgrounds in Detectors at the ILC. DESY-THESIS-08-036.

[93] F. Gaede. Clupatra - Topological TPC pattern recognition. AIDA WP2 Meeting 2011, CERN, October 2011. `http://indico.cern.ch/getFile.py/access?contribId=9&resId=0&materialId=slides&confId=159340` Accessed: February 28, 2013.

[94] K. Fujii (ACFA Sim-J Group). Extended Kalman Filter. `http://www-jlc.kek.jp/subg/offl/kaltest/` Accessed: June 7, 2013.

[95] ilcsoft Homepage. `http://ilcsoft.desy.de` Accessed: June 7, 2013.

[96] Mokka Detector Model ILD_O1_v03. `http://www-flc.desy.de/ldcoptimization/tools/mokkamodels.php` Accessed: June 7, 2013.

[97] MarlinReco Homepage. `http://ilcsoft.desy.de/portal/software_packages/marlinreco` Accessed: March 17, 2013.

[98] T. Behnke et al. Testbeams at DESY. June 2007. EUDET-Memo-2007-11.

[99] DESY Testbeam Webpage. `testbeam.desy.de` Accessed: February 16, 2013.

[100] S. Caiazza. Private communication, PhD thesis in preparation. `http://www-flc.desy.de/tpc/projects/desy_lp_module/index.php` Accessed: June 5, 2013.

[101] L. Jönsson and U. Mjörnmark. Front-end electronics and data acquisition for the LCTPC. January 2008. EUDET-Memo-2007-53.

[102] K. Zenker. Study on electrostatic distortions and possible corrections for a GEM based TPC readout module. 2013. LC Note LC-DET-2013-017.

[103] R. Diener. Private communication.

[104] R. Diener. Study of Reconstruction Methods for a Time Projection Chamber with GEM Gas Amplificaqtion System. DESY-THESIS-2006-040.

[105] R. K. Carnegie et al. Resolution studies of cosmic-ray tracks in a TPC with GEM readout. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 538:372–383, 2005.

[106] K. Zenker. Private communication.

[107] F. Müller. Private communication.

Index

Acknowledgments

First of all I would like to thank the reviewers of my dissertation and disputation: Prof. Dr. Erika Garutti, Dr. Philip Bechtle and Prof. Dr. Johannes Haller. Additionally I would like to thank Dr. Katarzyna Wichmann, Dr. Katja Krüger, Dr. Martin Killenberg and Dr. Steven Aplin who kindly agreed to be questioners for my disputation and Dr. Georg Steinbrück who agreed to be the chairperson for my disputation.
I would like to thank Dr. Ties Behnke for giving me the opportunity to work in the FLC group at DESY. Furthermore I would like to thank my supervisors Dr. Christoph Rosemann and Dr. Astrid Münnich. For helpful discussions I would like to thank Dr. Claus Kleinwort, Dr. Steven Aplin, Dr. Frank Gaede and Ralf Diener, Oliver Schäfer, Stefano Caiazza, Felix Müller and Klaus Zenker. I would also like to thank all current and former members of the FLC group (if not named before), especially Andrea Schrader and Ramona Matthes.
Finally I would like to thank Dr. Sebastian Schmidt for interesting discussions (not always about physics) and Antje Krüger for being a good friend in all those years since I started studying physics.