TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Diplomarbeit

Track Reconstruction in the Forward Region of the Detector ILD at the Electron-Positron Linear Collider ILC

Ausgeführt am

Institut für Hochenergiephysik (HEPHY)
der Österreichischen Akademie der Wissenschaften (ÖAW)

unter der Anleitung von

Univ.Doz. Dipl.-Ing. Dr.techn. Rudolf Frühwirth
Dipl.-Ing. Dr. Winfried Mitaroff

durch

Robin Glattauer

Wien, im Oktober 2012

Abstract

The subject of this thesis is the reconstruction of charged particle tracks in the forward region of the International Large Detector (ILD), one of two validated detector concepts for the future International Linear Collider (ILC).

Recent results from the Large Hadron Collider (LHC) suggest that the last missing piece of the Standard Model (SM) of particle physics, the Higgs boson, could have been found. Complementary to the LHC, an electron-positron linear collider would have the capability to explore with high precision the characteristics of the Higgs boson (e.g. spin, parity, coupling strengths), to compare them against the predictions of the SM, and also give unbiased contributions to the search for physics beyond the SM, such as supersymmetry or extra spatial dimensions.

Experiments at ILD will benefit from this detector's tracking system, consisting of a large central time projection chamber (TPC) augmented by several silicon tracking systems, granting unprecedented track resolution, redundancy and angular hermeticity. The forward region of ILD, covering the space between beam tube and TPC, contains the Forward Tracking Detector (FTD): two arms of seven disk-shaped silicon detectors (two Si pixel and five double-sided Si strip detectors).

In order to fully exploit ILD's hardware assets, the software for event reconstruction must aim for both the highest level of precision and efficiency. In this context, new software for track reconstruction in the forward region was developed by the author (packages `KiTrack` and `ForwardTracking`): it processes the signals of FTD with the goal to efficiently find and precisely reconstruct the genuine tracks that have traversed FTD and caused these signals.

The methods used are based on state-of-the-art algorithms: a cellular automaton (CA), a Kalman filter (KF), and a Hopfield neural network (HNN). The new packages follow a modern object-oriented design philosophy, granting high flexibility and maintainability. The results show superior performance w.r.t. older legacy software, yielding higher efficiencies and better handling of the expected background concerning ghost rate, efficiency and processing time.

The forward tracking packages presented in this thesis have been successfully implemented into the standard event reconstruction framework of ILD. They are currently used for benchmark event processing for the Detailed Baseline Design (DBD), a report outlining the feasibility and features of the International Large Detector, to be published around the end of this year.

Kurzfassung

Thema dieser Diplomarbeit ist die Rekonstruktion der Spuren geladener Teilchen im Vorwärtsbereich des "International Large Detector" (ILD), einem von zwei bestätigten Detektorkonzepten am zukünftigen "International Linear Collider" (ILC).

Neuere Ergebnisse am "Large Hadron Collider" (LHC) lassen vermuten, dass der letzte noch fehlende Baustein des Standardmodells (SM) der Teilchenphysik, das Higgs-Boson, gefunden wurde. Komplementär zu LHC hätte ein Elektron-Positron Linear-Collider die Fähigkeit, mit hoher Präzision die Eigenschaften des Higgs-Bosons (z.B. Spin, Parität, Kopplungsstärken) zu erforschen, diese mit den Vorhersagen des SM zu vergleichen, und auch annahmefrei zur Suche nach Physik jenseits des SM, wie Supersymmetrie oder zusätzliche Raumdimensionen, beizutragen.

Experimente am ILD werden vom Tracking-System dieses Detektors profitieren, das aus einer großen Zeitprojektionskammer ("time projection chamber", TPC) und mehreren Silizium-Spurdetektoren besteht, und das eine bisher unerreichte Spurauflösung, Redundanz und Raumwinkel-Hermetizität gewährt. Der Vorwärtsbereich von ILD, welcher den Raum zwischen Strahlrohr und TPC abdeckt, enthält den "Forward Tracking Detector" (FTD): zwei Arme mit je sieben scheibenförmigen Siliziumdetektoren (zwei Pixel- und fünf doppelseitige Streifen-Detektoren).

Um die Hardware-Vorteile von ILD voll nutzen zu können, muss die Software zur Rekonstruktion der Kollisionsereignisse sowohl auf Präzision als auch auf Effizienz ausgerichtet sein. In diesem Zusammenhang wurde vom Autor neue Software zur Spurrekonstruktion im Vorwärtsbereich entwickelt (Pakete `KiTrack` und `ForwardTracking`): diese verarbeitet die Signale des FTD mit dem Ziel, die wahren Spuren, welche den FTD durchquert und diese Signale verursacht haben, effizient zu finden und präzise zu rekonstruieren.

Die benutzten Methoden basieren auf Algorithmen am neuesten Stand der Technik: einem zellulären Automaten, einem Kalman-Filter, und einem neuralen Hopfield-Netz. Die neuen Softwarepakete folgen einer modernen objektorientierten Philosophie, welche hohe Flexibilität und Wartbarkeit garantiert. Die Ergebnisse zeigen eine überlegene Leistung im Vergleich zu bisher verwendeter Software: gesteigerte Effizienzen, und eine besserer Bewältigung der erwarteten Untergrundsignale was Artefakte, Effizienz und Rechenzeit betrifft.

Die in dieser Diplomarbeit vorgestellten Softwarepakete zum Vorwärts-Tracking wurden erfolgreich ins reguläre ILD-Framework zur Rekonstruktion von Kollisionsereignissen implementiert. Sie werden gegenwärtig in den Benchmark-Berechnungen für den "Detailed Baseline Design" (DBD) eingesetzt – einem Report, der die Machbarkeit und Grundeigenschaften des "International Large Detector" darlegt, und der um das Jahresende 2012 publiziert werden soll.

Contents

List of Figures

1. Introduction

1.1. Overview

The subject of this thesis is the reconstruction of charged particle tracks in the forward region[1] of the International Large Detector ILD [28], one of two validated detectors at a future electron-positron linear collider, the ILC [16].

The main goal achieved by this work is software developed for track reconstruction in the forward direction; in addition, contributions were made also to other parts of the reconstruction chain, including digitizing simulated hits, analysis (true track finder), and arbitration of different track collections into a unique subset.

Chapter 1, after a short review of the present theory, gives an overview of the experimental environment at high energy colliders, together with the main aspects of the ILD detector at ILC, with which this thesis is primarily concerned.

In Chapter 2, algorithms for pattern recognition are discussed, and in Chapter 3 recent approaches to track reconstruction are presented. Chapter 4 describes the implementation of the algorithms for track reconstruction in the ILD detector, and the software packages developed for this task. Chapter 5 gives the results of various benchmark tests for the tracking software, including a comparison with the older algorithms used for forward track reconstruction in the ILD.

Conclusions are drawn in Chapter 6, and an outlook is given in Chapter 7.

1.2. The Standard Model of particle physics

Our present knowledge of the microcosm is well described by two non-Abelian local gauge theories called the "Standard Model" (SM). It includes the so far discovered fundamental particles and their mediating forces: electromagnetic, weak and strong force. The fourth force – gravitation – is not included in the Standard Model, but efforts are made to find a theory ultimately combining all forces.

The Standard Model has been very successfully verified, and has been probed to distances as small as 10^{-18} m, with an accuracy of about 1‰ or 1% in the electroweak and strong sectors, respectively. Only a few basics are given below; for details refer to a textbook, e.g. [54].

The Electroweak Theory extends Quantum Electrodynamics (QED) by unifying electromagnetic and weak interactions, based on the gauge symmetry $SU(2)_{\mathrm{L}} \times U(1)$

[1] defined by the angle w.r.t. the beams; the backward region is always implicitly regarded as well.

of weak isospin and hypercharge. This symmetry is spontaneously broken by the "Higgs mechanism", introducing a heavy scalar particle, the Higgs boson (H^0) and giving the weak vector bosons ($W^\pm$ and Z^0) their masses while leaving the photon (γ) massless. The fundamental fermions gain their masses by hand-tuned Yukawa couplings to the Higgs field.

Additionally to the electromagnetic and weak forces, quarks and gluons are subject to the strong force, which is described by Quantum Chromodynamics (QCD). QCD is based on the gauge symmetry $SU(3)$ of colour, a generalized threefold charge which is source of the strong interaction, mediated by eight massless vector bosons (gluons, g). Its running coupling constant – in contrast to the electroweak ones – is decreasing with higher interaction energy. The consequences are "colour confinement" (naked colour is bound within hadronic sizes); and "asymptotic freedom", allowing perturbative calculations at high energies. The nuclear strong force that binds the protons and neutrons in atomic nuclei is a residual effect of the strong interaction.

The fundamental fermions exist in three generations of lepton and quark doublets:

generation:	1^{st}	2^{nd}	3^{rd}	charge	colour
leptons:	$\begin{pmatrix}\nu_e\\e^-\end{pmatrix}$	$\begin{pmatrix}\nu_\mu\\\mu^-\end{pmatrix}$	$\begin{pmatrix}\nu_\tau\\\tau^-\end{pmatrix}$	0 -1	none none
quarks:	$\begin{pmatrix}u\\d\end{pmatrix}$	$\begin{pmatrix}c\\s\end{pmatrix}$	$\begin{pmatrix}t\\b\end{pmatrix}$	$+2/3$ $-1/3$	$R,\ G,\ B$ $R,\ G,\ B$

The corresponding antiparticles are implicitly understood to exist as well.

Neutrinos (ν) interact only weakly, charged leptons (e^-, μ^-, τ^-) also electromagnetically. Quarks have a colour degree of freedom (R, G or B), thus interacting also strongly; confinement forbids quarks to act as free particles – observable are only colour-neutral bound states of e.g. 3 quarks (baryons) or quark-antiquark (mesons).

The observed mass eigenstates do not coincide with the doublet's flavour eigenstates, but are transformed by a unitary 3×3 mixing matrix (described by 4 non-trivial parameters: 3 rotation angles and 1 complex phase), separately in the quark and lepton sectors[2]. This allows for weak decays across generations, neutrino oscillations, and violation of CP conservation.

1.2.1. Physics beyond the Standard Model

Notwithstanding its great success, the SM (including non-zero neutrino masses) is far from being a satisfactory final theory. It has 26 free parameters (coupling constants; vector boson, Higgs boson and fermion masses; mixing angles and phases) which can only be determined experimentally. The nature of the neutrinos (Dirac or Majorana

[2] called the Cabibbo-Kobayashi-Maskawa (CKM) and Pontecorvo-Mako-Nakagawa-Sakata (PMNS) mixing matrix, respectively.

fermions?) and the origin of their tiny masses w.r.t. the other fermion masses cannot be explained. The observed Higgs boson mass requires fine-tuning of huge quantum corrections canceling each other (electroweak "hierarchy problem").

Theories beyond the SM address the problems above; they also suggest ways how to unify electroweak and strong interactions ("Grand Unification").

Supersymmetry (SUSY) is the common name for theories predicting for each SM fermion (boson) the existence of a partner boson (fermion), in order to solve the hierarchy problem in a natural way. None of the superpartners have been found so far, supposedly because they are too heavy. Thus SUSY is a broken symmetry, the breaking mechanism being explained by competing models. The "Minimal Supersymmetric Standard Model" (MSSM) predicts the existence of 5 Higgs bosons. The lightest SUSY particle, if stable as most models predict, is a candidate for cosmological dark matter.

1.3. Experiments in high energy particle physics

In modern physics research, high energy particle physics hits at one of the most elaborate frontiers. It is not the amount of energy itself that is of importance: e.g. at the most powerful collider to date, the LHC [11], an individual collision is comparable to the motion energy of a mosquito [37]. Applied to the size of subatomic particles, however, this energy is indeed very high, and the energy densities required can only be accomplished by large particle accelerators and colliders.

The collision energies at LHC, ILC or CLIC are of the TeV scale and open up a window to explore the physics of the standard model (including the discovery of its last pieces missing), and also to survey the realms beyond it. Those experiments may answer the questions for the origin of mass (the Higgs mechanism, or some other electroweak symmetry breaking ?), the existence of cosmological dark matter, the matter-antimatter asymmetry in the universe, or the puzzle of extra space-time dimensions. In essence, they explore the very fundaments of nature.

Thus, in order to study particles such as the Higgs boson (at ≈ 125 GeV)[3], the top quark (173.2 ± 0.9 GeV [18]), or other high energy phenomena, one first must create collisions providing enough center-of-mass energy for the underlying reactions.

This is accomplished in colliders, where charged particle beams get accelerated to high energies and collide with each other in a small interaction spot. In order to gain knowledge about the elementary processes one needs to reconstruct what happened in these events. Therefore a particle collider usually has one or more large and complex detectors, built around the interaction spot.

The data acquired from the individual sub-detectors are further processed by a chain of event reconstruction algorithms. This includes track reconstruction, which this thesis is about. Once an event is reconstructed, the processed data are saved

[3] An exact verification that the particle recently discovered at LHC [13, 1] is indeed the Higgs boson will need a lot of further investigations, e.g. by measuring its spin and coupling strengths.

and will be used for physics analyses: e.g. to find new particles, to determine the properties of known ones, or to verify or falsify the predictions of theoretical models.

1.4. Particle accelerators

Particle accelerators and colliders are the largest, but also the most useful tools for high energy physics. There are sources of more energetic particles in cosmic rays, but only accelerators give the possibility to study high energy events in a clean environment and at high luminosities.

Besides the center-of-mass energy, the most important characteristics of a collider is its luminosity $\mathcal{L}$, i.e. the proportional factor between the average collision rate $\mathrm{d}\bar{N}/\mathrm{d}t$ and the total cross section σ [48, p.49],

$$\frac{\mathrm{d}\bar{N}}{\mathrm{d}t} = \mathcal{L} \cdot \sigma, \qquad \bar{N} = \sigma \cdot \int \mathcal{L}\,\mathrm{d}t$$

with dimension $[\mathrm{cm}^{-2}\,\mathrm{s}^{-1}]$. Integrating over time and multiplying with a particular production cross section gives the number of events expected for a reaction. The higher the luminosity, the more events can be gathered; this is especially important for low cross sections, like the production of the Higgs boson.

There are different types of particle accelerators which can be distinguished by how they work and what they accelerate. The particles used for the beams must be stable[4]: protons, electrons and their antiparticles are the most commonly used ones. Examples are $p\,\bar{p}$ (Tevatron in USA), $p\,p$ (LHC at CERN), e^-e^+ (LEP at CERN, KEKB in Japan, ILC and CLIC), e^-p (HERA at DESY), and e^-e^-, $e^-\gamma$, $\gamma\,\gamma$ (ILC options).[5] Concerning the architecture of colliders, the two main types nowadays are synchrotrons acting as storage rings, and linear colliders for one-shot collisions.

Synchrotrons (like LEP or LHC) are closed ring structures where particles get bent into a circle by magnetic fields variated with time. By synchronizing (thus the name) the strength of the field with the particles' momentum they stay in orbit at a fixed radius, and get accelerated by electric fields every time they pass through a section with radio frequency (RF) cavities. Note that e^-e^+ storage rings suffer prominently from synchrotron radiation, which must permanently be compensated for by the RF cavities, thus effectively limiting the maximal beam energies attainable.

The most prominent example for synchrotrons at the time being is the Large Hadron Collider (LHC). It will be mentioned throughout this thesis, because of the large overlap of the researched physics with the ILC.

In a linear collider (like ILC or CLIC) the particles are accelerated along a straight path, so there is no problem with synchrotron radiation. As there is only one passing of particles through an RF section, the accelerating field gradients in the RF cavities must be very high: ILC (with superconducting cavities) aims at 31.5 MV/m and more [16], and CLIC (with an innovative "driver beam" technology) up to 100 MV/m

[4] Although there exist first ideas for a $\mu^-\mu^+$ ring collider.

[5] The photons are created by Compton backscattering of laser radiation off an e^-.

[36]. In order to achieve high enough one-shot luminosities, the beams must be very intense, and must be focused onto a very small interaction spot.

The concept of RF cavities is based on the behavior of charged particles passing through alternating electric fields. If the variation in the electric field is timed right, the particles are accelerated. Earlier this was achieved by shooting the particles through charged plates, which switched the sign of electric potential periodically, granting that the particles were always repelled by the plate they just passed and attracted to the next, resulting in acceleration. The speed of particles in high energy physics is extremely high, at relativistic energies it comes very close to the speed of light, and the frequencies are in the range of radio waves. For this reason nowadays RF cavities are used. They act as microwave resonators: the metal case of a fixed size results in standing waves with low energy loss. These standing waves are the high frequency equivalents to the plates with switching electric potential.

1.4.1. The physics case for electron-positron colliders

At first sight, LHC as a hadron collider covers a broad mass range: up to 14 TeV (at present 8 TeV) center-of-mass energy means a discovery potential way above 1 TeV. However, the beam energy of each proton is distributed statistically over its 3 valence quarks, the gluon fields, and virtual quarks-antiquarks participating in the collisions. For the fundamental reactions, on average only about $1/6$ of the nominal center-of-mass energy is available, and the actual energy participating is unknown. Moreover, the quarks and gluons also create a large background of soft QCD processes. This requires a powerful and sophisticated triggering of the data acquisition, based on simulation, with the risk of missing something important but not anticipated.

These drawbacks are not present in a lepton collider [60]. In every e^-e^+ collision, the full center-of-mass energy of the two beam particles commits to the fundamental reaction, and the sum of the energies of all created particles is known to be equal to that.[6] There is no additional background from soft QCD processes, and the environment is very clean.

1.4.1.1. Polarization

Electron-positron colliders offer the additional possibility to polarize their beams [44, 46, 45] (i.e. provide e^- and/or e^+ with a large fraction of only positive or negative helicity). At ILC, the e^- beam can be polarized up to 90%; the simultaneous polarization of the e^+ beam as well would further enhance the benefits and is therefore an option seriously discussed [45]. The importance of polarization stems from the fact that the helicities of the beam particles affect the cross section of different reactions.

Consider, e.g., the annihilation $e^-e^+ \to \gamma \,/\, Z^0$ [46, p.2]. In the limit of vanishing electron mass, angular momentum conservation (γ and Z^0 are vector bosons) dic-

[6] except in the presence of e^- or e^+ beamstrahlung, carrying away part of the energy.

tates that this process can only happen if e^- and e^+ have equal spins, i.e. opposite helicities (because of their momenta being opposite).

This can be used in many ways. We give three examples: in Higgs production, polarizing the beams can suppress some background processes by a factor of about two.

Polarized beams also allow to reconstruct the polarization of the particles created, and therefore enable measuring their CP violating and CP conserving couplings with high precision.

Also for theories beyond the Standard Model, for example SUSY, constraining the reactions can limit backgrounds, making polarization a valuable tool. For example in the production of scalar muons (hypothetical SUSY particles) $W^{\pm}$-bosons are a background to the desired events, but can be suppressed to a high level with 90% electron polarization.

1.4.2. Lepton colliders

The advantages listed in 1.4.1 are strong arguments for a TeV scale lepton collider, running in parallel with the LHC, in order to perform complementary precision studies of the standard model and beyond. Such a collider, with electron and positron beams, would be built as a linear collider, in order to avoid the excessive energy loss caused by synchrotron radiation in a storage ring.

1.4.2.1. Synchrotron radiation

A particle of mass m and elementary charge, circulating at energy E in an orbit of radius R, will constantly lose energy via electromagnetic radiation because of its centripetal acceleration. For one revolution, this amounts to [48, p.363]

$$-\Delta E = \frac{4\pi\alpha\hbar c}{3R}\beta^3\gamma^4, \qquad \gamma \equiv \frac{1}{\sqrt{1-\beta^2}} = \frac{E}{mc^2} \tag{1.1}$$

with Sommerfeld's constant $\alpha \approx 1/137$, $\hbar c \approx 0.2 \times 10^{-15}$ GeVm, and $\beta \equiv v/c \approx 1$ at relativistic energies. It gets impossible to further accelerate particles if the energy gained in the RF cavities is equal to that lost by synchrotron radiation. The energy loss per time is given, with the circulation frequency ν_{circ}, by

$$\frac{\mathrm{d}\Delta E}{\mathrm{d}t} = \Delta E \cdot \nu_{circ} \propto \frac{1}{R^2}\left(\frac{E}{m}\right)^4, \qquad \nu_{circ} = \frac{\beta c}{2\pi R} \tag{1.2}$$

The total power required for keeping the beams orbiting at constant energy is given by multiplying $\mathrm{d}\Delta E/\mathrm{d}t$ with the number of particles stored in both beams, divided by the operating efficiency of the RF cavities (which can be increased by optimizing their geometric layout and by the use of superconducting material).

The principal design parameters of a circular collider are its radius[7] R, the mass m of the beam particles, and the maximum energy E to be attained. For energies in the TeV range, let's discuss radius and beam particle mass:

The largest circular collider so far was LEP [47], with electron and positron beams up to 104.5 GeV orbiting in a ring of 26.7 km circumference (housed in the very same tunnel where now LHC is); its power consumption was roughly up to 30 MW. A similar collider for 500 GeV beams would need a circumference of more than 600 km [12]. Since the radius is a principal cost driver in the construction, this makes a circular electron-positron collider unfeasible for energies significantly higher than at LEP.

Synchrotron radiation decreases by the fourth power of the mass. If one wants to keep the advantage of colliding point-like leptons, an alternative would be to replace e^-e^+ by $\mu^-\mu^+$. Since muons are ≈ 207 times heavier [10], the synchrotron radiation would be more than 9 orders of magnitude less at same radius and energy. First ideas exist for such a muon collider [14], but a lot of R&D will be needed in order to overcome the main obstacle, the muon's instability: its mean lifetime of 2.2 μs, albeit dilated by the Lorentz factor γ in the lab frame, poses still a very big challenge.

LHC has the same circumference as LEP, but is colliding protons (1836 times heavier than electrons). Its synchrotron radiation is negligible, even at beam energies of 7 TeV. However, it has the disadvantages of a hadron collider mentioned in 1.4.1.

1.4.2.2. Linear colliders

At present, the only feasible choice for colliding leptons at the TeV scale is a linear electron-positron collider, composed of two opposite accelerators for e^- and e^+, thus completely avoiding energy loss by synchrotron radiation.

The price to pay is missing one advantage of synchrotron storage rings: there, the orbiting particles pass the RF cavities with every revolution and gain their energy step by step. In a linear accelerator the particles are shot only once from the start to the interaction point, thus all the energy has to be gained along one straight path. In order to attain a center-of-mass energy $\sqrt{s}$ in the TeV range, a linear collider with current technology would necessarily be several 10 km long.

Two projects exist for a TeV scale linear electron-positron collider. The International Linear Collider (ILC) [16] is based on mature technology of superconducting RF cavities to achieve $\sqrt{s}$ up to 1 TeV. The Compact Linear Collider (CLIC) [36], on the other hand, is based on an innovative but less advanced technology with RF cavities powered by a "driver beam", to achieve $\sqrt{s}$ up to 3 TeV.

Both projects are backed by the international community, with cooperation in fields granting synergy. Which one might eventually be realized depends on relevant results from LHC, hinting on physics beyond the standard model either below or above 1 TeV, thus favoring ILC or CLIC, respectively. However, most of the detector

[7] in reality the orbit is not a perfect circle, so an effective $R = \mathit{circumference}/2\pi$ is taken.

concepts and reconstruction software is useful for both.

1.4.3. The International Linear Collider (ILC)

ILC [16] is a linear electron-positron collider with superconducting RF cavities, operating at 1.3 GHz and attaining field gradients of 31.5 MV/m[8]; higher field gradients well above 40 MV/m are expected to be achievable for the second stage.

The first stage aims at a center-of-mass energy adjustable for precision scans in the range $\sqrt{s} = 0.2 \ldots 0.5\,\text{TeV}$, allowing for a Z^0, Higgs and $t\bar{t}$ factory. The second stage will upgrade to $\sqrt{s} = 1\,\text{TeV}$; the total length of ILC is 31 km.

The accelerated particles are grouped into so-called bunches, particles grouped together moving parallel through the RF cavities. The bunches themselves are grouped as well to so-called bunch trains wherein the single bunches are separated by a few hundred nanoseconds. A whole bunchtrain has the length of 1 ms, so it contains thousands of bunches. Bunch trains are emitted in a frequency of 5 Hz.

Design luminosity is $\mathcal{L} = 2 \cdot 10^{34}\,\text{cm}^{-2}\,\text{s}^{-1}$. Electron polarization ≥ 80 % and optional positron polarization are further assets.

Another option is using ILC for e^-e^-, $e^-\gamma$ and $\gamma\gamma$ collisions (with γ created by Compton backscattering of laser radiation off an e^-).

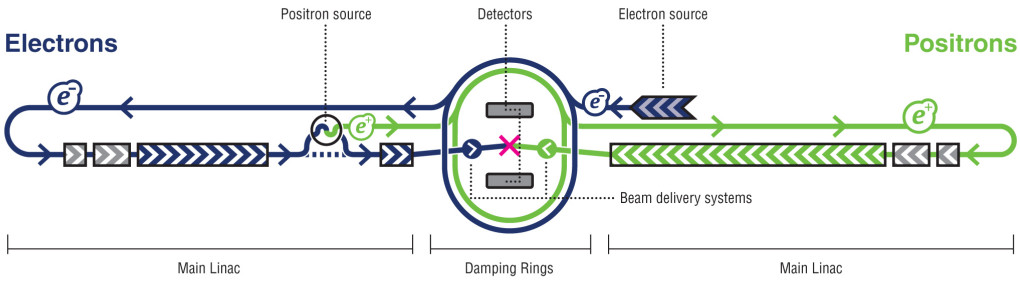


Figure 1.1.: A schematic layout of the International Linear Collider

Two detector concepts have been validated to operate at ILC: the International Large Detector (ILD) with a TPC + Si tracker; and the Silicon Detector (SiD) with an all-Si tracker. Both concepts make use of particle flow calorimetry.

In order to save costs, only one beam interaction zone will be available. Since only one detector at a time can reside there and take data, a push-pull system must be used: the detectors are attached to a rail system and can be moved in and out for being replaced with each other. Afterwards a careful re-alignment of the detector new in place has to be done; thus the push-pull system is not without a cost.

[8] the same as used for the free electron X-ray laser XFEL at DESY.

1.4.4. The Compact Linear Collider (CLIC)

CLIC [36] is an alternative linear electron-positron collider with normal RF cavities, operating at 12 GHz and attaining very high field gradients of 100 MV/m; this is achieved by power transfer from a parallel high-current "driver beam".

Center-of-mass energies are $\sqrt{s}$ = 1, 2 and 3 TeV, to be achieved in 3 stages; thanks to the high field gradients, the total length is only 48.3 km.

The design luminosity at 3 TeV is $\mathcal{L} = 6 \cdot 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$.

CLIC has many common points with ILC, but different parameters, among them the beam bunch structure. The higher energy causes much more beam-induced background of e^-e^+ pairs.

There will also be a push-pull system for two detectors. The two CLIC detector concepts are variants of those for ILC, called CLIC_ILD and CLIC_SiD [36, Volume 3]. Cooperation exists with the ILC detector hardware and software groups.

1.5. Particle detectors

At the heart of every high energy physics experiment are the particle detectors, which measure the trajectories, the energies and other characteristics of the particles emerging from a collision event. It usually consists, from inside outwards, of

- **Tracking system:**

  Detectors that are able to measure the trajectories of charged particles by their impacts on position-sensitive layers, consisting of some solid (mostly silicon) or gaseous material which gets ionized by the passage of a charged particle. Several layers have to be used in order for a pattern recognition to be able to match the corresponding impact points and find the common tracks that caused them.

  A special part of a tracker is the **Vertex Detector**, consisting of high-resolution layers close around the beam tube, for a precise position measurement.

  The tracks found are reconstructed by statistically estimating their parameters (position, direction, momentum, charge), based on an appropriate track model (helix in case of a homogeneous magnetic field) and taking care of material effects (e.g. multiple Coulomb scattering off nuclei). Some detectors like time projection chambers (TPC) also allow a good measurement of the energy loss $^{dE}/_{dx}$, which gives additional input to the identification of the type of the particles.

  The reconstructed tracks may further be combined to common vertices, and subjected to kinematic fitting (applying energy-momentum constraints).

  The momentum resolution of a tracker is very much dependent on its lay-out. For the transversal component w.r.t. a magnetic field, it is usually

parametrized as $\Delta p_T/p_T = \sqrt{(a \cdot p_T)^2 + b^2}$. Analytic formulas exist for calculating simple cases [27, 50], but in general one has to rely on a Monte Carlo simulation. A fast simulation tool like LiCtoy [51] is useful for an early optimization.

- **Calorimeters:**

  Detectors which measure the overall energy of particles by stopping them completely, i.e. forcing them to release all their energy. It consists of cells of crude or fine granularity, allowing some position measurement. Also neutral particles, which are invisible to the tracking system, get observed. Two types of calorimeters are usually used in high energy physics: electromagnetic and hadronic.

  In an electromagnetic calorimeter (ECAL), electrons, positrons and photons are stopped. In the dense material develops a shower of particles: e.g. electrons undergo bremsstrahlung and lose energy by emitting photons; these photons can create electron-positron pairs, and so on. The particles of the shower can then be detected, e.g. with a scintillation detector.

  A hadronic calorimeter (HCAL) is usually placed behind the electromagnetic one. It works with the same concept as the electromagnetic calorimeter, but the particles will mainly interact via the strong interaction.

  The energy resolution of a calorimeter is parametrized as $\Delta E/E = x/\sqrt{E}$. It is worse for hadronic calorimeters than for electromagnetic ones, and both are worse than the resolution of a well-designed tracker for momenta up to several 10 GeV. A modern approach for increasing the energy resolution is particle flow analysis (PFA) [57], which requires very fine-grained calorimeters in order to efficiently match with all charged tracks; for those only the tracker measurements are used, i.e. the calorimeters measure only the unmatched neutral tracks.

- **Solenoid magnet:**

  To produce a quasi homogeneous magnetic field parallel to the beam tube. In such a field charged particles move on a helix trajectory which allows an easy track model for the reconstruction. It is also important for a TPC.

- **Muon detection system:**

  Muons of high energies are able to traverse through all the before mentioned detectors. Adding a muon detection system on the outside of the detector allows to measure their trajectories, helps with their identification, and sometimes also catches the tails of particle showers created in the calorimeters.

Two detectors following this typical layout are the Silicon Detector (SiD) and the International Large Detector (ILD), that will both operate at the ILC.

1.5.1. The Silicon Detector (SiD) at ILC

The main difference of SiD [6] w.r.t. ILD (see 1.5.2) is its tracking system which is based only on silicon detectors, and its strong solenoid field of 5 T.

1.5.2. The International Large Detector (ILD) at ILC

The ILD detector [28] has, in contrast to SiD (see 1.5.1), a heterogeneous tracking system, consisting of a large time projection chamber (TPC) together with a silicon tracker. Its solenoid field is 3.5 T, upgradeable to 4 T when running at $\sqrt{s} = 1$ TeV.

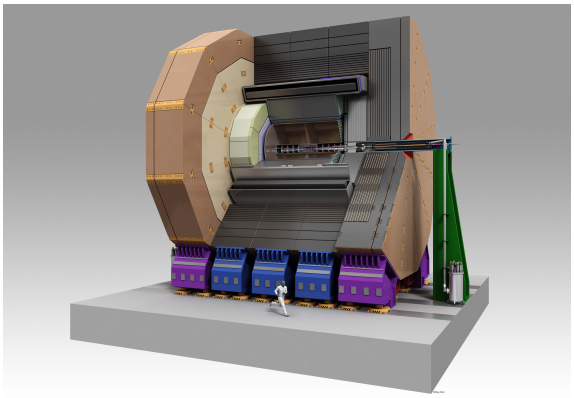


Figure 1.2.: Schematic view of ILD

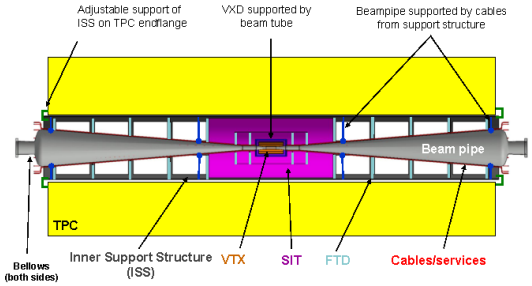


Figure 1.3.: Inner tracking part of ILD

The ILD detector aims at transverse momentum and jet energy resolutions (see 1.5) of $a < 2 \cdot 10^{-5}/\,\mathrm{GeV}$, $b < 10^{-3}$ (about 1 order of magnitude better than for LHC detectors), and $x < 30\%\,\sqrt{\mathrm{GeV}}$ [28]. This can be be achieved by an excellent tracking system (see below), and a calorimetry system based on PFA.

1.5.2.1. The tracking system of ILD

Tracking in ILD is accomplished by a combination of a central large time projection chamber (TPC) and several silicon detectors, the inner ones as seen in Figure 1.3.

The main track reconstruction is accomplished by the TPC, which consists of about 200 pad-rows, giving its main asset: a lot of space point hits per track. The TPC covers the whole barrel region, and therefore can measure most tracks with polar angles in the range $11.5° < \theta < 168.5°$ (at least 10 pad-rows hits). However, because of its relatively slow readout, the TPC is not able to deliver absolute longitudinal (z) values for the measured space points; therefore it is necessary to extrapolate and match the tracks reconstructed in the TPC to hits of fast detectors inside and outside the TPC.

For this task the Silicon Inner Tracker (SIT) and the Silicon External Tracker (SET) are used. Both are “false” double-sided silicon strip detectors consisting of

2 layers and 1 layer, respectively. Thus each track traversing SIT, TPC and SET gives 3 additional space points which the reconstructed TPC track can be matched with.

Further inside, close around the beam tube, is the Vertex Detector (VTX). It is a silicon pixel detector for high precision measurements close to the interaction point. This contributes to the high impact parameter resolution necessary for the reconstruction of the decay vertices of short-lived particles, like b hadrons and the τ lepton.

The VTX–SIT–TPC–SET cover the barrel region as defined above. However, tracks with a polar angle θ closer to the beam axis z should be detected and reconstructed as well. This is accomplished by the Forward Tracking Detector (FTD), consisting of 2×7 layers of silicon disks placed perpendicular to the z axis, thus covering the so-called forward region between the conical shaped beam tube[9] and the inner wall of the TPC. Since this thesis is about track reconstruction in this region, a bit more information about FTD is provided in the next section below.

1.5.2.2. The Forward Tracking Detector (FTD) of ILD

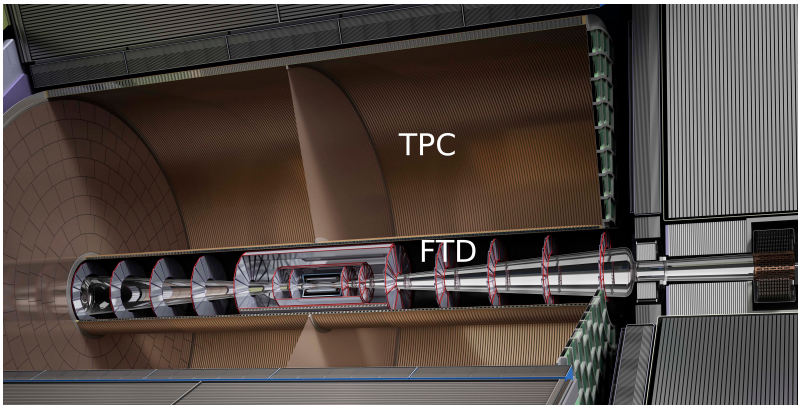


Figure 1.4.: Cut view of the inner tracking part of ILD

Figure 1.4 shows the Forward Tracking Detector. It consists of 2 halves, each with 7 disks in the forward or backward direction, respectively. The two inner disks are silicon pixel detectors, and the 5 outer ones are "false" double-sided silicon strip detectors. These disks are constructed of so-called petals of trapezoidal shape. The Si sensors on the petals need frames. In a simple approach, there would be non-sensitive material between them where tracks could escape detection. In order to avoid this, the petals are made bigger, overlapping a bit, while staggered forth and back. This also leads to a small overlap of sensitive parts, which means additional space points for tracks passing through these areas of the sensors.

The exact characteristics are still an open issue, as the technologies are advancing fast, and some better solution might be available once the construction of ILD is

[9] the beam-induced background of e^-e^+ pairs is produced mainly in the forward directions.

about to start. This is especially true for the pixel detectors.

For the strip detectors each petal will most likely have two sensors on the front and two on the back, oriented with a stereo angle between them, such that the combination of the strips gives space points (details see below). In order to suppress artificial combinations (so-called ghost hits), the stereo angle of the strips will be chosen shallow (e.g. at 5°), thus most ghost hits being pushed out of the sensitive areas.

For a stand-alone track reconstruction the FTD is able to cover a region from the beam tube at $\theta \simeq 5°$ up to $\theta \simeq 23°$, i.e. covering the ranges $5° < \theta < 23°$ and $157° < \theta < 175°$. This is the region where quasi straight tracks can cause 4 hits or more.

At $\theta > 23°$ or $\theta < 157°$ some FTD disks are still hit, but only 2 or 3 hits are not enough for a decent stand-alone track reconstruction. Yet these single FTD hits can be used in combination with the TPC (tracks in that region are well reconstructed there), by matching and adding them to an extrapolated TPC track. Such an update is actually performed by the `FullLDCTracking` processor.

1.5.2.3. Pixel and strip detectors

Silicon detectors [9] are essentially doted silicon plus an electric potential together with read-out electronics. A simple version could consist of the main sensitive area consisting of n-doted silicon[10]; a positive potential is applied on the side opposite of the read-out electronics; and the read-out is connected to p-doted silicon. So when a charged particle traverses the bulk of the n-doted silicon it causes electron-hole pairs. Due to the electric potential the electrons will travel away from, and the holes will travel towards, the p-doted areas connected to the read-out. There they cause an electric current, which is collected by the electronics.

Strip detectors

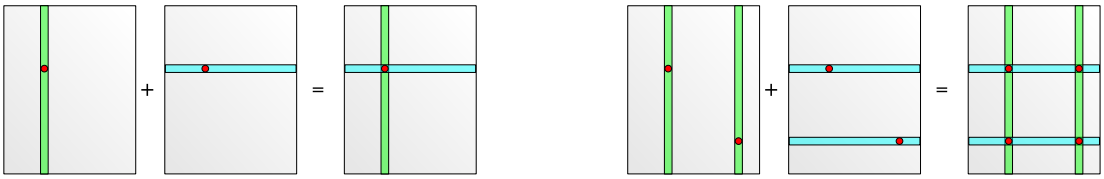
These are a special version of this concept, where the p-doted silicon areas are in the form of stripes. The read-out is then simply connected to the ends of the stripes. Such a detector can be used for 1-dimensional measurements: Imagine a strip sensor in the x-y plane, perpendicular to the z axis at a known z coordinate, with the strips pointing into the x direction. When a hit occurred, the measurement gives a y coordinate (with a value usually achieved by looking for the center of the charge distribution collected from adjacent strips). An x coordinate cannot be determined (other than its value being somewhere between the beginning and the end of the strips). The measurement results in exactly one equation relating y and z, providing one dimension of measurement.

Complete space points (2-dimensional measurements) can be achieved by a double-sided strip detector: mounting at the bottom of a normal strip sensor a second set of stripes perpendicular to the first ones, i.e. pointing into the y direction. Because this has some technical difficulties, often a "false" double-sided strip detector

[10]The vice-versa kind of detector (p-doted bulk material with n-doted electrodes) works as well; but for historical and technical reasons is far less common.

is used, i.e. two single-sided strip sensors just attached back to back. In both cases, the bottom side provides an additional measurement of the x coordinate. Thus one has measured values for x and y, together with a known value for z, i.e. a space point, as wanted for track reconstruction.

There is however a downside to this technology: artificial ghost hits. See Figure 1.5: whereas a single hit on the sensor causes the creation of one corresponding space point, 2 hits already cause an additional 2 ghost hits, i.e. space points which are created from the occupied strips but have no real meaning. In such a setup, for n true hits, n^2 space points are created, where $n \cdot (n-1)$ are ghosts. In case of high occupancies this means a drastic combinatorial explosion of space points, making track reconstruction a much harder task.



(a) 1 hit (b) 2 hits

Figure 1.5.: 1 and 2 hits on a “false” double-sided silicon sensor

This can be avoided by using stripes which are not perpendicular, but at an angle $\alpha < 90°$ oriented with each other. Such a shallow stereo angle will push most of the ghost hits over the physical boarder of the sensor, thus giving an easy method to identify them, since real hits must necessarily lie within (see Figure 1.6). However, one can only reduce the number of ghost hits, not get entirely rid of them.

Note that a shallow stereo angle also deteriorates the spatial resolution in one direction, while improving it in the perpendicular one.

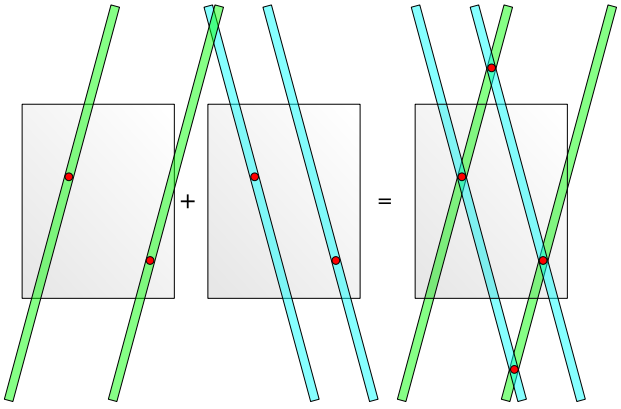


Figure 1.6.: 2 hits on a ”false” double-sided sensor with a small stereo angle, ghost hits are pushed off the sensor

Pixel detectors

These are a detector type completely avoiding the ghost hit problem. Instead of using stripes where only the pitch width is contributing to the precision measurement, a pixel detector uses square or rectangular electrodes, thus measuring directly two coordinates (x and y); together with the known z, this yields a space point.

The read-out, however, which is rather easy for strip sensors (simply attached at the ends of the stripes), becomes considerably difficult for pixel sensors. The question is, how to connect the pixels with the read-out electronics?

There are a few possible solutions, each with some drawbacks:

- Placing the read-out directly on top of the pixels will increase the material budget, leading to more multiple Coulomb scattering and worse track resolution;
- Integrating the read-out into the sensor itself will require more space needed for the pixels, leading dirctly to worse spatial resolution of the sensor;
- "Clocking through" the signal to the end of the sensor will need a lot more time for the read-out, leading to event pile-up in case of high luminosities.

For the FTD pixel disks, currently the clocking through method is envisioned. The long read-out time frame means integration over the signals from multiple bunch crossings, causing a pile-up of hits from many events (in the order of 100).

In conclusion, both pixel and strip technology have their advantages and disadvantages. FTD will use a combination of both. The two inner disks will be pixels the outer five "false" double-sided strip detectors. Since the artificial hits from the 5 outer disks (ghost hits) and the additional hits from the inner disks (hits integrated from other events) are uncorrelated and won't match except by chance, wrong tracks can be easier discarded by this strategy of using both technologies.

1.5.3. The ILD software framework

When building the software framework for an experiment, it is crucial to make sure that the simulation of data needed to test and optimize the detector, and the reconstruction software to analyze these data, are well separated. Since ILD [28] is still in an R&D phase, all data must be generated by Monte Carlo simulation.

1.5.3.1. The chain of simulation and analysis

The stages from simulation to analysis can roughly be described as follows [24]:

1. **Event generation**
   Simulates the underlying physics of standard model and/or beyond. Events are generated by a package of choice, from simple to sophisticated ones such as Whizard [31]. Output are the kinematic parameters of the created particles,

and possibly also short-lived decays within the beam tube (mainly c and b hadrons and τ leptons). A common de-facto standard output format is `stdhep` [26].

2. **Detector simulation**
   Simulates as realistic as possible the passage of the particles through the detector, including the decays of long-lived particles (K_s^o and Λ^o). It simulates material effects like multiple Coulomb scattering and energy loss, and also creates the hits in track-sensitive detector layers. For this it relies on a detailed detector description. ILD's simulation framework is `Mokka` (for practical reasons, simulation of the digitized responses to the hits is relegated to the next step).

3. **Event reconstruction**
   This is the point where real data will eventually start to be processed. Reconstruction of an event includes track reconstruction (i.e. track search and fit), vertex reconstruction, and possibly kinematic fitting. Part of the detector description is needed as well. ILD's reconstruction framework is `Marlin`.

4. **Physics analysis**
   From here the events are investigated as a sample, in order to extract the relevant information for studying the physics reactions of interest.

1.5.3.2. The simulation package Mokka and GEANT4

`Mokka` is a framework to simulate the passage of particles through a realistic virtual version of the ILD [28, p.7]. It is based on the general-purpose toolkit GEANT4 [5], and written in C++. The different detectors can be individually programmed as separate "drivers", giving the possibility to compare different set-ups of the ILD. The details needed to describe the single detectors are stored in a MySQL database, thus are easily accessible and changeable. Provided with a `stdhep` input file created by some event generator, it tracks the particles through the whole detector, realistically simulating diverse physics effects like radiation, scattering, decay etc. It also creates the hits in the track-sensitive detectors, and saves the results in LCIO format [23].

1.5.3.3. The detector geometry description toolkit Gear

Another `Mokka` output, besides the `LCIO` file, is the detector geometry description file `Gear` (Geometry API for Reconstruction) [25]. This is a file with geometry information about the detector, essentially an extract of the MySQL file, containing only the information that is needed for reconstruction. It is an `xml` file, written in a human readable form, giving the possibility for looking at the most important detector parameters, or to change them by hand for debugging.

1.5.3.4. The reconstruction framework Marlin

`Marlin` (Modular Analysis and Reconstruction for a Linear Collider) [28, p.7] [24] is the modular framework where the reconstruction of events of the ILD is performed. It is written in C++ and follows a modular design: the different tasks such as digitization, track and vertex reconstruction, calorimetry and particle flow, etc. are each separated into different modules, called "processors". `Marlin` is run with a steering file in human readable `xml` format, which tells `Marlin` when and how to run each processor. This approach allows easy change of processors and/or parameters, making it possible to compare different algorithms and to fine-tune the parameters. It also makes adding new software considerably easier, as one can work within a processor without too much interfering with others. All data passed between processors is via `LCIO` objects stored in the event. The final results are normally also stored in `LCIO` format.

1.5.3.5. The persistency framework LCIO

`LCIO` [23] is a data model covering all the classes that are needed to describe an event, like hits, tracks, vertices, clusters etc. It is used from simulation to analysis, thus guaranteeing consistent persistency without information loss. `LCIO` is also an external file format, used by `Mokka` to save the fully simulated events, and by `Marlin` to save the reconstructed events for a subsequent physics analysis.

2. Algorithms of event reconstruction

The events created in high energy physics result in measurable signals in different detectors. Event reconstruction deals with finding the causes of these signals, i.e. with gaining as much knowledge about the underlying events as possible. In the chain of simulation and analysis (see Subsection 1.5.3) this is the last step before the physics analysis can start. The ultimate goal of reconstruction is finding the particles, e.g. their identity, path, momentum, energy and production vertex. The reconstruction can be separated into distinct tasks, each dealing with different detector hardwares and reconstruction algorithms [21].

2.1. Areas in event reconstruction

In order to reconstruct an entire event, information from different areas of reconstruction have to be combined. These areas include: track and vertex reconstruction, muon tracking (tracking done in the muon detectors), calorimetry (determination of particle energy and maybe some tracking), particle jet reconstruction, time-of-flight measurements and particle identification. Here track reconstruction is discussed and then a short introduction to vertexing and particle identification is given, as both have some overlap with tracking and can use its information.

2.1.1. Track reconstruction

Track reconstruction is the task of finding the trajectories of charged particles in the detector and to estimate the parameters of these tracks. A good introduction to the topic is given in [35] (about the hardware) and in [55] (about the algorithms). Finding the path of a particle through the different sub-detectors is important for two reasons. First, it enables the calculation of the kinematic parameters of the particle at the point where it was created, the vertex, helping to reconstruct the underlying physics of the event.

Second, the trajectories can be used to link different sub-detectors, which is a principle typical for most modern detectors. In the ILD detector (see Figure 2.1) for example the Time Projection Chamber (TPC) is responsible for the central tracking. To the inside the tracks from the TPC can be linked to the vertex detector, giving precise measurements close to the vertex. Also the tracks can then be linked to the calorimeters on the outside, which measure the energy of the particle. There the

information from the reconstructed tracks can help to improve the reconstruction in the calorimeters. The approach of using information from reconstructed tracks for the calorimetry is called "particle flow" [57] and is a fixed part in the event reconstruction for the ILD detector. Its main benefit is a better jet energy resolution.

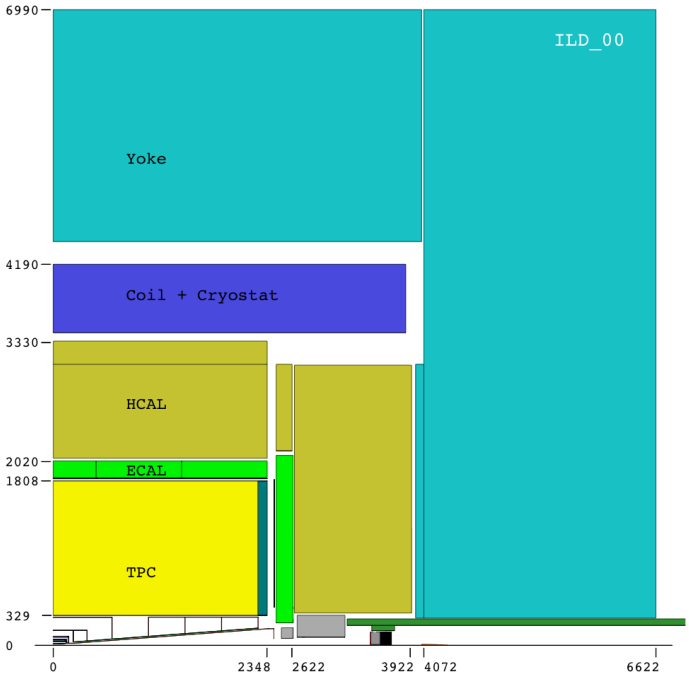


Figure 2.1.: The TPC is surrounded by other detectors on the in- and outside: tracking is needed to link the signals in those detectors.

The tracks can most easily be reconstructed and extrapolated if they are not disturbed by material. For this reason tracking detectors are usually built with a material budget as low as possible[1] [35]. If there were no material effects and the magnetic field would be perfectly homogeneous, the charged particles would move on helical trajectories with an axis parallel to the z-axis [7]. Such a helix can be described by 5 parameters for a given reference point. For the ILD detector (more specific `LCIO`) the helix is described by the parameters Ω, ϕ_0, d_0, z_0 and $\tan\lambda$. d_0 for example is the distance of the helix to the reference point in the x-y plane. For details and definitions see [34].

The interaction with the detector material causes the helices to change along the track, making it necessary for reconstruction to take into account that the helix parameters can change. The goal is to reconstruct the path and estimate the parameters along the whole track of the particle in the detector. This information can later be used for vertex or particle identity reconstruction.

Track reconstruction can be subdivided into separate steps. These steps vary however with the used algorithms; they can be further subdivided or intertwined. A

[1] An upside of the material effects is that they can be used to determine the identity of particles, as they differ in the energy loss they have when traversing through material.

relatively basic classification, as is used in this document, consists of three distinct steps:

- At first the the hits are combined to track candidates by **track finding**.
- **Track fitting** then calculates the parameters of the track candidates.
- Finally **ambiguity resolving** makes sure that the final set of tracks is consistent.

The same classifications can also be made for vertex reconstruction. The first two steps are also known as pattern recognition and parameter estimation in more general terms.

2.1.1.1. Track finding

Track finding means to group the hits from the sensors together that belong to the same true track. There are mainly two errors that can happen, when reconstructing a track (or any other general pattern): One can fail to find some or all tracks, thus losing important information. And one can reconstruct tracks that do not exist, i.e. are combinatorial background. These two errors are measured by the quantities called efficiency and ghost rate [41]:

$$efficiency = \frac{true\ tracks\ found}{all\ true\ tracks\ to\ be\ found} \tag{2.1}$$

$$ghost\ rate = \frac{false\ tracks\ found}{all\ found\ tracks} \tag{2.2}$$

Both are percentages ranging from 0 to 1. It is obvious that a low ghost rate (0) and a high efficiency (1) are desired. In order to calculate these values it is necessary to define what qualifies a pattern as "found" and what is considered a ghost. A track is said to be found if there is a reconstructed track that can be linked to it, which usually means that the majority of hits in the reconstructed track come from the true track. There can be, however, further constraints. For example a reconstructed track could contain all hits of a true track, but have additional contamination by signals not belonging to the original. The problem with contamination is that during the step of parameter estimation wrong hits can cause wrong track parameter results. Outliers, i.e. hits that either do not belong to the true track or are highly distorted by noise, must be dealt with carefully. So called robust methods used in fitting can do outlier removal to a certain degree, nonetheless track finding is desired to deliver tracks as pure as possible.

A track could also contain only hits from a single true track, but miss some of its true hits. As reconstructing the full tracks gives higher accuracy, complete tracks are ideal, but in general the track parameters can also be determined from shorter track segments.

Alternatively to determining "found" tracks by their purity and completeness, it is possible to use the parameters of the reconstructed tracks for mapping (see [41, p. 569]. If the parameters of a reconstructed track are close enough to those of a true track, the true track is considered to be found. Throughout this document the first method will be used, where reconstructed tracks are mapped to true tracks based on the hits.

It is also important to specify what is meant by "all true tracks to be found". As a measure of quality an efficiency of 1 should mean that everything one wanted to find got reconstructed. There are however tracks that are either not necessary to reconstruct (because they come from background signals or are uninteresting for the further analysis) or cannot be reconstructed. An example for latter ones are tracks with a highly irregular behavior, such as having strong kinks from multiple scattering. If there is only a small number of hits per track, kinks are nearly impossible to find[2].

It is important to always give the definitions of what tracks are to be found and when they count as found together with the actual numbers, when talking about efficiency or ghost rates.

There are more quality criteria for pattern recognition, for instance the clone rate and parameter resolution (also see [41]). A clone is a track reconstructed multiple times, i.e. when multiple reconstructed tracks can be associated with a single true track. As none of the compared algorithms in Chapter 5 allows for clones in the final collections, the clone rate is a minor concern here. For example, in the packages written by the author, the clones will get sorted out by the Hopfield neural network at the end. Parameter resolution is a term for the residuals of the reconstructed parameters $P_{true} - P_{reco}$ and is an important value, but is mainly dependent on the hardware attributes and not the pattern recognition algorithms.

Track finding algorithms can be subdivided into 3 different general approaches as in [17], where other sources only use the first two categories [41]:

- **Global:** Has its name from taking into account all hits simultaneously. The tracks can be described by a set of parameters (e.g. helix parameters). These span the parameter space, in which every track is a single point. As measurements (i.e. hits) constitute equations limiting the possible parameters of a corresponding track, they can be mapped into this space.

  The Hough transformation for circular tracks can serve as an example. This is of relevance, because the helices in a homogeneous magnetic field are circles in the projection onto the x-y plane. A circular tracks in two dimensions can be described by 3 parameters: the coordinates of its center (x_0, y_0) and the radius (R). The circle equation is:

$$R^2 = (x - x_0)^2 + (y - y_0)^2 \tag{2.3}$$

[2]For tracks with more hits, kinks can be identified by having two tracks with the same momentum and charge meeting at one place, which suggests that at this place the track underwent a high multiple scattering that changed the direction but not the magnitude of the momentum.

A hit on the circle gives a value for x and y. For every hit one gets an equation (2.3 with fixed x and y) with 3 unknowns (R, x_0, y_0). The equation is therefore under-determined by two degrees of freedom: there is an infinite number of circles that could go through one single hit. But the equation still serves as a restriction on the possible circles. In the parameter space (R, x_0, y_0) this equation is a surface: every point on the surface corresponds to one allowed circle through the hit. If multiple hits lie on the same track, the surfaces in parameter space will intersect in the point corresponding to the parameters of the circle. To find those intersection one usually histograms the parameter space, i.e. subdivides it into small boxes and counts the number of surfaces crossing through. The maxima, i.e. the boxes with local highest occupation numbers correspond to tracks.

The various global algorithms use different parameter spaces and different methods of finding these maxima. Examples are: template matching [41], fuzzy Radon transformation [41] and Hough transformation [41, 55].

- **Local:** These methods take hits into account one at a time and build the track candidates by adding hit after hit. They start from so called track seeds[3], short segments of tracks that allow to extrapolate the track with a track model. Via extrapolation to other measurement sites hits are picked up if they pass a certain quality criterion, i.e. fit to the track. The algorithms available differ in the various steps: in the quality criteria applied (distance, χ^2 increase), the track models, the extrapolation algorithms (Kalman filter, helix fit) and the seeding techniques (track road, track following).

- **Semi-Global:** These algorithms treat all hits at the same time, but are based on local seeds. The system then consists of interlinked track seeds, which all interact locally, but are processed at the same time. An example is the cellular automaton [17].

2.1.1.2. Track fitting

Once the track finding is finished and hits supposedly belonging to the different tracks are identified, the properties of the tracks need to be determined. This procedure is based on a track model, which defines the free parameters of a track. For a setup with a homogeneous magnetic field this for example is a helical track, which can be described by 5 parameters.

In the fit, parameters are searched that result in a track that best explains the measured hits. The hits as created by the digitizers are not a perfect representation of the true place of impact of the particles on the sensors because of noise in the hardware and the algorithms determining the place of the hit. This means that there

[3] Although sources differ about whether distant hits are seeds. For example [55] considers the starting hits in the track road algorithm not to be seeds, while [41] does not require close distance for track seeds.

usually is not a single parameter set where all hits are perfectly on the track, but rather one has to find the values of the parameters where the hits come as close as possible to the resulting fitted track. As mentioned in [55]: "the estimation amounts to some kind of statistical procedure". A typical one is to minimize the χ^2-value as will be described in Subsection 2.3.2.

From the stochastic uncertainties of the hit positions follows an uncertainty in the track parameters, which is also calculated by the fitting algorithm and is expressed by a covariance matrix.

The separation of track fitting from track finding is an artificial one as both can be mixed together. Track finding procedures can already do a first estimate of the track parameters or even do the final fitting. Track following methods that implement the Kalman filter for example not only deliver a track when finished but also the track parameters[4]. In this case it might make more sense to distinguish between the steps "seed finding" and "track finding + fitting".

2.1.1.3. Ambiguity resolving

After track finding and fitting are done, one is left with a collection of tracks that all seem to fulfill the track hypothesis. It may however be that some of them are not compatible, i.e. share hits.

In principle it is possible that two different tracks cross each other exactly at a detector surface and create a hit that belongs to both of them. It is however extremely rare. On the other hand, the reconstruction algorithms work from the hits and ghost tracks that get created by them can also contain hits of true tracks. For example, a reconstructed track could consist of 4 background hits and two hits belonging to two different real track. This track will then be incompatible with those two real tracks. This makes it very likely that any incompatibilities are due to ghost tracks and should be removed. It is however not necessarily crucial to resolve this during track finding, as additional information at a later stage, for instance from the calorimeters can help to further distinguish between true and ghost tracks. On the other hand, particle flow algorithms need the reconstructed tracks for better pattern recognition in the calorimeters. Maybe the best approach would be if the different stages of reconstruction could communicate and iterate on a final solution step by step. Implementing this would, however, probably be a non-trivial task.

In order to resolve incompatible track situations different algorithms can be used. First the goal needs to be defined: One wants a subset of the tracks that is in itself completely compatible and maximizes some sort of quality. The quality could for example be the sum of the χ^2-probabilities or the mean χ^2-probability.

The maybe simplest method one could come up with is this easy and fast algorithm:

1. Find the track with the highest quality

[4] Although it usually is still necessary to use the smoothing method of the Kalman filter after the track is found in order to get the best estimate at all measurement sites.

2. Erase all tracks that are incompatible with it
3. Remove the track from the collection and put it in the final collection
4. Repeat until the collection is empty

The downsides are that there is no quality criterion that would separate true tracks from background completely and that this algorithm allows a single track to discard many incompatible tracks, while keeping as many tracks as possible seems favorable. One could say that if the sum of the single tracks qualities is to be maximized, this algorithm is suboptimal in such situations. An alternative is the Hopfield Neural Network, which will be discussed in Subsection 2.3.3.

2.1.2. Particle identification (PID)

The determination of the particle identities is an important task, as it allows classification of the events. This is especially needed for event selection in physics analysis where only specific interactions are of importance.

In this context it is helpful that only a handful of particles live long enough to reach the different detectors in a typical high energy physics experiment: Electrons, muons, neutrons, protons, kaons, pions, photons and their antiparticles (if they have one) can be detected. Neutrinos are also long-lived enough, but cannot be detected by the usual kinds of detectors.

The subdetectors used in the ILD detector are fairly typical w.r.t the general areas of measurement[5]: from inside to outside there is tracking, electromagnetic calorimetry (ECAL), hadronic calorimetry (HCAL) and muon detection.

The distinction between the particles can partly be done by using their different interactions with the subdetectors [40]:

- **Photons** do not interact with the tracking detectors and create a shower in the electromagnetic calorimeter.
- **Electrons and positrons** are measured in the tracking part and are stopped by the ECAL.
- **Muons and antimuons** leave traces in all subdetectors.
- **Protons, antiprotons, charged kaons and pions** interact mainly with the tracking detectors, ECAL and HCAL.
- **Neutrons and** K_L^0 are detectable by the ECAL and the HCAL. The largest part of the energy is deposited in the HCAL [40].
- **Neutrinos and antineutrinos** cannot be seen by any of those detectors.

[5]For example the same ones are found in this order in CMS and ATLAS

Additional knowledge comes from the tracking: the sign of the helix in the magnetic field is determined by the charge of the particle, so one can distinguish particles and antiparticles.

Another important source of information is the mass of the particles. From tracking the momenta of the charged particles are known. Via $E^2 = m^2c^4 + p^2c^2$ the mass can be determined after measuring the energy in the calorimeters. All these particles have sufficiently different mass, making it easy to distinguish them, except that the masses of muons and pions ($m_\mu(105.66\,MeV) \simeq m_\pi(139.57\,MeV)$) are of the same order. But the latter can be distinguished using the fact that mainly only the muons reach the muon detector and that they do not cause particle showers in the calorimeters like pions.

Neutrinos can only be reconstructed indirectly. In an electron-positron collider the energy and momenta are well known. So when the reactions at the primary and secondary vertices are reconstructed one can compute the energy of the neutrinos taking part by using energy and momentum conservation.

2.1.3. Vertex reconstruction

The vertices, i.e. the places where particles are produced, are where the interesting physics processes happen.

Usually primary and secondary vertices are distinguished [21]:

- The primary vertex is the first vertex in an event. At the interaction point two particles collide (or a particle with a fixed target) and create new particles.

- Secondary vertices are simply all others. Particles created by the primary or another secondary vertex can decay (e.g. B-Mesons will decay into other particles shortly after creation) or interact with detector material.

Vertices are found by extrapolating the tracks of different found particles to a common point in space and possibly time. But as there can be many vertices in one event, one has to first determine which tracks belong to the same vertex. As in track finding, there are many different methods that can be applied, such as clustering or topological methods.

After candidates for vertices are found they get fitted in order to estimate their position and covariance matrix. Next, the single tracks are tested if it is likely that they truly belong to the vertex and outliers are removed. With the decimated number of tracks the vertex is fitted again. Also a kinematic fit can be done: from the momentum and energy conversion the mass hypothesis of the particles partaking in the vertex process can be tested.

In combination with particle identification the event is then fully reconstructed and handed over to physics analysis, where the properties of the underlying physics are determined.

2.2. The challenge of track reconstruction

There is no universal ideal solution to the problem of track reconstruction, because it is highly dependent on the environment. As an example, the hits of a typical ILC event without any background are depicted for the TPC in Figure 2.2 and the FTD in Figure 2.3. In both pictures all hits are shown, but no tracks. In the TPC the pad rows are so close that the tracks can be easily seen[6]. The fewer layers of the FTD make it quite hard to make out any tracks. Nevertheless tracking in the TPC is far from easy. Especially with background and tracks that are very close, advanced algorithms are necessary. But the key point here is that the tracking situations are quite different. Besides the detector type and specifics, also the energy of the events, the type of background, the computing hardware and its limitations as well as the the weighing of the different quality benchmarks of the reconstruction are important to keep in mind when designing a tracking algorithm.

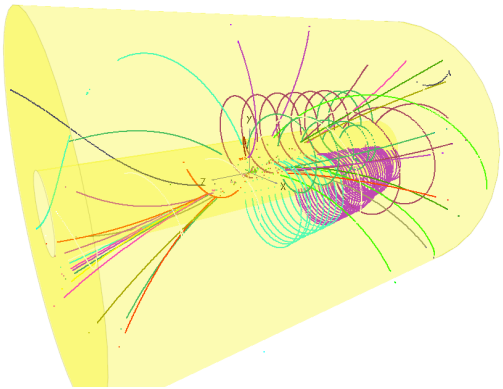


Figure 2.2.: Event in the TPC without background

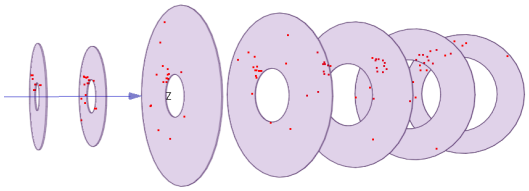


Figure 2.3.: Event in the FTD without background

As an example for different influences, here are some of the boundary conditions of tracking for the FTD:

- Tracks have only a few hits, as there are only 7 layers on the FTD, i.e. the number of hits in a track is very limited. Most often tracks in the FTD will have around 5 hits (see Figure 2.4).
- The silicon detectors cause more multiple scattering and energy loss than the TPC.
- Especially the inner disks will see a large beam-induced background of e^-e^+ pairs due to proximity to the interaction point.
- The pixel disks will get pile-up background from multiple events due to readout delay.

[6]The hits of the different tracks have different colors in Figure 2.2, but even if they had the same color, they could in this case be rather easily distinguished.

- The strip disks will have ghost hit background, especially from jets.
- There are overlapping regions in the FTD, which need to be taken into account.
- There can always be a hardware failure or an over-occupied sensor, so skipping layers has to be possible.
- Often, the tracks will not hit the first layers of the FTD at all, for geometric reasons (see Figure 2.5).

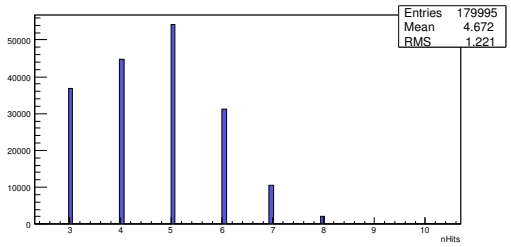


Figure 2.4.: Number of hits of tracks in the FTD (with 3 hits or more)

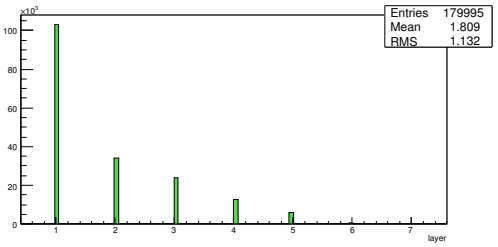


Figure 2.5.: Number of the first layer of the FTD hit by a true track

2.3. The tools

The algorithms "cellular automaton" for track finding, "Kalman filter" for track fitting and "Hopfield neural network" for ambiguity resolving have been chosen for the track reconstruction in the forward direction of the ILD detector. In 2.3.1 - 2.3.3 these algorithms are presented, as well as the reasons for their use in the packages.

2.3.1. The cellular automaton

The cellular automaton (CA) is a tool that originally did not come from high energy physics, but that got adapted in the 1990s [33]. There is no point in looking for the "first" use of a cellular automaton, as the basic idea behind it is a very general one: "Cellular Automata are discrete dynamical systems whose behavior is completely specified in terms of a local relation, ..." [58]. Many systems can qualify as cellular automata; all they need is a discrete state that dynamically changes depending on the local environment. The naming derives from biological cells, which also give a good example. Cells are discrete and their behavior is determined by their local environment. In the context of the present work, the comparison to biological cells is too much off-topic, so the famous "Game of Life" by John Horton Conway can serve as a far easier example.

This “game” is a simulation based on the principles of cellular automata. The setting is a two dimensional grid, such as a computer monitor. The single entities of the grid (e.g. the pixels of the monitor) are called the “cells”. They are, as mentioned above, discrete. Dynamics come into play, by letting the situation evolve via locally applicable rules.

These rules are the heart of every cellular automaton, and for the game of life they read as follows: A cell has a state which is either “alive” or “dead”. Dead cells are white pixels, alive cells are black. Cells can interact with the surrounding 8 cells, which are called neighbors. Overpopulation, which means more than 3 alive neighbors leads to death, as does underpopulation (less than 2 neighbors). Dead cells surrounded by 3 alive cells become alive again, which is reproduction.

These rules and the starting configuration completely determine the evolution of the system. An example is shown in Figure 2.6, where starting from one configuration a few iterations are made. One can see how the configuration evolves over time. This is far more impressive when animated and with more complex starting positions. What is amazing is that although the rules are so simple, many different stable patterns can emerge. The example given in Figure 2.6 results in a stable loop; ultimately after 15 iterations it returns to its beginning. Other objects can keep their overall (oscillating) shape and traverse through space. Certain objects can create those gliders periodically, while others have the potential to annihilate them.

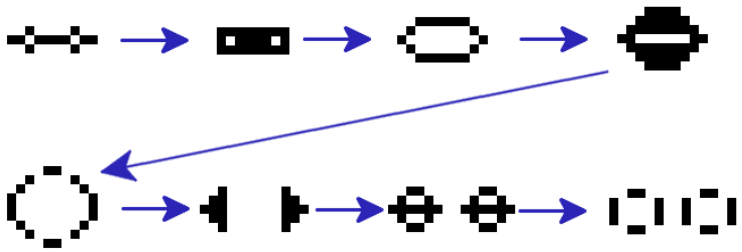


Figure 2.6.: Iterations in the ”game of life”

Already this simple game shows that stable patterns can emerge in such a setup and how dependent the situation is on the input, i.e. the starting values. A slightly different start in Figure 2.6 would have resulted in a non-stable situation.

Such a cellular automaton implementation gives the impression to be useful as a simulation tool rather than for reconstruction. But the high dependence of the outcome on the rules and the situation at start suggests that with the right rules the cellular automaton may be used to find patterns in a given starting situation. A simple ansatz could be formulated with the question “Can rules be formulated, such that cells forming a certain pattern survive and others do not?”

The first documented use of the cellular automaton in such a way in high energy physics was done by Kisel and Ososkov [33] for track reconstruction in a multi-wire proportional chamber for the ARES experiment. Due to the setup, the situation

consisted of rectangles next to each other, very much resembling the situation in the game of life. The rules were formulated in such a way that cells that could be part of a track survive. Also the possibility of hardware malfunction in single wires was taken into account.

Later Kisel worked for the NEMO collaboration [32], where he refined the tracking with the cellular automaton. This time Geiger drift cells in multiple layers where used for tracking. While in ARES the wires and their rectangular surrounding were the cells, for NEMO a further abstraction took place, and segments – the connections between two hits in neighboring[7] layers became the cells of the CA. The rules were that only segments with a compatible angle (below a certain cut-off value) that share a hit in the middle are compatible and called "neighbors". Other segments above the cut-off angle, even if they share the middle hit, are not neighbors.

The states of the cells were not binary anymore; a positive integer number starting with 1 was used. In each iteration the CA checks all segments for a neighbor on the inside with the same state. If one is found, then at the end of the iteration the state of the segment is raised by 1. This is continued until no changes of state happen anymore. With this configuration of rules, at the end the segments will have states corresponding to their place in a possible track. Tracks then consist of segments with states in descending order, for example (when going from outside to the inside): 4,3,2,1. Starting from the segments with the highest states, segments with a next smaller state are added to the track candidate. This way only tracks with a proper order (like 5,4,3,2,1) are collected, which sorts out much of the combinatorial background.

This procedure is the basis from which the implementations in CATS [3, 2, 17], the Belle II track reconstruction package [38] and the track reconstruction in the packages `KiTrack` and `ForwardTracking` for the ILD detector were developed.

The cellular automaton in this form qualifies as a semi-global track finding algorithm. While all segments are updated synchronously, the rules that dictate the behavior are local. It searches for the structure of a track while inserting information by cuts as early as possible.

2.3.1.1. The cellular automaton used in ForwardTracking

The implementations of the cellular automaton for track reconstruction differ from experiment to experiment, due to personal likings and programming styles, different requirements, hardware and environments. In Appendix A a more detailed introduction to the cellular automaton as used in `KiTrack` and `ForwardTracking` is given, with examples for a toy detector as well as for the Forward Tracking Detector in the ILD.

Two specifics are worth pointing out: First, the interaction point is used as an additional hit, which adds information, but with the cost of losing tracks that come from places far from the IP. So far the loss seems to be rather small, especially as

[7] Also skipping a layer was possible.

the FTD does not cover enough area in order to find most of the tracks from places distant to the IP. This is however changeable and will be changed if it is required by the situation and by a detailed analysis. The second specifics are the criteria that are used by the cellular automaton, i.e. the cut-off values that qualify segments to be neighbors. While in earlier experiments [32] the angle between segments was used, for the ILD detector a multitude of criteria can be applied.

Criteria In order to build the segments of the cellular automaton as well as for operating it, criteria need to be defined. A criterion is a value that is calculated from two segments, such as the angle between them. For every criterion cut-off values are formulated (e.g. the angle of a segment must be between a minimum and a maximum value). These cut-offs enable to distinguish between neighbors (that can be part of a track) and combinatorial background. Of course each of these criteria also is a potential source of efficiency loss, as some tracks will always be outside the cuts. One has to make a compromise between goals like efficiency, ghost rate, speed and maintainability.

Different experiments use different definitions of cells. In order to reduce combinatorics it is desirable to sort out wrong combinations early on, e.g. already when starting with single hits it is a good idea to distinguish between compatible and incompatible ones. For this reason the cellular automaton for the FTD uses the concept of segments of different lengths. Starting with the single hits, already those are considered as the shortest possible segments and called 1-hit-segments. These hits are only connected if they fulfill criteria designed for them. An example for a criterion for two single hits is their distance. If they are much more apart than the layers they are on, they probably do not belong to the same track. An example is shown in Figure 2.7, where instead of connecting all hits only those fulfilling a list of criteria are connected.

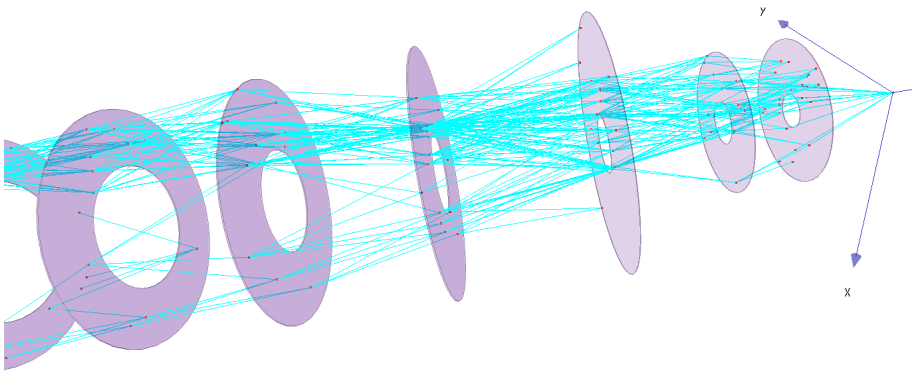


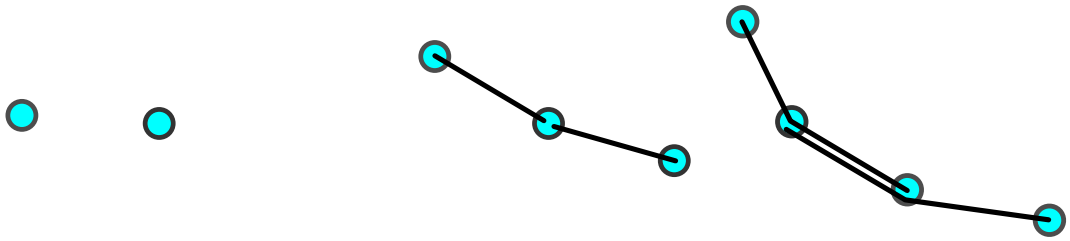
Figure 2.7.: Connected 1-hit-segments

Based on these criteria the cellular automaton can perform, making sure that

only segments survive that are part of a connection to the IP, where all segments in the connection fulfill the criteria. As a next step one can take the 1-hit-segments and form longer segments with 2 hits from them. These 2-hit-segments can then be tested again with applicable criteria such as the angle between two such segments (like was done in the NEMO experiment). These are what is classically referred to as segment. With these 2-hit-segments again the cellular automaton is used to sort out combinatorics. One can take this principle to the next stage and make even longer segments consisting of 3 hits: 3-hit-segments. Of course the cellular automaton procedure is done with them as well.

For every length of segments the cellular automaton is run, guaranteeing a connection of segments fulfilling multiple criteria, i.e. forming a possible track. With every rerun segments get sorted out, shrinking the overall number of track candidates in the cellular automaton. After each run the surviving segments are combined to form the next longer stage. In principle there is no limit to this concept and segments could be made even longer, but as the length grows the quality[8] of the track candidates does too and dismissing wrong ones becomes harder. In the implementation for the FTD the step with 3-hit-segments already has only a very small effect, although the cuts applied there are the strongest.

When the segments get tested for compatibility (i.e. whether they are neighbors) they have an overlap, only the innermost and outermost hit are different. That means that 2-hit-segments share the hit in the middle and that 3-hit segments share the 2 hits in the middle (see Figure 2.8). Therefore when comparing 1-hit-segments, there are 2 different hits. For 2-hit-segments there are 3 different hits (2 at the ends and 1 shared) and for 3-hit-segments there are 4 different hits (2 at the ends and 2 shared). A general formula for the number of different hits is $n = n_{segment} + 1$.



(a) two 1-hit-segments (b) two 2-hit-segments (c) two 3-hit-segments

Figure 2.8.: Two connected segments of different lengths

In Table 2.1 the different criteria that are contained in the KiTrack package are listed for the different numbers of hits they deal with.

[8]Quality here means how much they resemble what one is looking for: an approximately helical trajectory.

Table 2.1.: The different criteria available in the `KiTrack` package
(The time is given relative to the fastest criterion)

name	hits	time	description
DeltaRho	2	1.00	The difference of the distances to the z-axis: $\Delta\rho = \sqrt{x_2^2 + y_2^2} - \sqrt{x_1^2 + y_1^2}$.
RZRatio	2	1.00	The distance of two hits divided by their z-distance: $\frac{\sqrt{\Delta x^2+\Delta y^2+\Delta z^2}}{\lvert\Delta z\rvert}$
StraightTrackRatio	2	1.04	Best suited for straight tracks: if the line between the two hits points towards IP. Calculated is $\frac{\rho_1}{z_1}/\frac{\rho_2}{z_2}$, where $\rho = \sqrt{x^2 + y^2}$. Is equal to 1 for completely straight tracks.
DeltaPhi	2	1.30	The difference between the ϕ angles of two hits in degrees. ϕ is the azimuthal angle in the x-y plane w.r.t. the positive x axis: $\phi = \text{atan2}(y, x)$.
HelixWithIP	2	1.43	Checks if two hits are compatible with a helix through the IP. A circle is calculated from the two hits and the IP. Let α be the angle between the center of the circle and two hits. For a perfect helix $\frac{\alpha}{\Delta z}$ should be equal for all pairs of hits on the helix. The coefficients for the first and last two hits (including the IP) are compared: $\frac{\alpha_1}{\Delta z_1}/\frac{\alpha_2}{\Delta z_2}$. This is 1 for a perfect helix around the z-axis.
ChangeRZRatio	3	1.23	The coefficient of the RZRatio values for the two 2-hit-segments. Ideally this would equal 1.
2DAngle	3	1.23	The angle between two 2-hit-segments in the x-y plane.
2DAngleTimesR	3	1.46	The 2DAngle, but multiplied with the radius of the circle the segments form, in order to get better values for low momentum tracks.
3DAngle	3	1.25	The angle between two 2-hit-segments.
3DAngleTimesR	3	1.48	3DAngle times the radius of the circle.
PT	3	1.30	The transversal momentum as calculated from a circle in the x-y plane. This criterion includes knowledge about the magnetic field and in this way differs from the rest. A more basic version would be to either use the radius of the circle or its inverse Ω. Using p_T was chosen for reasons of readability.

IPCircleDist	3	1.30	From the 3 hits a circle is calculated in the x-y plane and the distance of the IP to this circle is measured.
IPCircleDistTimesR	3	1.30	Distance of the IP to the circle multiplied with the radius of the circle to take into account higher deviations for low transversal momentum tracks.
DistOfCircleCenters	4	1.66	Circles are calculated for the first and last 3 hits. The distance of their centers is measured.
RChange	4	1.66	The coefficient of the radii of the two circles.
DistToExtrapolation	4	2.21	From the first 3 hits the relation of α to Δz is calculated. This is used to predict x and y of the fourth hit for the given z-value. The distance of this prediction to the actual position in x and y is measured.
NoZigZag	4	2.30	A criterion to sort out tracks that make a zig zag movement. The 2-D angles are measured for the first and the last three hits. Then they are transposed to the area of $-\pi$ to π and multiplied. A zig-zagging track would give angles with different signs and therefore a negative multiplication result.
2DAngleChange	4	2.30	The coefficient of the 2-D angles.
3DAngleChange	4	2.41	The coefficient of the 3-D angles.
PhiZRatioChange	4	2.50	The coefficient of the PhiZRatio of the first 3 and the last 3 hits.

Usefulness of criteria There is a multitude of criteria one can come up with and there are certainly more possibilities than in Table 2.1. But not all criteria are equally useful. First, as they are used quite often in the cellular automaton in order to filter away the combinatorial background, they need to be fast. There is no reason why one could not implement a Kalman filter criterion for longer segments, but it costs quite some time. In Table 2.1 the calculation time, relative to the fastest one, is shown[9]. Each of the criteria is based on rather simple and fast calculations, so that the time used to do the actual calculation is in the order of the time needed to reserve memory for the variables.

Another important factor is the distribution of the criterion for true tracks. As an example in Figure 2.9 and 2.10 one can see the differences between the 2- and 3-dimensional angles between 2-hit-segments. The tail of the 2DAngle criterion

[9]Each criterion was used a thousand times with given segments by an executable. The executable was monitored with the profiling tool callgrind. The computing times of the criteria were normalized, so that the fastest one had the value 1.

reaches far to the right[10]. If one wants to use the 2DAngle criterion and not loose too much of the true tracks, one would have to keep segment connections with very large angles and thus be only able to sort out a tiny fraction of the combinatorial background. The 3DAngle criterion on the other hand is much more useful, having almost no tail beyond 10°.

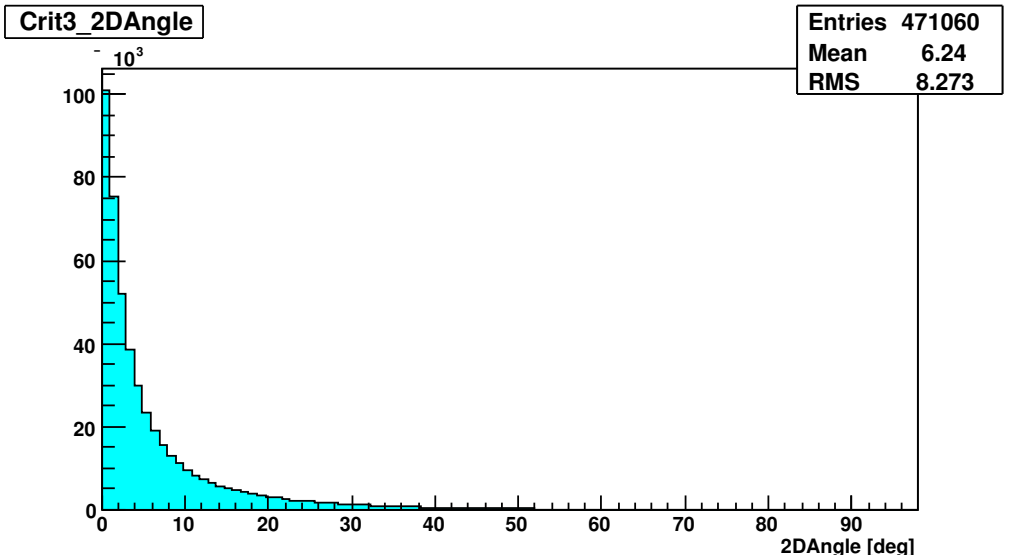


Figure 2.9.: The 2DAngle criterion

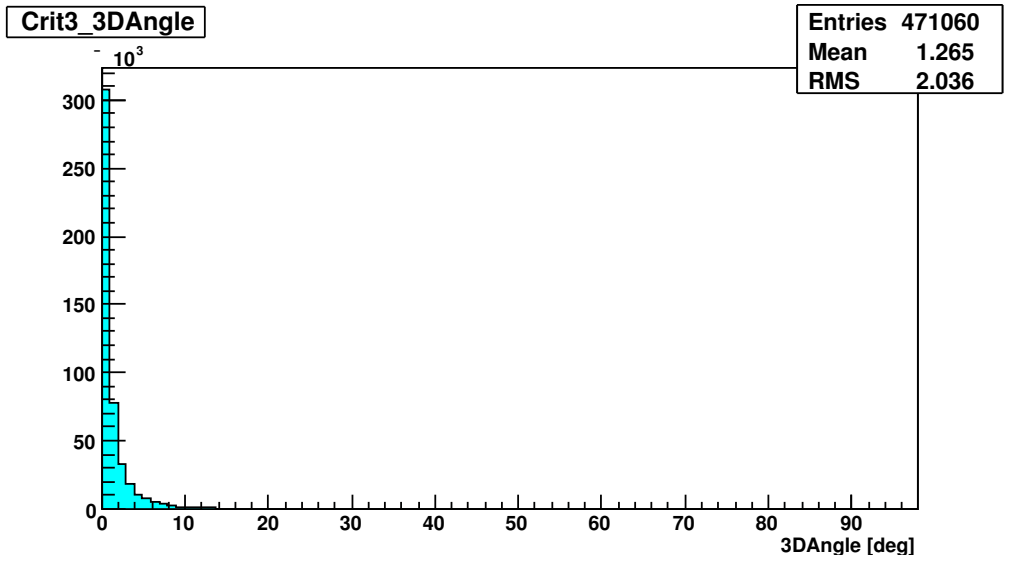


Figure 2.10.: The 3DAngle criterion

Another important factor is the dependence of the criterion on the transversal momentum. A good example for this is the criterion DistOfCircleCenters for 3-hit-

[10]The maximum value for these angles is 90°.

segments (i.e. 4 different hits). It calculates two circles: one from the first and one from the last 3 hits. Then the distance between the centers of the circles is measured. This works rather well for low transversal momentum tracks; it fails, however, for high p_T tracks. The reason is simple: high p_T tracks have a large radius, they are almost straight lines. A tiny change in one of the three hits that define the circle results in a huge change of the position of the center. It is even possible that the sign of the curvature changes with small deviations, giving a center of the circle in a completely different position. As for the ILD detector it is more important to keep high momentum tracks than low momentum ones, this criterion cannot be used[11].

Also criteria will have overlaps with other ones. If two segments are incompatible, this can be found by multiple criteria. So if criterion A and criterion B filter away the same, it would be a waste of computing time to use them simultaneously.

The criteria currently used in the reconstruction of FTD tracks are a subset of those in table 2.1 and are hand selected to avoid problems like those just mentioned, to keep the reconstruction fast and still efficient. For the future a clear definition catalogue that qualifies and disqualifies criteria could add further systematics and let the criteria be tuned and selected automatically.

The merits of the cellular automaton for track finding The main advantage of the cellular automaton is that information can be used very early. When set up the right way, the criteria are able to filter away much of the combinatorics as early as possible. Before two hits are considered to be compatible, they are tested for the different criteria. As many of those criteria can be formulated with easy calculations, the cellular automaton consumes relatively little computing time[12].

The implementation with integers as states of the cellular automaton additionally filters all segments that are not part of a longer valid chain of connections, i.e. that are not able to form a track.

The tunability of the single criteria and the possibility to easily add further ones suited to certain situations makes it a flexible tool that can be adapted to many different detector layouts.

[11] At least not in the search for high p_T tracks. It is however imaginable that in the future a dedicated low transversal momentum track finder module is used, and there it could certainly be of value.

[12] This is of course highly dependent of the tightness of the cuts and the specific implementation, but as an example, there is a current discussion to use the CA as an online trigger for Belle II, which requires a high speed. In `ForwardTracking` the needed computing time is not dominated by the CA, but by the Kalman filter: if the CA gives too many track candidates, the fitting can take a very long time.

2.3.2. The Kalman filter

Once track candidates are found by track search algorithms like the cellular automaton, one needs to estimate the parameters of the tracks. As previously mentioned, the path of a charged particle in a homogeneous magnetic field is a helix and can be described by 5 parameters for a given reference point. In order to determine these parameters a so-called fit is done, which means finding the parameters that best approximate the actual track and are in compliance with the measured values.

The global least squares fit, which is still very important in high energy physics, will be presented briefly and then the Kalman filter, which has since around 1987 probably become the most used fitting technique in this area, is discussed in more detail.

Formulas from [20] for the global least squares fitting and [19] for the Kalman filter are used. As both have overlapping use of variable names with different meanings, the notation in [19] is used and the other formulas are adapted to it. Also the additional variable $\boldsymbol{m}_p$ (definition below) is introduced for convenience.

The basis for fitting are the hits measured by the detectors. They provide one-dimensional (e.g. strip detectors) or two-dimensional (e.g. TPC, pixel, double sided strip) measurements[13].

The 5 parameters of the track are put together in the so called state vector[14] $\boldsymbol{x}$. One can use these parameters to predict measurements by extrapolating the track and determining where sensors are hit. This is not always solvable in an analytic fashion, but often has to be done by iterative algorithms like the Newtonian solver. These predicted measurements $\boldsymbol{m}_p$ are of course a function of the track parameters:

$$\boldsymbol{m}_p : \boldsymbol{x} \rightarrow \boldsymbol{m}_p(\boldsymbol{x}) \tag{2.4}$$

The global least squares method (LSM) works with the linear expansion of the actual track model by neglecting higher order terms:

$$\boldsymbol{m}_p(\boldsymbol{x}) = \boldsymbol{m}_p(\boldsymbol{x}^0) + \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}_0) + O((\boldsymbol{x} - \boldsymbol{x}_0)^2) \tag{2.5}$$

$$\boldsymbol{A} = \frac{\partial \boldsymbol{m}_p}{\partial \boldsymbol{x}}|_{\boldsymbol{x}=\boldsymbol{x}_0} \tag{2.6}$$

It is important to note that the appropriateness of the linear expansion is dependent on a good choice of parameters for the description of the path.

Let the matrix $\boldsymbol{W}$ be the weight matrix[15] of the measurements $\boldsymbol{m}$. Then the least

[13] A three-dimensional space point is only a two-dimensional measurement, because only two independent equations can be formed from it.

[14] e.g. in `LCIO` this would be the vector $(\Omega, \phi_0, d_0, z_0, \tan(\lambda))$

[15] The weight matrix is the inverse of the corresponding covariance matrix, which is by definition symmetric and positive-definite.

squares method aims to minimize the so called χ^2 value[16]:

$$\chi^2 = [\boldsymbol{m}_p(\boldsymbol{x}^0) + \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}_0) - \boldsymbol{m}]^T \boldsymbol{W} [\boldsymbol{m}_p(\boldsymbol{x}^0) + \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}_0) - \boldsymbol{m}] \tag{2.7}$$

It is the minimization of the quadratic distances of the predicted to the measured values, but with an additional weight on each measurement, so that measurements with huge errors contribute less than more accurate ones.

In order to minimize χ^2, $\frac{\partial \chi^2}{\partial \boldsymbol{x}} = 0$ has to be calculated. To do the minimization, at least as many measurements as track parameters are required. Otherwise the equation system is under-determined. Setting the derivative to 0 gives:

$$\boldsymbol{x} = \boldsymbol{x}^0 + (\boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{W} [\boldsymbol{m} - \boldsymbol{m}_p(\boldsymbol{x}^0)] \tag{2.8}$$

$$\mathrm{cov}(\boldsymbol{x}) = (\boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A})^{-1} \tag{2.9}$$

Thus one has estimated the parameters of the helix, minimizing χ^2.

It can be shown that if the linear approximation is valid and the variation of the errors w.r.t the track parameters are small enough to approximate them as constant in the neighborhood of the path , the linear LSM "has minimum variance among the class of linear and unbiased estimates"[20, p.246].

There is however a flaw in the approach so far, namely that the tracks get disturbed by material effects leading to tracks being multiple helices attached to one another, changing parameters every time there is interaction with material. This can be included in the linear LSM by the price that the matrices that need to be inverted become very large and the computing time rises considerably. Also the implementation becomes more complex.

This problem is solved by the Kalman filter, which works in a local way in contrast to a global fit. The Kalman filter is an iterative procedure, not taking into account all hits at a single time, but adding them sequentially[17].

The parameters of the track change depending on the reference point. If for every new measurement site a new reference point on the site is chosen, the parameters vary with the position $\boldsymbol{z}$ on the track (and the corresponding measurement site).

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{z}) \tag{2.10}$$

As there is only a finite number of sensors in an experiment, these positions on the track can be treated in discrete steps. The material between two sites is usually concentrated to one or more infinitely thin layer containing all the material effects

[16] In case of unbiased measurements with Gaussian errors, and a strict linear model, these values follow a χ^2-distribution.

[17] This sub-sequential adding makes it useful for local tracking methods, for example as the so called "Combinatorial Kalman filter" or "Concurrent Track Evolution" [43].

between the sites. The parameters at a place k can then be calculated from the parameters at the place $k-1$:

$$\boldsymbol{x}(\boldsymbol{z}_k) \equiv \boldsymbol{x}_k = \boldsymbol{f}_{k-1}(\boldsymbol{x}_{k-1}) + \boldsymbol{w}_{k-1} \tag{2.11}$$

$\boldsymbol{f}$ is called the track propagator, because it propagates the parameters from one measurement site to the next. $\boldsymbol{w}_{k-1}$ is a random disturbance that occurs between $k-1$ and k. This could be for example multiple scattering.

The measurement at a detector is dependent on the track parameter. If a detector measures x and y for a given z, these values change in dependence on the track parameters. In this process too there is an additional error that always comes with measurement:

$$\boldsymbol{m}_k = \boldsymbol{h}_k(\boldsymbol{x}_k) + \boldsymbol{\varepsilon}_k \tag{2.12}$$

$\boldsymbol{h}_k$ calculates the measured values from the parameters.

The usual Kalman filter is not only discrete, but linearized as well, which means that $\boldsymbol{h}$ and $\boldsymbol{f}$ can be described by matrices $\boldsymbol{H}$ and $\boldsymbol{F}$, giving the basic equations:

$$\boldsymbol{x}_k = \boldsymbol{F}_{k-1}(\boldsymbol{x}_{k-1}) + \boldsymbol{w}_{k-1} \tag{2.13}$$

and

$$\boldsymbol{m}_k = \boldsymbol{H}_k(\boldsymbol{x}_k) + \boldsymbol{\varepsilon}_k \tag{2.14}$$

The Kalman filter works in 3 different steps:

- Prediction is estimating track parameters at the next measurement site. From them estimated measurement values can be calculated.
- Filtering, or Updating calculates the parameters and error matrices at a given point from the measurement at the point and the previous predicted state vector.
- Smoothing means to go back to previous hits and recalculating their parameters based upon the full information acquired so far. This is essential, as the Kalman filter, if no smoothing is used always has full information from previous measurements at the last one, but only there. Smoothing brings this information back to previous sites, making the estimates there as good as possible.

The variables used in the following equations:

- $\boldsymbol{Q}_k = \mathrm{cov}\{\boldsymbol{w}_k\}$ Covariance of the process noise
- $\boldsymbol{V}_k = \boldsymbol{G}_k^{-1} = \mathrm{cov}\{\boldsymbol{\epsilon}_k\}$ Covariance of the measurement error (= inverse of the weight matrix for the measurement errors)
- $\boldsymbol{x}_{k,t}$ true value of $\boldsymbol{x}$ at k

- $\boldsymbol{x}_k^i$ estimate of $\boldsymbol{x}_k$ based on the measurements up to i. ($\boldsymbol{x}_k^k = \boldsymbol{x}_k$)
- $\boldsymbol{C}_k^i = \mathrm{cov}\{\boldsymbol{x}_k^i - \boldsymbol{x}_{k,t}\}$ Covariance of the difference of the true value to the estimated one.
- $\boldsymbol{r}_k^i = \boldsymbol{m}_k - \boldsymbol{H}_k \boldsymbol{x}_k^i$ The residual of the measurement: difference between estimated measurement and actual one.
- $\boldsymbol{R}_k^i = \mathrm{cov}\{\boldsymbol{r}_k^i\}$ Covariance of the residual.

Before the Kalman filter can start with the first prediction, it needs to have an initial guess (to be given extremely small weights). This could for example come from a quick helix fit of three points of the track with a huge covariance matrix, so that the guess has negligible impact. Also track parameters could already be estimated in the track finding procedure.

2.3.2.1. Prediction

The track parameters at a measurement site $k-1$ are extrapolated with the evolution matrix $\boldsymbol{F}_{k-1}$ to the place k.

$$\boldsymbol{x}_k^{k-1} = \boldsymbol{F}_{k-1} \boldsymbol{x}_{k-1} \tag{2.15}$$

The covariance of the track parameters has to be extrapolated as well:

$$\boldsymbol{C}_k^{k-1} = \boldsymbol{F}_{k-1} \boldsymbol{C}_{k-1} \boldsymbol{F}_{k-1}^T + \boldsymbol{Q}_{k-1} \tag{2.16}$$

From the estimated parameters at k ($\boldsymbol{x}_k^{k-1}$) a measurement result can be predicted:

$$\boldsymbol{m}_{k,p}^{k-1} = \boldsymbol{H}_k \boldsymbol{x}_k^{k-1} \tag{2.17}$$

The difference of this predicted measurement $\boldsymbol{m}_{k,p}^{k-1}$ to the true measurement[18] $\boldsymbol{m}_k$ is the so called residual:

$$\boldsymbol{r}_k^{k-1} = \boldsymbol{m}_k - \boldsymbol{m}_{k,p}^{k-1} \tag{2.18}$$

The covariance of the residual is:

$$\boldsymbol{R}_k^{k-1} = \boldsymbol{V}_k + \boldsymbol{H}_k \boldsymbol{C}_k^{k-1} \boldsymbol{H}_k^T \tag{2.19}$$

This is the covariance of the actual measurement error added to the covariance coming from the extrapolation.

[18] It is noteworthy to stress the point that $\boldsymbol{m}_k$ and $\boldsymbol{V}_k$ are the only actual measured values in the whole procedure. The covariance of the measurement error $\boldsymbol{V}_k$ can be determined beforehand in a simplified way, or more accurately be calculated based on the measurement signal. For example if a cluster is built in a silicon sensor, the shape of the cluster can influence the error.

2.3.2.2. Filtering

Here the filtering method with the so called "gain matrix formalism" is shown. There is an equivalent formalism that is called "weighted means formalism" that can be used as well and can be more efficient for small state vectors.

Filtering, or updating, is the important step in which the prediction is combined with the actual measurement and the parameters of the track are modified accordingly. This is done in by the Kalman gain matrix $\boldsymbol{K}_k$:

$$\boldsymbol{x}_k = \boldsymbol{x}_k^{k-1} + \boldsymbol{K}_k \boldsymbol{r}_k^{k-1} = \boldsymbol{x}_k^{k-1} + \boldsymbol{K}_k(\boldsymbol{m}_k - \boldsymbol{H}_k \boldsymbol{x}_k^{k-1}) \tag{2.20}$$

The Kalman gain matrix transforms the information from the residual to the actual change of the parameter vector. If one were to ignore the covariances, one would simply transform the residuals of the measurement to residuals of the parameters by multiplying with $\boldsymbol{H}_k^T$ and add that to the old parameter vector. $\boldsymbol{K}_k$ contains exactly this, plus the influences from the different errors:

$$\boldsymbol{K}_k = \boldsymbol{C}_k^{k-1} \boldsymbol{H}_k^T (\boldsymbol{V}_k + \boldsymbol{H}_k \boldsymbol{C}_k^{k-1} \boldsymbol{H}_k^T)^{-1} \tag{2.21}$$

As stated $\boldsymbol{H}_k^T$ transforms measurement values to parameter values. On the left of it is $\boldsymbol{C}_k^{k-1}$, the covariance of the parameters. The higher the errors in the determination of the parameters from k to $k-1$ are the higher is the influence of the residuals. If one is very uncertain about the parameters, the measurement gets more weight.

In the bracket to the right on the other hand, the covariance of the measurement errors $\boldsymbol{V}_k$ and the covariance of the prediction of the measurement are added. Taking the inverse of it gives a weight of the measurement. The higher the errors in the measurement or the prediction of the measurement values, the lower the influence of the measurement on the result.

The Kalman gain matrix can also be written with the covariance $\boldsymbol{C}_k$ as:

$$\boldsymbol{K}_k = \boldsymbol{C}_k \boldsymbol{H}_k^T \boldsymbol{G}_k \tag{2.22}$$

$$\boldsymbol{C}_k = (\boldsymbol{1} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{C}_k^{k-1} \tag{2.23}$$

The residual $\boldsymbol{r}_k$ can be easily determined by:

$$\begin{aligned} \boldsymbol{r}_k &= \boldsymbol{m}_k - \boldsymbol{H}_k \boldsymbol{x}_k \\ &= \boldsymbol{m}_k - \boldsymbol{H}_k (\boldsymbol{x}_k^{k-1} + \boldsymbol{K}_k \boldsymbol{r}_k^{k-1}) \\ &= \boldsymbol{m}_k - \boldsymbol{H}_k \boldsymbol{x}_k^{k-1} - \boldsymbol{H}_k \boldsymbol{K}_k \boldsymbol{r}_k^{k-1} \\ &= \boldsymbol{r}_k^{k-1} - \boldsymbol{H}_k \boldsymbol{K}_k \boldsymbol{r}_k^{k-1} \\ &= (\boldsymbol{1} - \boldsymbol{H}_k \boldsymbol{K}_k) \boldsymbol{r}_k^{k-1} \end{aligned} \tag{2.24}$$

And the covariance of the residual:

$$\begin{aligned} \boldsymbol{R}_k &= (\mathbf{1} - \boldsymbol{H}_k \boldsymbol{K}_k)\boldsymbol{V}_k \\ &= (\mathbf{1} - \boldsymbol{H}_k \boldsymbol{C}_k \boldsymbol{H}_k^T \boldsymbol{G}_k)\boldsymbol{V}_k \\ &= \boldsymbol{V}_k - \boldsymbol{H}_k \boldsymbol{C}_k \boldsymbol{H}_k^T \end{aligned} \tag{2.25}$$

Now that all the values at k are calculated, one can determine the χ^2 increment (χ^2_+). As before the χ^2 value is the quadratic form of the residuals multiplied by their weight:

$$\chi^2_+ = \boldsymbol{r}_k^T \boldsymbol{R}_k^{-1} \boldsymbol{r}_k \tag{2.26}$$

$$\chi^2_k = \chi^2_{k-1} + \chi^2_+ \tag{2.27}$$

This is a very important value, because it contains an estimate for the goodness of the fit. The χ^2 values of different fits will follow a χ^2-distribution if the measurement errors are unbiased and Gaussian, and the linear model is a good approximation. The distribution depends only on the number of degrees of freedom (see Figure 2.11). If for example a helix is fitted with 4 2-dimensional measurements, this gives 8 equations. 5 are at least needed, leaving 3 degrees of freedom.

As χ^2 is dependent on the residuals, a higher χ^2 means more deviation of the measurements from the fit. If for example one would fit a circle to a number of hits lying on a perfect circle, the result would be a χ^2 of 0, as all the hits have no distance to the actual circle. If one instead had hits distributed along a square and tried to fit a circle with that, of course not all hits would be on the fitted circle, but differ from it, giving it a high χ^2.

If the track model is correct[19] and the effects that are taken into account in the propagation are too[20], then high χ^2 values are improbable for true tracks. For every χ^2 one can calculate the probability that such a high χ^2 or a higher one occurs for the given degrees of freedom[21]. Random combinatorial background from the track finding algorithms can be discarded to a certain degree, if one applies a probability cut here. For example if all tracks with a fit of a probability of below 0.005 are cut, then only 0.5% of true tracks get discarded while many among the combinatorial background are sorted out. The χ^2 can also be used during the fitting procedure as well, by not adding hits that would give a large χ^2_+. A more sophisticated version of that can be used for outlier removal in the track.

[19] Deviations could for example come from not taking into account a not perfectly homogeneous magnetic field.

[20] Possible deviations: material over or underestimation, incorrect treatment of multiple scattering or energy loss.

[21] This "upper tail probability" is simply 1 minus the cumulative distribution function for this value

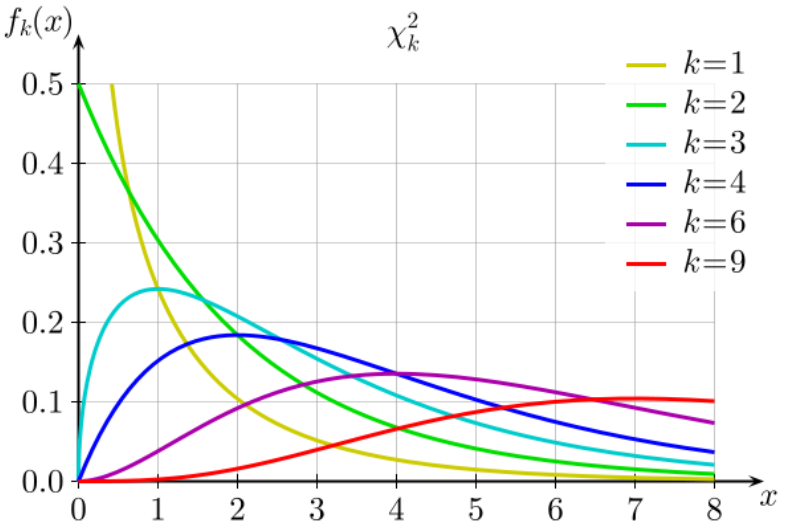


Figure 2.11.: χ^2 distributions with different degrees of freedom (k)

2.3.2.3. Smoothing

Once the Kalman filter reaches the last hit in the fit, the full information from all hits has been used for this last hit, giving it the optimal estimate of parameters. But the hits further back in the fit have less and less accurate estimates due to their lack of information at that points. For this reason fits in collider experiments are usually done from outside to inside, as the most important point for knowing the parameters is near the vertex. This is however not necessary with the use of the so called smoothing, where the Kalman filter goes back again to previous points, once all the information is available, and updates them once more.

For smoothing the smoother gain matrix $\boldsymbol{A}_k$ is used[22]:

$$\boldsymbol{x}_k^n = \boldsymbol{x}_k + \boldsymbol{A}_k(\boldsymbol{x}_{k+1}^n - \boldsymbol{x}_{k+1}^k) \tag{2.28}$$

$\boldsymbol{x}_k^n$ is the smoothed state vector at the place k. The term in the bracket is the difference between the smoothed state vector of the place $k+1$ and the state vector at $k+1$ as extrapolated from k. So it is the difference between the extrapolation and the best guess we have at that site. This difference needs to be extrapolated from $k+1$ to k in order to correct the state vector there. This is done by the smoother gain matrix $\boldsymbol{A}_k$:

$$\boldsymbol{A}_k = \boldsymbol{C}_k \boldsymbol{F}_k^T (\boldsymbol{C}_{k+1}^k)^{-1} \tag{2.29}$$

The covariance matrix of the smoothed state vector is:

$$\boldsymbol{C}_k^n = \boldsymbol{C}_k + \boldsymbol{A}_k(\boldsymbol{C}_{k+1}^n - \boldsymbol{C}_{k+1}^k)\boldsymbol{A}_k^T \tag{2.30}$$

[22] An equivalent approach for smoothing would be to run a second filter in the backward direction, calculating at each place k the weighted mean of the forward filter and the backward prediction.

After smoothing all the way back, the parameters are determined at every (discrete) site of the track. From them the momentum and the charge of the particle can be calculated.

2.3.2.4. The merits of the linear Kalman filter

The Kalman filter has meanwhile risen to be the standard tool for track fitting in high energy physics for different reasons:

- Its local approach keeps the calculation time relatively low[23], as the size of the matrices that have to be inverted[24] is determined by the dimension of the measurements and the state vector. The largest matrices are usually of the size 5×5 (dimension of the state vector). Also the smoothing takes only little effort, if intermediate results of the previous filtering steps are kept.
- One does not lose any information by that approach and it is as efficient as a global least squares fit, taking all effects into account: "If $\boldsymbol{w}_k$ and $\boldsymbol{\epsilon}_k$ are Gaussian random variables, the Kalman filter is *the* optimal filter; no other filter can do better. In other cases it is simply the optimal linear filter." [19].
- Although it is already sophisticated statistics, it is easier to work with than other equally good methods. Material effects can more easily be included and altered.
- It fulfills important criteria for a good estimator: the variance is minimal (for a linear estimator), for Gaussian errors it is efficient and unbiased.
- The χ^2 values can allow to discard hits on the go or to give a feedback of track quality after the fit.
- Via the extension to a Combinatorial Kalman filter it can be used for track finding too [43].
- A non-linear generalization of the Kalman filter, the Gaussian-sum Filter is very effective to take non-Gaussian noise into account.

2.3.3. The Hopfield neural network

The Hopfield neural network can be used to resolve ambiguities in the final track collection and is, as the name suggests, a neural network invented by J.J. Hopfield[29]. Neural means that all elements are called neurons, because they can (like neurons in biological terms) be connected to and influence each other. The original neurons in the Hopfield net are binary, so they are either in state 0 or 1. Each neuron updates its state in each iteration. The new value it gets (again 0 or 1) depends on the other

[23]The calculation time is roughly proportional to the number of measurement sites.
[24]The computing time for inverting a matrix is proportional to the 3^{rd} power of its dimension.

neurons around it. Every neuron may be connected to it and have an activating or damping effect. If a neuron is updated the activation is calculated by:

$$a_i = \sum_{j \neq i} w_{ij} x_j \tag{2.31}$$

The x_j are the states of the neurons and the matrix w_{ij} gives the strength of the connection. The new state of the neuron x_i is then either 0 if a_i is below a certain activation threshold or 1 otherwise.

A simple example is shown in Figure 2.12. There are 4 neurons (in red) connected to each other (connections are in blue). The strengths of the connections are written besides them. (The dashed ones are 0, which just means there is no connection).

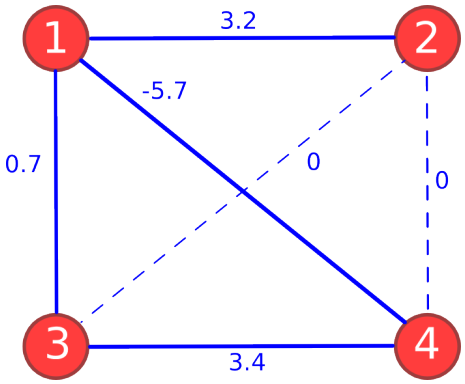


Figure 2.12.: Neurons connected to each other with different weights on the connections

The matrix $\boldsymbol{W}$ contains the strength of these connections:

$$\boldsymbol{W} = \begin{pmatrix} 0 & 3.2 & 0.7 & -5.7 \\ 3.2 & 0 & 0 & 0 \\ 0.7 & 0 & 0 & 3.4 \\ -5.7 & 0 & 3.4 & 0 \end{pmatrix} \tag{2.32}$$

In this case the connections are symmetric and therefore so is $\boldsymbol{W}$. All neurons are assumed to have state 1 at the moment. Updating neuron number 1 gives:

$$a_1 = \sum_{j \neq 1} w_{1j} x_j = 0 \times 1 + 3.2 \times 1 + 0.7 \times 1 - 5.7 \times 1 = -1.8 \tag{2.33}$$

The new state of the neuron now depends on the activation threshold. If 0 is chosen as activation threshold, below 0 means inactive and above 0 means active, so the neuron number 1 is now inactive (its state is now 0).

Though the original idea of Hopfield was to describe a way information can be handled by neural cells and how memory storage can work within that context, the Hopfield neural net can be used as a tool for many different areas like combinatorial optimization.

As shown in [53] the Hopfield neural network can be used to find maximal compatible sets and with a modification [22] also to find nearly optimal subsets, with optimal meaning as large as possible and with a quality as high as possible.

It works the following way: the matrix w_{ij} is set up like this:

$$w_{ij} = \begin{cases} -1 & \text{if i and j are incompatible,} \\ (1-\omega)/N & \text{if i and j are compatible.} \end{cases} \tag{2.34}$$

N is the number of neurons in the network and ω is a tunable parameter that controls how important the quality of a neuron is: the higher ω, the more influence comes from the quality. The updating of the neurons is done asynchronously (one at a time in a random order) to avoid cycling between states.

There is an additional vector holding the quality of the neurons:

$$w_{i0} = \omega \cdot q_i \tag{2.35}$$

where q_i is the quality of neuron i. This quality indicator (QI) must be between 0 (lowest quality) and 1 (highest quality). For example the χ^2-probability could be used as QI.

For this version of the Hopfield neural network the discrete states of the neurons get replaced by a real number between 0 and 1. So instead of being inactive (0) or active (1) there is a continuous interval in between.

At the beginning the states of all neurons are set to a small random number (closer to 0 than to 1, so they are close to inactive). Then the iterations begin and in each of them the state x_i of a neuron is calculated like this:

$$a_i = w_{i0} + \sum_{j \neq i} w_{ij} x_j \tag{2.36}$$

So in addition to the influence from the other neurons the quality of the specific neuron itself helps it getting activated. Instead of checking whether the value a is below or above a threshold, a so called activation function is used.

$$x_i = \frac{1}{2}(1 + \tanh(a_i/T)) \tag{2.37}$$

The activation function maps any value between $-\infty$ and ∞ to a number between 0 and 1. It is depicted in Figure 2.13 for different values of T. The lower T gets, the sharper the assignment, at $T = 0$ either 0 or 1 is assigned[25], which is precisely the situation when only discrete states are used. So one can call the activation function method a generalization of the discrete method, approaching it towards $T = 0$.

The idea behind T is that it resembles a temperature. Instead of neurons going into full activation or deactivation, while it is "hot" the temperature prevents them from doing this. If the system starts with a certain temperature and cools down gradually, this makes sure that the neurons get enough iterations so that the system can gradually evolve and avoid local minima, i.e. situations where the neural network

[25] i.e. the function approaches the Heaviside step function

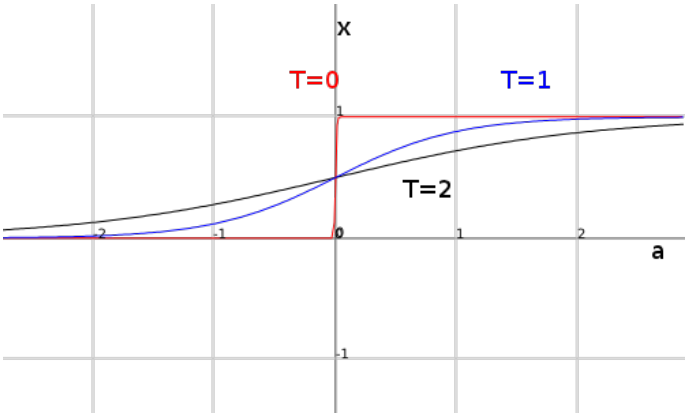


Figure 2.13.: Activation function

reaches a stable state that is not optimal. It is a bit like marbles in movement on a surface with cavities of different depth. If they are too slow at the beginning they will get stuck in the first cavity, even if it is just a minor one. Shaking the surface ensures that they will rather end up in one of the deeper cavities.

The start is at a finite temperature T and all neurons get updated in a random order. After the first round the temperature gets lowered by:

$$T_{n+1} = \frac{1}{2}(T_n + T_\infty) \tag{2.38}$$

As the temperature gets cooler, the winners and losers become gradually clearer. The procedure is stopped, once the changes of the states are sufficiently small (like below 0.01 for example). At this time all states above a certain threshold (like 0.75) are considered active and form the best subset.

The Hopfield neural network in this form is very fast, because it only relies on matrix multiplication, no inversion is needed like for the Kalman filter. In operation in `ForwardTracking` its computing time was always far below every other algorithm in the course of track reconstruction, so that optimizing speed was not needed at all.

In the application so far it has shown to be more than is needed in the sense that the simple algorithm presented in Sub-subsection 2.1.1.3[26] came very close in the results. This could come from the fact that most overlapping comes from clones at the moment and for them, the method of taking the best quality track is completely sufficient. Only when the situation becomes more entangled, the Hopfield neural network can use its advantage of optimizing for highest quality and total number of tracks. It works well as a compatible subset finder, but the conditions under which it can surpass the simpler algorithm should be further investigated in the future.

[26]Saving the track with highest quality, discarding all incompatible ones and repeating the step until all tracks are either discarded or saved.

3. Related work

As track reconstruction is a key element that is part of almost all high energy collider experiments, the number of related works would exhaust this thesis. This chapter will focus on experiments and track reconstruction strategies that are thematically linked to `ForwardTracking` (track reconstruction for the Forward Tracking Detector FTD of the ILD). All works are summarized rather briefly, there are always more details to be taken into account.

First, the track reconstruction software that was used for FTD track reconstruction before `ForwardTracking` and now still runs in parallel is discussed. Its importance lies in the direct comparison to `ForwardTracking`, which will be made in Chapter 5. As different experiments yield different environments for reconstruction, the comparison of two methods from different setups is hardly meaningful. So the opportunity to compare algorithms under the same circumstances is highly useful.

The next method discussed is the current tracking in CMS, as this is a contemporary experiment and a good example of the usage of the Kalman filter for track finding. Also the combinatorial Kalman filter used there could in principle be implemented in a similar fashion for the FTD tracking as well.

The next portrayed experiment, HERA-B, is already finished, but is mentioned as it gives a good overview of different approaches, using three different tracking methods in the same experiment. Also HERA-B was a milestone for the cellular automaton: the concept was further advanced and the results were the inspiration for `ForwardTracking` as well as for the vertex detector track reconstruction in Belle II. Finally the Belle II track reconstruction for the Silicon Vertex Detector (SVD) is discussed as its software grew in parallel to `ForwardTracking` and utilizes the same basic algorithms.

3.1. The previous track reconstruction for the FTD: SiliconTracking

The track reconstruction in the International Large Detector as it was used for the Letter of Intent[1] [28] was done in two separate modules [49]. The tracks in the Time Projection Chamber (TPC) were found by the `LEPTrackingProcessor`[2] and

[1] The Letter of Intent (LoI) was the first full document on plans and studies of the ILD detector concept.

[2] As the name suggests, it contained mainly algorithms still from the Large Electron Positron Collider (LEP). The main job of the `LEPTrackingProcessor` was to invoke the LEP algorithms, which were in the form of Fortran routines, with C++ wrappers.

the tracks through the silicon detectors were reconstructed by `SiliconTracking`. Afterwards the results of these were combined by `FullLDCTracking`: the tracks going through silicon detectors and TPC were matched, remaining hits were picked up and finally the tracks were fitted and saved.

`FullLDCTracking` is still operating[3], but `LEPTracking` has meanwhile been replaced by `Clupatra` (tracking for the TPC by cluster pattern recognition). `SiliconTracking`, as `FullLDCTracking`, has been updated to `SiliconTracking_MarlinTrk` (it will be still referred as `SiliconTracking` here for the shorter form) and is now purely based on C++ and utilizes the modern Kalman filter `KalTest`.

`SiliconTracking` is reconstructing tracks through the vertex detector VTX, the Silicon Inner Tracker SIT, the Silicon External Tracker SET and the Forward Tracking Detector FTD. The description here concentrates on the tracking algorithms for the FTD, as this is what is needed to be compared to the new software for the FTD track reconstruction. While `ForwardTracking` now reconstructs tracks through the FTD in parallel to `SiliconTracking`, the tracks through the other silicon tracking detectors are still solely searched for by `SiliconTracking`.

`SiliconTracking` is mainly programmed in the procedural paradigm. It is a class itself, because all modules in Marlin need to be derived from the base class `Processor`, so they can be invoked by Marlin, but besides this, `SiliconTracking` only takes advantage of the object oriented paradigm for fitting methods (e.g. Kalman- and Helix fit). The other algorithms, mainly the pattern recognition, are directly contained within the processor, which has the disadvantage of a lower maintainability and flexibility, and makes understanding the code harder.

The algorithm in `SiliconTracking` used for track finding is a triplet search: hit triplets from different combinations of layers in the FTD are checked with a quick helix fit. If several criteria for the helix fit are fulfilled (like χ^2 being below an upper limit or d_0 and z_0 being low, i.e. the helix being close to the IP) the helix is extrapolated to other layers to pick up further hits if they are close enough to the intersection point of the helix and the layer. This rather simple approach is stable and fast for easy situations, but when there are additional hits, e.g. from higher background, the combinatorics grow very fast, i.e. the number of possible triplets rises considerably.

SiliconTracking basically does a good job and is able to reconstruct the tracks through the FTD to a good degree[4], but has deficits in maintainability, readability, flexibility and background handling. For those reasons the standalone track reconstruction for the FTD `ForwardTracking` was envisioned.

[3] although it was updated by Steve Aplin to `FullLDCTracking_MarlinTrk`

[4] At the beginning of my work the efficiencies of `SiliconTracking` ranged around 60% for p_T >1 GeV, which is considered to be very low. This was one of the incentives to replace it with `ForwardTracking`. Meanwhile bugs and reasons for inefficiencies have been found and removed, giving `SiliconTracking` quite reasonable efficiencies.

3.2. Track reconstruction in CMS

The Compact Muon Solenoid detector (CMS) is located at the Large Hadron Collider and is together with the ATLAS detector, the main experiment where the search for the Higgs boson is performed. One of its main features is the tracking section of the detector, which is like the SiD completely silicon based. A combination of pixel, strip and back to back strip sensors are used to cover the region between the beam pipe and the calorimeters.

The track reconstruction for CMS is done in multiple steps [8, 30]. First, only the pixel hits, which are closest to the IP, are used to find vertices. This is done by first searching for hit pairs [15]: different sets of two layers are considered[5]. For every hit on the outer layer, a possible area on the inner layer, where compatible hits could be, is calculated. This area is determined based on kinematic constraints such as the transversal momentum and limitations on vertex positions. For every hit within the allowed region a hit pair is stored.

A third hit is added from a third layer, which is given for each two-layer combination. On this layer again constraints on the possible area of hits are made. This approach resembles the one used in ForwardTracking as the first constraints are already made for pairs of hits and for triplets a next set of constraints is introduced and used: in every stage wrong combinations get sorted out as fast as possible, instead of allowing them to pollute the next stage.

The triplets are enough to estimate helix parameters, which then get extrapolated and vertices are reconstructed. These so called pixel vertices play an important role, as from there the more general tracking in the CMS detector starts. This concept could also be implemented for the ILD detector track reconstruction in a slightly different way: vertices from the best tracks could be reconstructed and used to sort out ghost tracks by dismissing tracks that do not come close enough to one of the vertices. With the remaining tracks the final vertices could be reconstructed. The benefit would be that once the vertices are known they can be used as a further constraint on tracks. If for example tracks with only 3 hits are reconstructed, the ghost rate will be considerably high. But constraining these 3-hit-tracks to those, which can be extrapolated to a known vertex could greatly reduce the ghosts.

The general tracking in the CMS that follows utilizes the Combinatorial Kalman filter. The doublets and triplets are turned into track seeds if they correspond to a vertex and then are extrapolated outwards. This is done with the Combinatorial Kalman filter, meaning that the tracks are extrapolated and updated as described in Subsection 2.3.2, but in this version the hit that is going to be added to the fit on the next layer is not known previously. Hits close to the extrapolation are searched and for every hit within a certain range a copied version of the track is created and the hit gets added. So after a layer has been processed, for every compatible hit there is a new version of the track, including the track without any added hit to take into account the possibility that a layer was missed. Tracks are collected once

[5] including a certain redundancy in order to ensure sufficient efficiency in case of hardware failures.

they reach the outermost layer. In order to prevent the number of tracks to grow exponentially in every step the number of tracks is decreased by applying quality cuts. Ambiguities of tracks are resolved at the end by taking the tracks with the best quality (length and χ^2) and discarding incompatible ones.

This approach could be implemented for the track reconstruction in the Forward Region of the ILD detector as well. A problem could be the seeding, as the set up is slightly different. In CMS the vertex detector covers all possible directions of the track, if it afterwards traverses the outer silicon sensors. For the FTD this is not given. Not only will the VTX not be hit in most cases, but also the innermost FTD layers are skipped frequently by tracks (see Section 2.2). This means that one cannot use the inner layers for seeding, at least not exclusively. Also the outer layers and the middle ones are not always hit. If such an approach should be used, combinations of different layers all over the FTD need to be taken into account for seeding. This however has the problem that over larger distances the kinematic region of allowed hit pairs and triplets grows because of the high distances and more material effects. In the end this is the approach that `SiliconTracking` uses, without using a combinatorial Kalman filter, but a Helix extrapolation. Also the triplet building is done without the cuts in the hit-pairs. It would be interesting to modify `SiliconTracking` in this way and check out the gain.

3.3. Track reconstruction in HERA-B

HERA-B was a fixed target experiment originally built to study CP-violation in the decays of B mesons into $J/\Psi\, K_0$. But with the rise of the B-factories at KEK and SLAC and due to construction delays, it was finally decided to use HERA-B to study heavy flavor physics. It implemented a new version of fixed target experiments, where protons with 920 GeV were collided with a wire target which was moved into the halo of the proton beam, yielding high collision rates.

Track reconstruction in HERA-B makes for a good discussion of different tracking algorithms, as three different packages where used in parallel to reconstruct tracks: a local approach, a global and a semi-global one. As mentioned before, it is always useful to compare different algorithms under the same circumstances.

3.3.1. The local approach RANGER

The package `RANGER` [42] uses a local pattern recognition technique for track reconstruction. First, seeds are found and then extrapolated with a Kalman filter. As with most local approaches the seeding step is crucial. If the number of seeds is too high, the computing intensive Kalman filter will exhaust computing time[6]. Different cut-offs can be introduced early on, but this can of course cost efficiency. The seeds are built at the beginning of the tracker where the magnetic field is low and the

[6]This is the reason why `SiliconTracking` runs with a faster helix fit as standard and only uses the Kalman filter in the finals fits.

paths therefore almost straight. Triplets are used as seeds and are constructed by taking all possible doublets and combining them pairwise if they share a hit and have the same direction within a certain allowed deviation. To prevent an excess of triplets, if two or more triplets too close to each other are found, they get clustered. The seeds then get extrapolated to farther detector surfaces. In order to make the calculations faster the detector is divided into "domains" and a look-up table provides information on where to extrapolate next.

The track candidates (or the seeds at the beginning) can either have the flag "alive", "mature" or "dead". All candidates start with the flag "alive". In a recursive method, the track candidates get extrapolated to the allowed domains, where hits within a certain range to the prediction get picked up and a new branch of the track candidate containing the hit is created. This approach is the already mentioned Combinatorial Kalman filter, which was named "Concurrent Track Evolution" [43] at the time. This way the growth of the number of branches can be very fast, thus it is necessary to discard wrong ones as fast as possible. For this, only hits are added that add a χ^2 below a certain limit to the fit. And secondly, after each evolution step the quality[7] of the branches gets assessed and branches with a low quality get sorted out. As a further mechanism, every track candidate is only allowed to have 5 branches in each step at most. If there are more, only the best ones are kept.

Once there are no further layers to propagate to a track is labeled as "mature" if it has enough hits, or "dead" if not. When there are no tracks left "alive" the procedure is done and the "mature" tracks are collected and saved.

3.3.2. The global approach TEMA

As mentioned before, global approaches take into account all measurements at the same time. A popular one, as used in the package TEMA [52], is the Hough transformation as mentioned in Subsection 2.1.1. The hits are mapped to the parameter space, where via a binning method maxima are searched for that represent multiple measurements coming from one track. There are however strong limitations that cause Hough transformations to be less useful than other methods and are the reason why they have not been considered for `ForwardTracking`:

- For a helix 5 parameters need to be estimated, leading to a 5-dimensional parameter space. But as the computation time is proportional to n^D, with n being the number of hits and D the dimensions, dimensions above 2 quickly need unrealistic huge computation power. So the models are usually reduced: instead of three dimensional helices, circles in two dimensions are searched or the tracks are estimated as straight lines, losing of course important information.

[7] The quality was calculated based on the number of hits in the track, number of missed layers and the χ^2 value.

- But even if a 5 dimensional Hough transformation were practical, the material effects cause deviations from the track model. This is why the so successful Kalman filter needs a different track state at each measurement site. As soon as material effects come into play, the maxima in the parameter space get smeared out and become harder to detect and start overlapping with other maxima.

- Also the resolution (here the ability to distinguish between the parameters of two different tracks) depends on the binning. The bigger the bins, the easier it happens that two different tracks occupy the same bin in parameter space. This can be avoided by making the bins smaller, but that on the other hand raises the computing time again. Infinitely thin binning would take an infinite time to process.

The above reasons suggest, why the Hough transformation needs to be used carefully, as with higher occupancy or more material effects it can quickly become very inefficient.

3.3.3. The semi-global approach CATS

The CATS [17] package successfully utilizes a combination of a cellular automaton and a Kalman filter for track reconstruction. The Cellular Automaton as described earlier is a semi-global method and brings different advantages over the methods of the other two packages TEMA and RANGER, mostly time and efficiency.

The CA version of CATS works in principle like the cellular automaton described in Subsection 2.3.1, but is (as it was one of the first implementations) a bit simpler. The cells in this scenario are the space points and there are no longer segments involved like in `ForwardTracking`. Two cells are compatible if their χ^2-distance is low enough.

After the cellular automaton has collected the track candidates, a Kalman filter propagates the tracks to further layers, picking up hits and finally refitting the tracks with outlier removal. Once this is done, a χ^2-cut is applied and the remaining tracks are given to an elastic neural net for ambiguity resolving.

This general approach is also the basis for the Forward Tracking in the ILD detector as well as the tracking in the Belle II SVD, although they all differ by the specific implementations, specializations and adaptations due to the different detector setups.

3.3.4. Comparison of the 3 approaches in HERA-B

In [17] the approaches were compared for a $J/\Psi \rightarrow \mu^+\mu^-$ decay plus two superimposed inelastic events as background. The comparison showed that the global approach TEMA acquired the lowest efficiencies, followed by RANGER and then CATS. CATS also ranked first for the lowest ghost rate. Additionally a comparison

of RANGER and CATS was made on the dependence on superimposed inelastic interactions. It showed that CATS keeps its superiority with higher background, but of course loses efficiency and increases ghost rate, as is expected with addition of background. Also the computing time of CATS is faster than RANGER for higher backgrounds (compare to the results of SiliconTracking vs. ForwardTracking in 5).

The results show that the race between a combinatorial Kalman filter and a cellular automaton + Kalman filter is a close one and that the cellular automaton proves to be a viable tool in track reconstruction.

3.4. Belle II track reconstruction

Belle II is a detector for the SuperKEKB e^-e^+-collider in Japan, which currently is under construction, as is the software it will use. It is a successor of the Belle detector, which was used to determine the properties of the CKM matrix (as mentioned in Section 1.2). While the results from Belle are in compliance with the Standard Model, Belle II aims to find small violations in order to search for "beyond the standard model physics".

Outside the pixel vertex detector of the Belle II detector is the silicon vertex detector (SVD), which not only provides high precision hits for tracks extrapolated in from the outer detectors, but also has the goal to find tracks that do not reach outer detectors at all. Tracks with a low transversal momentum (below about 100 MeV) will stay contained within the SVD and loop there, or if they reach the surrounding Central Drift Chamber they do not cause enough hits there in order to be found. For the reconstruction of these tracks a cellular automaton approach has been developed by Jakob Lettenbichler and is currently under refinement and testing. Jakob Lettenbichler deserves a special mention, as we both worked on the cellular automaton and shared and discussed many ideas. For the details on his reconstruction I recommend reading [39]. In principle we both base our programs on the same algorithm combination: cellular automaton, Kalman filter and Hopfield neural network.

The main differences in our works are the technical approaches and the environments. As the main excellence of SuperKEKB lies in high luminosity and not energy, the low momentum loopers are more important than for the ILD detector. On the other hand high transversal momentum tracks can be extrapolated in from the outside detectors and do not need to be necessarily found in the SVD. The reconstruction in the outer detectors will even happen first, and the hits, picked up from the high p_T tracks, could[8] be subtracted from the hit collection which is used by the SVD reconstruction algorithm. This is a method that could be used for the intermediate region in the ILD, where tracks hit the inner FTD disks and also cause enough hits in the TPC in order to reconstruct the track there. As in the FTD only few layers are hit, but many in the TPC, the TPC has an advantage in reconstruction in the intermediate region. Extrapolating the tracks from the TPC

[8]This is work in progress and under current discussion.

to the FTD and picking up hits, including their removal from the hit collection, could ease the combinatorial situation for `ForwardTracking`.

For tracking in the FTD finding the high p_T tracks is way more important, as for smaller θ (angle of track momentum to the beam pipe) these tracks cannot be found by the TPC and extrapolated, thus they must be found by the FTD tracking alone. Losing those tracks would mean a high loss of energy and momentum information at the vertex.

The backgrounds and signals are different as well. While in the high energy environment of the ILD jets[9] and electron-positron pairs from photon conversion cause background, in Belle II other effects, such as the Touschek-effect, [10] play a more important role.

Other differences concern the environment. The tools that are available in each framework differ, for example ILD uses the Kalman filter package `KalTest`, while Belle II utilizes the package `Genfit`. The structure of the overall framework and its demands are also of course different.

In the implementations there are further major differences: While the packages KiTrack, KiTrackMarlin and ForwardTracking use a highly object oriented approach, the tracking for the Belle II SVD is more procedural. Another example is the concept of segments of different lengths that I use in the cellular automaton; in Belle II, on the other hand, there is a stronger focus on sector specific cut-offs.

[9] The jets themselves are of course not background as they come from the events to be reconstructed, but their dense tracks can cause a high rate of ghost hits on the silicon strip detectors (see Subsection 1.5.2)

[10] single particles leaving the beam and interacting with the detector

4. Implementation

The algorithms that were written for track reconstruction in the forward region of the ILD detector are separated into 3 distinct packages:

- **KiTrack:** Contains the core software with the basic algorithms such as the Hopfield neural network and the cellular automaton. It also includes the abstract base classes to support these algorithms and define hits, tracks etc. This package is not dependent on `Marlin` and can be used in other frameworks as well[1]. For using the package (in `Marlin` or another framework) a package that implements the abstract base classes within the specific framework is necessary. The package depends on `ROOT` for mathematical classes such as vectors.
- **KiTrackMarlin:** Implements the abstract base classes in the framework `Marlin`. It combines the classes from `KiTrack` with the `LCIO`-classes and handles the fitting of tracks via the `MarlinTrk` interfaces. The package depends on the packages `KiTrack`, `Marlin`, `MarlinTrk` for fitting, `MarlinUtil`, `ROOT` and `GSL` for randomization.
- **ForwardTracking:** Contains the actual processors used for track reconstruction and tracking specific analysis of the events in the ILD detector. It depends on the packages `KiTrack`, `KiTrackMarlin`, `Marlin`, `MarlinTrk`, `MarlinUtil`, `ROOT` and `GSL`.

The separation into 3 packages has several reasons:

1. It allows other packages within the `Marlin` framework to use the algorithms of `KiTrack` and `KiTrackMarlin` without creating mutual dependencies. This is especially important if `ForwardTracking` is dependent on other packages. If for example the package `MarlinTrkProcessors` (which contains other processors for tracking in the ILD detector) would contain its core algorithms and `ForwardTracking` would also contain its core algorithms, this would mean that if they want to each use algorithms from the other package, this gives rise to a cyclic dependence. If they both outsource their algorithm classes to distinct packages, this problem is resolved.
2. The core algorithms are well separated from the rest, so changes in the implementation will never affect the basic algorithms and vice versa.

[1] There is a technical dependency on Marlin at the moment, simply for debugging convenience - a few output statements using the debug-output methods of Marlin are still in place. This can however be resolved rather easily.

3. As a part of the AIDA project, the packages are a first implementation of the work package D2.4, which consists of creating a general tracking toolkit. As `KiTrack` is not dependent on ILD or Marlin, it can be more easily included in this toolkit.

All three packages are written in C++ and are part of the current ILD software repositories, which can be viewed at `https://svnsrv.desy.de/viewvc/marlinreco/`. They follow an object oriented design philosophy, thus granting flexibility and possible comparison of different variations. The classes are documented with Doxygen, which allows the generation of HTML files documenting the code and the dependencies of the classes.

4.1. Requirements

There are several requirements that need to be met by the new track reconstruction packages:

- **Flexibility:** Allows for the configuration of the behavior of the algorithms at runtime without modification of the algorithms themselves. This is especially important in order to compare different approaches. One could for example design different additional algorithms for track reconstruction such as Hough transformation, combinatorial Kalman filters or others and directly compare them to and combine them with the current approaches.

- **Background handling:** The more robust (in terms of efficiency) the algorithms are concerning growing background, the bigger the field of problems that can be solved with them. It could for example be that instead of a 500 GeV - 1 TeV ILC, a CLIC with 3 TeV gets built, which has estimated background levels higher than ILC.

- **Speed:** One of the main reasons background can become a problem is the increased calculation time for the higher combinatorics. As the amount of physics events recorded at the ILC will be huge, a fast reconstruction is necessary in order to make them available to physics analysis as fast as possible. One could always increase the computational power, but that adds additional cost to the project.

- **Object orientation:** Although this is no strict requirement for track reconstruction, it gives several benefits, especially in terms of readability, flexibility and maintainability.

4.2. KiTrack

`KiTrack`, as in "Kit for Tracking", is a collection of classes that contain algorithms for track reconstruction and the necessary interfaces. At the moment there are mainly two groups of classes: the cellular automaton and its environment and the subset algorithms.

The task of the subset classes is to resolve ambiguity situations as described in Sub-subsection 2.1.1.3. Such situations are not dependent on tracking, but can happen in almost any subject, thus this can be defined in a very generalized form. For these reasons the subset classes are defined as templates. Templates in C++ [56] are a concept that separates algorithms from the data they process. As an example, this means that a sorting algorithm does not require the complete information of the object being sorted, merely access to a criterion by which it can sort the object.

Likewise the subset problem can be defined independent of tracking: A set consists of elements that can be compatible or incompatible with each other and are able to have different qualities. The task is to find a subset that contains only compatible elements, while maximizing for a specified quality and the size of the subset.

The base class `Subset` defines a few basic methods for this. Derived from it are two other classes `SubsetHopfieldNN` and `SubsetSimple`, which implement the actual algorithms. `SubsetHopfieldNN` is a subset finder using a Hopfield Neural Network as described in Subsection 2.3.3. For this the class `HopfieldNeuralNet`, which contains the core algorithm of a HNN is used. `SubsetSimple` is an implementation of the simple algorithm mentioned in Subsection 2.1.1.3. As a class diagram this gives Figure 4.1.

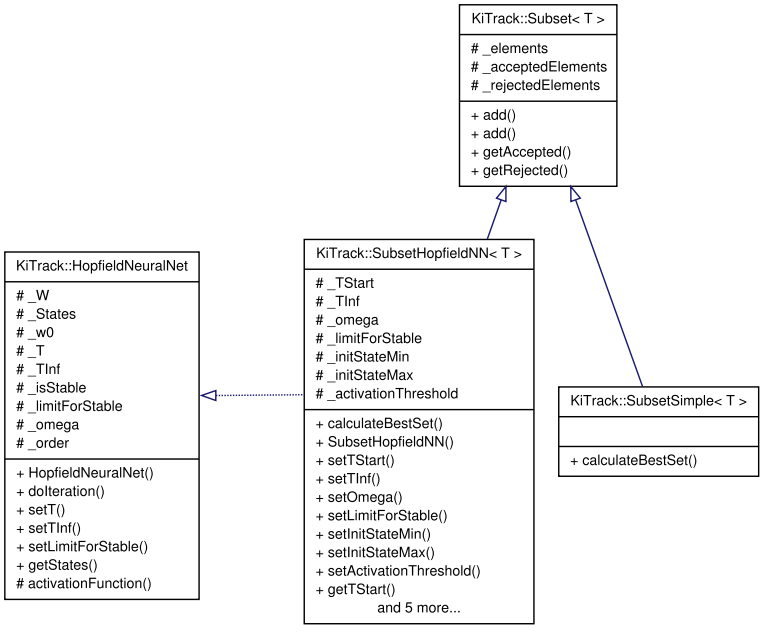


Figure 4.1.: The subset classes

Additional flexibility of these subset classes is provided by the way they determine quality and compatibility of the elements: users can define functors[2] for quality and compatibility and hand them over to the template on creation. This gives the possibility to compare different quality indicators and to adapt the compatibility determining method to the specific needs.

The classes for the cellular automaton are shown in Figures 4.2-4.4. The core algorithms of the cellular automaton are contained in the class `Automaton`. It takes care of the iterations and extending of the segments. It also creates track candidates as its final result[3].

The criteria, which are needed to determine the compatibility of two segments, are stored as pointers to a generic base class `ICriterion`. This is what is called the strategy design pattern [59]: the `Automaton` does not determine the criteria it is going to use, it only has a container for criteria, which are derived from the abstract base class `ICriterion`[4]. `ICriterion` provides an abstract interface defining the methods required by all derived classes. For example a class derived from ICriterion must have a method that calculates the compatibility of two segments. There is a class for every criterion used and all of them are derived from the same `ICriterion` base class. An example can be seen in Figure 4.3.

This leads to an increase in flexibility: the class `Automaton` can then fulfill its requirements without prior knowledge of which criteria will be used. Because of this the behavior of the cellular automaton can be steered at runtime: the user decides which criteria to use.

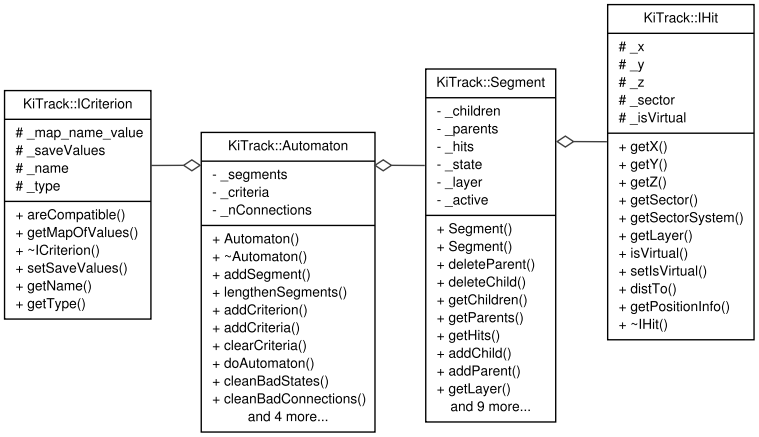


Figure 4.2.: The class `Automaton`

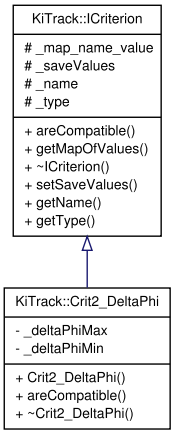


Figure 4.3.: A criterion

The class `SegmentBuilder` is responsible for creating the first 1-hit-segments from the basic hits. The main task is to establish the connections between the segments.

[2] A functor in C++ is a class that is mainly consisting of a single function with specified input and output variable types.

[3] A track candidate there is simply a vector of hits.

[4] The naming convention in the following will be that a class starting with "I" is an interface class.

This is one of the main features of the `Segment` objects: they hold pointers to their inner and outer connecting segments.

The space is divided into discrete sectors. Every hit is determined to lie in only one sector. The method of encoding and decoding the place is flexible (this again follows the strategy pattern). The method of connecting segments again uses a pointer to a base class: `ISectorConnector`. Classes derived from `ISectorConnector` are able to find for any given sector, all the sectors with which a given sector is allowed to connect. In an example for the FTD this could mean that it will connect the hits from FTD layer 6 to those on layer 5, but not those on layer 2.

The hits which need to be checked for compatibility are provided by the classes derived from `ISectorConnector`. Also criteria are provided for the `SegmentBuilder` in the same manner as they are provided for the cellular automaton. Based on the allowed connections and the criteria, the segments are built and connected. In a final step the `SegmentBuilder` creates an `Automaton` object and provides it with the segments.

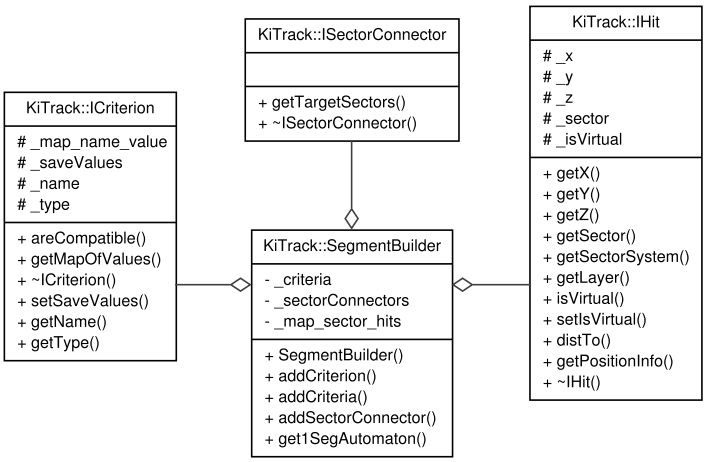


Figure 4.4.: The class `SegmentBuilder`

4.3. KiTrackMarlin

In the `KiTrack` package different abstract base classes are defined, making the algorithms independent of the concrete implementation. `ForwardTracking` on the other hand is responsible for the specific track reconstruction in the forward region of the ILD detector concept and therefore needs to work within the ILD software framework (`Marlin`). For this reason a concrete implementation, which utilizes the generic algorithms described in the `KiTrack` library, is needed. `KiTrackMarlin` is a library which allows for efficient implementation of tracking strategies within the `Marlin` framework (`LCIO`, `Gear`) .

There all the different abstract base classes of `KiTrack` get implemented in a framework specific fashion. For example, `KiTrack` defines the abstract base class `ISectorConnector`, which contains the rules that a sector connector class needs to fulfill so that the `SegmentBuilder` can operate with it. In `KiTrackMarlin` this abstract class is implemented with the class `FTDSectorConnector`.

While `ISectorConnector` has no content concerning the specifics of the underlying sectors, `FTDSectorConnector` is based on the real FTD. It allows connections based on the geometry of the Forward Tracking Detector. The previously abstract "sectors" are now replaced by the concept of real sensors on different layers and modules of the FTD.

For every class that is defined abstractly in `KiTrack` there is an implementation in `KiTrackMarlin`. Additionally there are classes that handle the fitting procedures (Kalman fit and helix fit). In principle these classes could have been included in `ForwardTracking`, but experience has shown that it is good practice to keep the algorithm libraries and the processors separated to prevent unwanted dependencies.

4.4. ForwardTracking

`ForwardTracking` contains all the processors that can be run by `Marlin`. The most important one has the same name as the package and is responsible for reconstructing the tracks in the Forward Tracking Detector of the ILD. It uses the classes from `KiTrack` and `KiTrackMarlin` for the reconstruction.

Here is a short summary on how `ForwardTracking` does track reconstruction:

1. All collections of hits on the FTD are read in.

2. The hits are stored in a map that links the sectors of the hits to the hits themselves. The sectors specify the sensors the hits are on.

3. A safety check is done to ensure no single sector is overflowing with hits. This could give a combinatorial disaster leading to very long calculation times. If a sector is overflowing, it is dropped and the reconstructed tracks are marked as having a bad quality.

4. A virtual hit is added at the position of the IP. It is used by the Cellular Automaton as additional information.

5. Hits on overlapping petals are looked for. If two petals from the FTD overlap, a track may pass through both and thus create two hits in close range. For pattern recognition as it is now, they are not useful (the lever arm is too short for good cut-offs). Such short connections are therefore looked for and stored, but are not dealt with until later the track candidates have been found.

6. The `SegmentBuilder` uses the hits and a vector of criteria. The criteria tell the `SegmentBuilder` when two hits might be part of a possible track. Additionally

an `FTDSectorConnector` is passed that specifies in which sectors of the FTD hit-pairs may be looked for. This reduces computing time, as not all possible hit-pairs need to be checked with the criteria. The `SegmentBuilder` builds the allowed segments from the hits and stores them in a new created `Automaton`.

7. The `Automaton` performs. For every rerun with longer segments it is passed a new vector of criteria suited for the different segment-lengths. As a safety mechanism the number of connections in the automaton is monitored. If an upper limit of the number of connections is exceeded, the Automaton is rerun with tighter criteria. These tighter criteria can be specified in the steering file, the number of steps of tighter criteria can be set flexibly. In case the last set of criteria still yields too many connections, the procedure is stopped. This is a harsh method, but a mere safety mechanism. While it can happen for high backgrounds that the automaton has to be rerun with tighter cuts, a complete drop is highly unlikely, especially for real data. It is nonetheless implemented that way, because it might happen that in the simulation or digitization something goes wrong.

8. After the automaton has performed for 3-hit-segments, the track candidates are read out and further processed.

9. The hits from overlapping petals are added and every possible combination of the track candidate and the suited hits from overlapping petals is created. The version with the best fit result (highest χ^2-probability) is then kept, while the others are discarded.

10. During this procedure cuts can be applied. Tracks with a χ^2-probability below a certain value get discarded.

11. Also a simple helix fit cut is applied. As it is faster than the Kalman fit because it does not take into account any material effects, it is performed previously. It uses currently rather loose cuts in order to only discard tracks that will also give a bad Kalman fit (at the moment the rule is $\frac{\chi^2}{Ndf} < 500$)

12. The ambiguities are resolved: the tracks gathered may not be all compatible with each other (i.e. share hits). This is solved with a best-subset-finder, for instance the Hopfield neural network or an alternative simple algorithm.

13. At the end the tracks are finalized (all necessary information is added to the track objects) and stored in the output collection.

The package `ForwardTracking` also contains different other processors mainly used for tracking analysis and background simulation. They will not be described here, but are documented in the source code with Doxygen.

4.5. Integration in the framework Marlin

In Figure 4.5 the integration of `ForwardTracking` in the chain of simulation and analysis of the ILD detector can be seen.

`Mokka` is propagating the particles generated in the events (by programs such as Whizard) through the detector and creates the raw hit collections. In a real application of the ILD detector this step will be replaced by the experiment itself. The data then comes from the read out electronics on the sensors.

The digitizer `PlanarDigiProcessor` uses the raw hit collections from `Mokka` to simulate the digitization process and create hits that can be used by the track reconstruction software.

The hits on the "false" double-sided silicon strip detectors are not yet ready for track finding as this would need a 3-dimensional space point and not strips (2-dimensional "lines"). The processor `SpacePointBuilder` therefore combines the strip measurements to space points. This is the stage where ghost hits are created.

For the different hit collections, there are multiple reconstruction processors taking care of them. As mentioned before `ForwardTracking` uses all hits on the FTD as input. `SiliconTracking_MarlinTrk` processes them too and other silicon tracker hits as well. Tracks in the TPC are reconstructed by `Clupatra`. When the tracking is done, the different collections need to be combined to a final track collection containing all tracks in the ILD detector.

First the results of `SiliconTracking_MarlinTrk` and `ForwardTracking` are combined via the `TrackSubsetProcessor`, which uses the Hopfield Neural Network from `KiTrack` to get a compatible subset of tracks. This is needed as both processors search for tracks on the FTD, i.e. will have overlapping results. This method makes it possible to also add other tracking processors to search for tracks in the silicon detectors.

`FullLDCTracking_MarlinTrk` is the last tracking step. Here the tracks from the silicon detectors and the TPC are merged and a final track collection is saved.

The tracks are then used to find neutral vertices (`V0Finder`) and kinked tracks from charged particle decays (`KinkFinder`). The resulting information is together with the data from the calorimeters passed to `MarlinPandora`, which does the calorimeter reconstruction and particle flow.

Finally the `VertextFinder` reconstructs the primary and secondary vertices. The results of the processors are saved in the data format `LCIO`, so they can later be accessed by physics analysis.

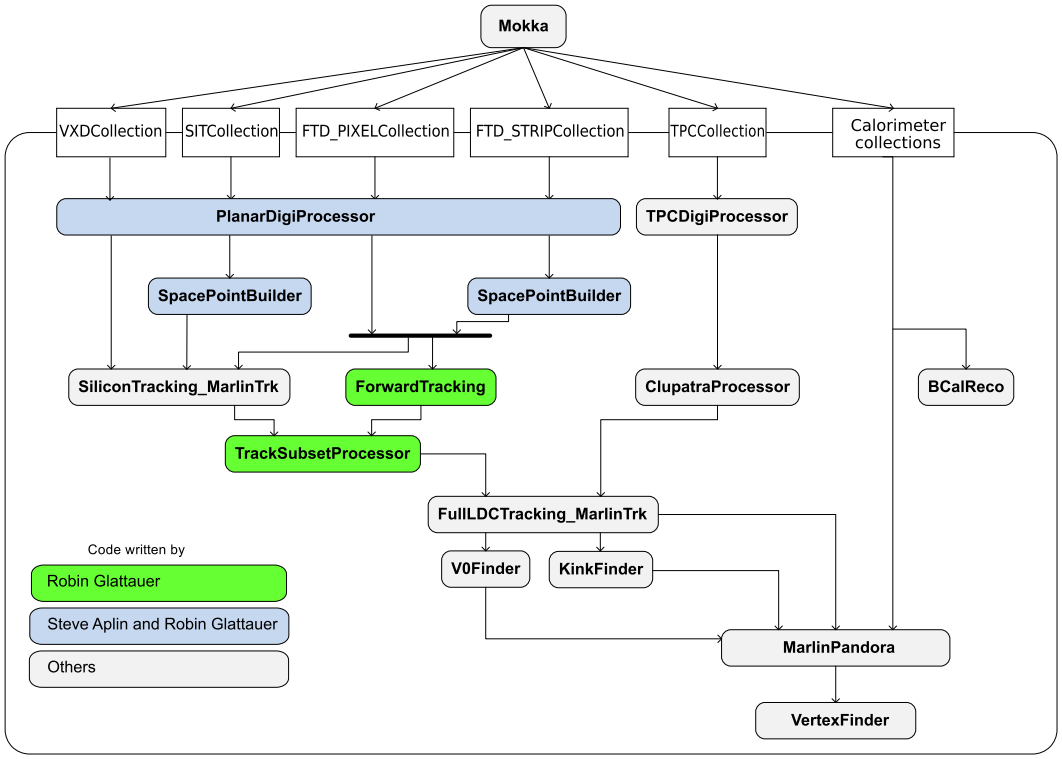


Figure 4.5.: Activity diagram of reconstruction in the `Marlin` framework

5. Results

This chapter presents the results of the new written tracking software. First the benchmark parameters used to measure the success are explained, then different possibilities of tuning the used algorithms to optimize these parameters are introduced. At last the final results are presented and a comparison to `SiliconTracking` is given.

5.1. Important benchmark parameters of track reconstruction

There are different parameters that mark the quality of track reconstruction. The most important ones are

- **Efficiency:** the fraction of true tracks that are successfully reconstructed. Of course this should be as high as possible and is maybe the most important parameter as every true track that is lost is a loss of information.
- **Ghost rate:** the rate of tracks in the collection of reconstructed ones that are combinatorial background. Many ghost tracks can be discarded in further steps, like when linking the tracks to the measurements of the calorimeters, but still ghost rates as low as possible are desired.
- **Speed:** as ten thousands of events need to be reconstructed in order to analyze the physics behind them, slow algorithms delay the time the physics analysis can start. Therefore track reconstruction should be as fast as possible.

The speed was measured by taking the average reconstruction time the processors needed in seconds. This is dependent on the computer hardware and the computers used for future event reconstruction in grids are probably faster than the local machine used by the author. The main concern here is the order of time and the comparison between the different algorithms.

Ghost rate and Efficiency were already defined in Chapter 2 as:

$$efficiency = \frac{true\ tracks\ found}{all\ true\ tracks\ to\ be\ found} \tag{5.1}$$

$$ghost\ rate = \frac{false\ tracks\ found}{all\ found\ tracks} \tag{5.2}$$

There also the importance of defining the methods for determining these values was mentioned. The rules applied here are:

- A reconstructed track is associated with a true track,
  - if the reconstructed track consists to at least 75% of hits from the true track (purity $\geq 75\%$)
  - and more than 50% of the true track hits are in the reconstructed track (completeness $> 50\%$).
- Tracks to be found (the denominator in the efficiency) have only the following cut: The number of space-point-like hits (2-dimensional measurements) must be at least 3 (nHits ≥ 3).
- A ghost track is a track that cannot be associated with a true track.
- A true track is regarded as found if at least one reconstructed track is associated with it. Otherwise it is a lost track.
- The software used for track reconstruction with `ForwardTracking`, `SiliconTracking` and the `TrackSubsetProcessor` is the same version as used for the DBD.
- The steering of the three processors was identical to the one in the standard steering file "bbudsc_3evt_stdreco.xml" from the `ILDConfig` package, except for tuning the parameters.
- The detector model used by `Mokka` was ILD_o1_v05, also used for the DBD.
- The simulated events come from an "$e^-e^+ \rightarrow$ WW $\rightarrow$ 4quarks" event generator file as the hadronic jets cause higher occupancies.
- Only hits in the FTD were processed by the digitizers. The comparison gives no information on reconstruction in any other detector. Therefore the efficiency, time and ghost rate of `SiliconTracking` are not necessarily the same as in its other track reconstruction parts, i.e. the reconstruction in the barrel region[1].

[1] However similarities are to be expected as the basic methods used in both directions are very much alike

5.2. Tuning of parameters

For each stage in the track reconstruction some tuning is necessary, because the algorithms depend on multiple parameters determining their behavior. For every step of reconstruction an example of a tunable parameter is given and others are mentioned.

5.2.1. Tuning the cellular automaton

The cellular automaton uses multiple criteria in every reiteration. Those criteria all define a lower and an upper cut, which can be tuned.

For example, a criterion that can be applied to two hits, is the difference of their distance from the z-axis[2]. In Figure 5.1 the distribution of this value for hit-pairs of true Monte Carlo tracks in a collection of 10000 events is shown.

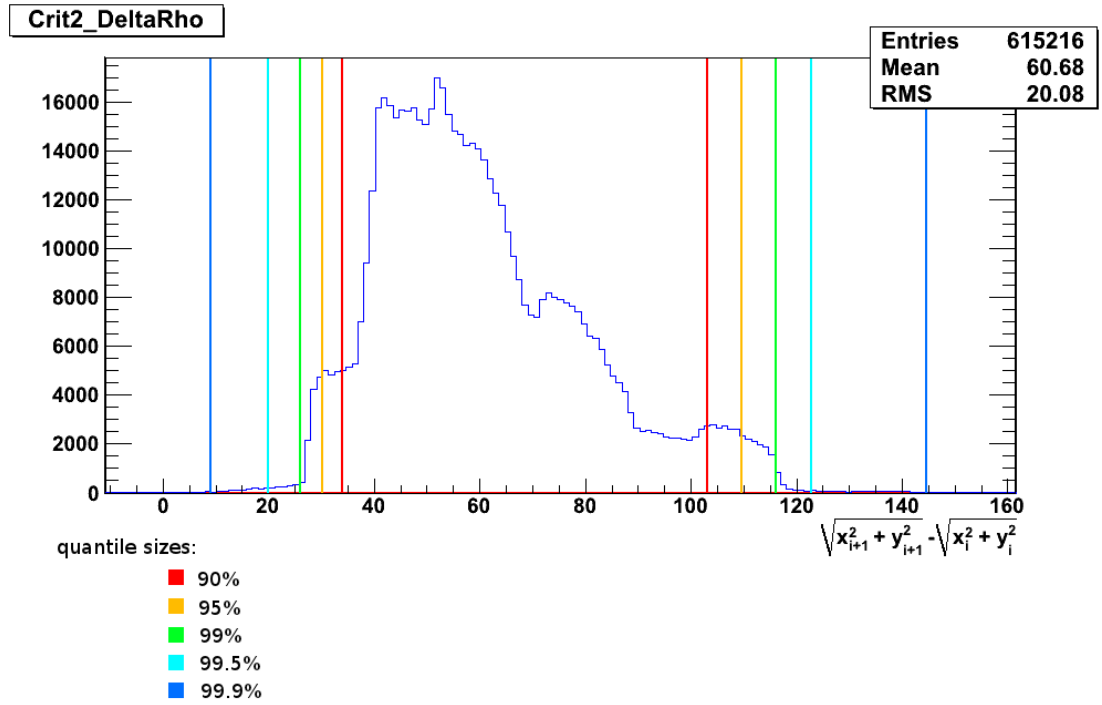


Figure 5.1.: Histogram of a cellular automaton criterion

If the Cellular Automaton shall keep these true tracks, it must set its lower and upper cut in a way that as many true tracks as possible are in between. The figure also shows 5 different cuts that can be made. In this instance the cuts are all set symmetric, therefore as much is cut away for low as for high values.

Making the quantiles as big as possible to include 100% may seem reasonable, but multiple scattering and energy loss cause the tracks to deviate from perfect

[2]This is mainly a useful criterion, as it can be calculated fast and tracks that don't curl back between the hits, will always give a positive value.

helices. As these processes are of a stochastic nature any physical possible value will appear for a large enough sample. Also the cuts define the selection power of the cellular automaton; broadening them too much takes aways much of its capability to reduce combinatorics. One can therefore never keep all tracks without accepting more combinatorial background and therefore slowing down the reconstruction and boosting the ghost rate.

One way to decide which cuts to make, is to use a fixed quantile size for all criteria. This approach, however, has limitations, as different criteria have different characteristics. Sometimes it does not make sense to use a cut on one side at all. For example there is a criterion (`DistToExtrapolation`) that measures the distance of a hit to its extrapolation based on other hits. The lowest possible value 0 (i.e. a perfect prediction of the hit-position) is ideal and any cuts on the left side would be counterproductive. A solution for this can be to determine a right-left distribution of the quantile for every criterion, but there is also the additional point that different criteria have also a different usefulness: some are faster, some sort out more background, some have very high efficiencies.

The current approach is to base the decisions on 99.5% quantiles[3] and to refine these cuts by manual adjustment. The probably ideal solution would be to exactly define a coefficient of what is desired, which needs to include parameters such as computing time, efficiency and ghost rates and then to vary all the different cuts stepwise to find a maximum of this parameter. This however needs a careful approach and a lot of time.

Finally it is noteworthy to point out the visibility of the underlying structure in the histogram. The distribution of values already gives a hint that multiple distributions are superimposed here, namely 2-hit combinations from different layers all over the FTD. Hit-pairs from layer 5 and 6 for example have a different distribution than those from layer 1 and 2, because first, the distances are larger on the outside and second: the further away from the vertex, the more tracks come curling back. In Figure 5.2 the same histogram is plotted, but only hit-pairs from layer 5 and 6 were taken into account. One can see, how the cuts could be made tighter by making them sector specific. This is for example done in [38] and could be implemented for tracking in the forward region of the ILD as well in the future.

[3] with hand-tuned right-left distributions

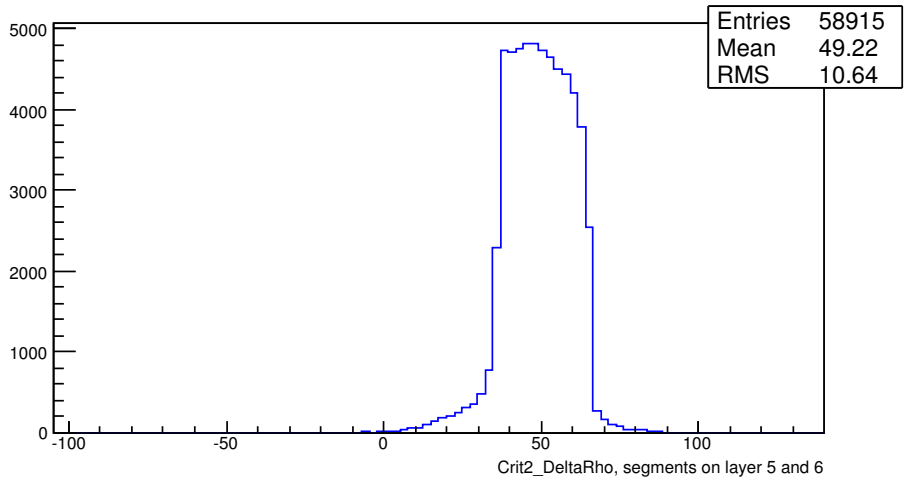


Figure 5.2.: Criterion only for 2 hits on layer 5 and 6

5.2.2. Tuning the Kalman filter cuts

As mentioned in Subsection 2.3.2, χ^2 is the sum of the weighted quadratic forms of the residuals. For errors that are independent and Gaussian χ^2 follows the so-called χ^2-distribution. Based on this the χ^2-probability can be calculated: the probability that for a number of degrees of freedom a random χ^2 has a certain value or higher.

For a correct track model and if material effects are taken into account correctly, the χ^2-probability should be a flat distribution. In Figure 5.3 this is plotted for true tracks in the FTD from 10000 simulated events fitted with `KalTest`. The result is rather flat as it should be, the peak on the left is a common problem with the χ^2 test, when errors are not perfectly Gaussian (see [41, page 606]). Especially low momentum tracks are subject to this problem. In comparison, in Figure 5.4 the distribution for high p_T tracks can be seen, the tail on the left is far less dominant.

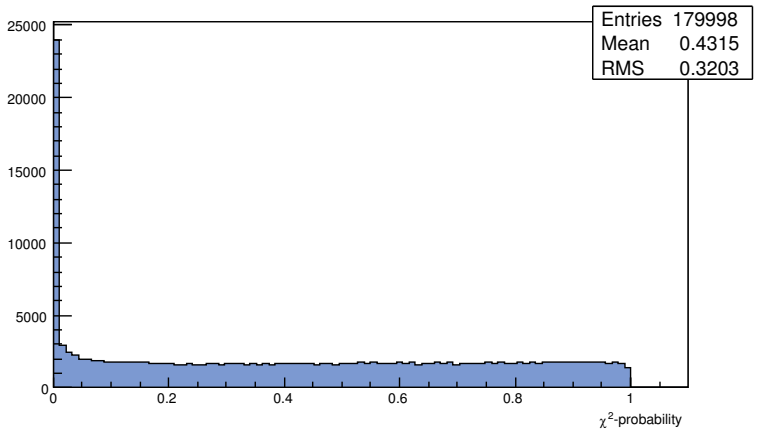


Figure 5.3.: χ^2-probability distribution of true tracks in 10000 WW events

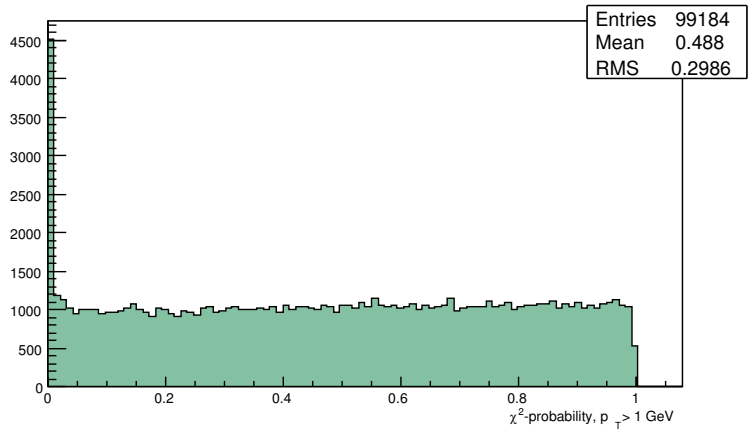


Figure 5.4.: χ^2-probability distribution of true tracks with $p_T > 1\,GeV$ in 10000 WW events

As a measure for the goodness of fit the χ^2-probability can be useful. A flat distribution would guarantee that if one applies cuts for tracks with a very low χ^2-probability, only tiny fractions of true tracks are dismissed. The peak on the left however shows that in the given situation a higher fraction of true tracks would be cut away. Also "small χ^2 probability does not necessarily imply a bad estimation of the parameters, hence special care is required when a χ^2 cut is to be used to eliminate improperly reconstructed tracks" [41].

In Figure 5.5 tuning of the χ^2-probability-cut in `ForwardTracking` is shown. Efficiency and ghost rate are plotted over the value of the cut. Also the values that efficiency and ghost rate reach when the cut is not used at all is depicted. For this plot the ambiguity resolving was switched off. The tracks are the result of the cellular automaton with the addition of suited hits from overlapping petals.

One can see that the cut can lead to a reduction of ghost rate, for example a cut at 0.001 reduces the ghost rate by a factor of 2, while the efficiency is only reduced by a few percent. Note however that for a perfect χ^2-probability distribution the reduction should only be in the order of 0.001 – this is the influence of the low-probability peak.

At the moment this cut is not used in `ForwardTracking` but at a later stage in `FullLDCTracking` (with the value 0.001), because at later stages more information is available and further hits might have been added to the tracks or others excluded.

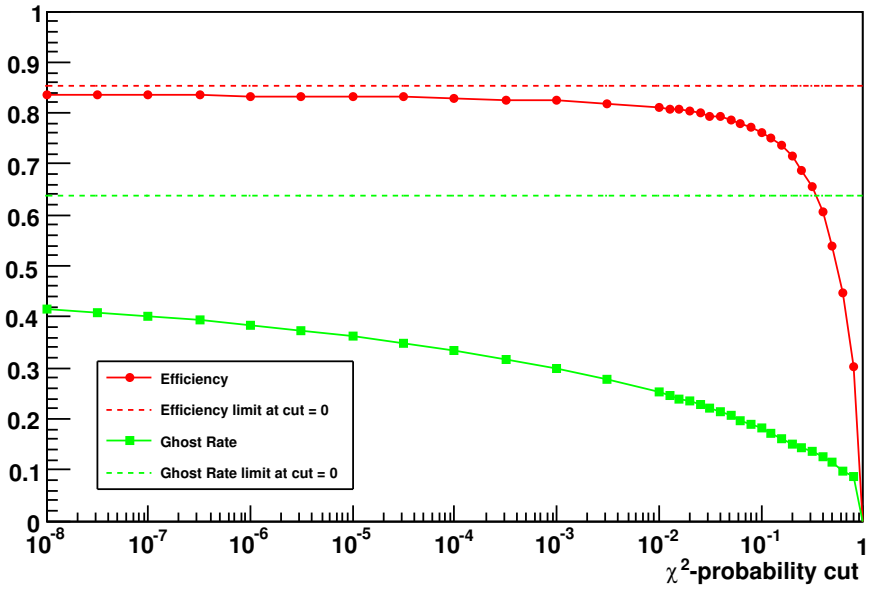


Figure 5.5.: Tuning of the χ^2-probability cut

5.2.3. Tuning the Hopfield neural network

The Hopfield neural network and its capabilities for ambiguity resolving were described in Subsection 2.3.3. The algorithm is used via the class `SubsetHopfieldNN`. It possesses a multitude of tunable parameters as listed in Table 5.1. The most influential parameters are ω and the quality indicator.

Different quality indicators have been tried, two are presented here: the χ^2-probability (p) and a modified version[4] (p'). Quality indicators for the Hopfield neural network have to be in the range between 0 and 1. The modified version of the χ^2-probability distinguishes between tracks with 3 hits and tracks with 4 hits or more, thus granting a bonus for longer tracks:

$$p' = \begin{cases} \frac{p}{2} & \text{3 hits or less} \\ 0.5 + \frac{p}{2} & \text{4 hits or more} \end{cases} \tag{5.3}$$

For tuning the Hopfield neural network, at first ω was varied over its whole range from 0 to 1. Figures 5.6-5.9 show the efficiency and ghost rate for different values of ω and the two mentioned quality indicators. The Hopfield network works with the tracks reconstructed by the cellular automaton. Each measurement is based on 100 event samples with additional background (half of LoI background, see Subsubsection 5.3.0.1). No χ^2-probability cut was applied.

[4] which does no longer correspond to an actual probability

parameter	description
T_0	The initial temperature
T_∞	The temperature limit
ω	controls the influence of the quality of the tracks, ranges from 0 (no influence) to 1 (full influence)
activation threshold	controls which elements are in the final set
QI	Quality Indicator

Table 5.1.: Tunable parameters of the Hopfield neural network

Additionally the results for no ambiguity resolving[5] and the results of a simple algorithm[6] (`SubsetSimple`) for the same quality indicator are shown.

The conclusions one can draw from the plots are:

- For both `SubsetHopfieldNN` and `SubsetSimple` the modified quality indicator gives better results - a longer track with a good χ^2-probability is more likely to be a true track, than a shorter one.
- Concerning efficiency the simple algorithm performs a bit better than the Hopfield network, which might be due to the reason that most incompatibilities come from clones, while the Hopfield network is especially useful for highly entangled situations. If for example the incompatibilities can be separated into distinct mutual exclusive groups, as would be the case for incompatibilities coming only from clones, the simple algorithm is by default the best one: if the groups are mutually exclusive and there are no cross incompatibilities between the groups, the solution contains exactly one element from each group. There cannot be more elements coming from a single group as they are mutually exclusive within the group and therefore are not compatible. The remaining task is to select the one best element from each group. This is exactly what the simple algorithm does: it saves the element with highest quality and deletes all incompatible ones. Therefore for the case of distinct groups, it will take the element with highest quality from each group and discard the whole rest of the group. It will therefore maximize the sum of quality indicators in the final set.
- The ghost rate suppression is best done with the Hopfield network at $\omega = 1$.

[5] which however also results in a clone rate above 0

[6] The simple algorithm was described in 2.1.1.3. The track with the highest quality in the set is saved and all incompatible ones are discarded from the set. This is repeated until the set is empty.

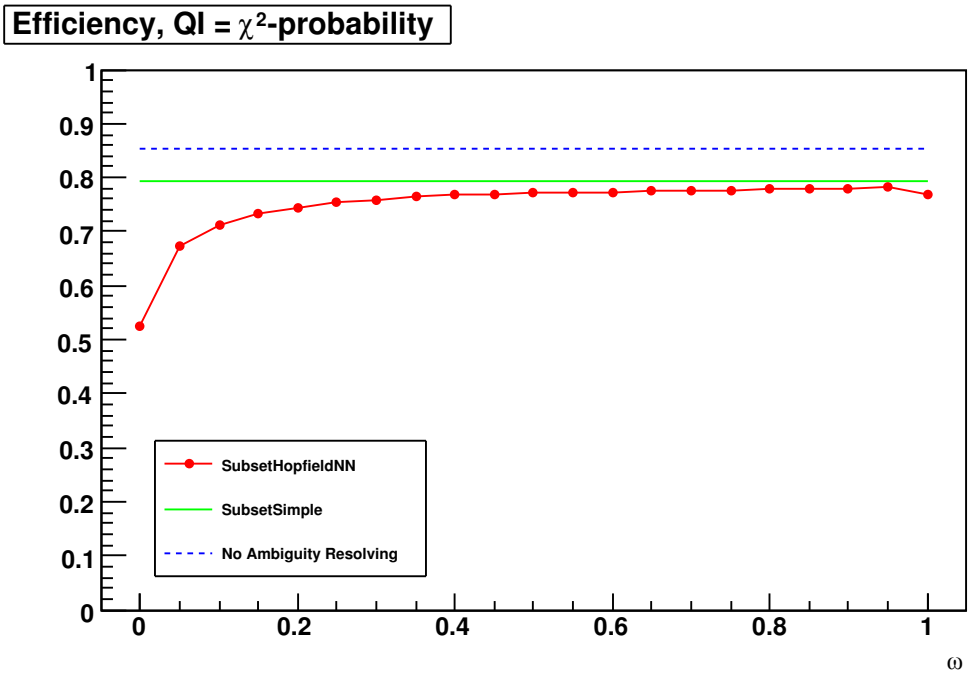


Figure 5.6.: Efficiency of subset algorithms, χ^2-probability is used as QI

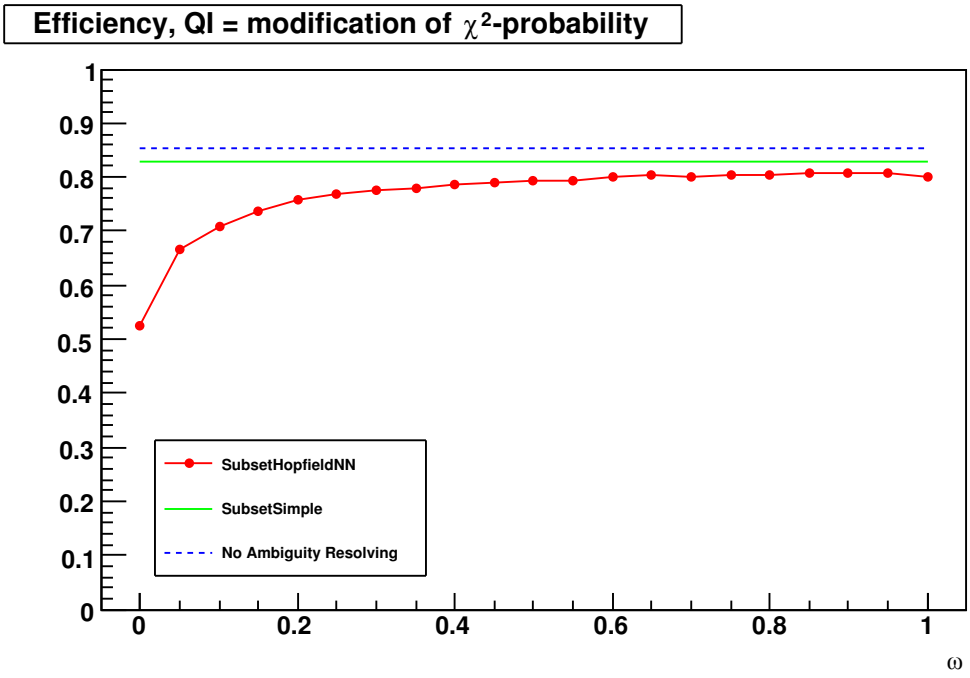


Figure 5.7.: Efficiency of subset algorithms, a special QI is used based on the χ^2-probability

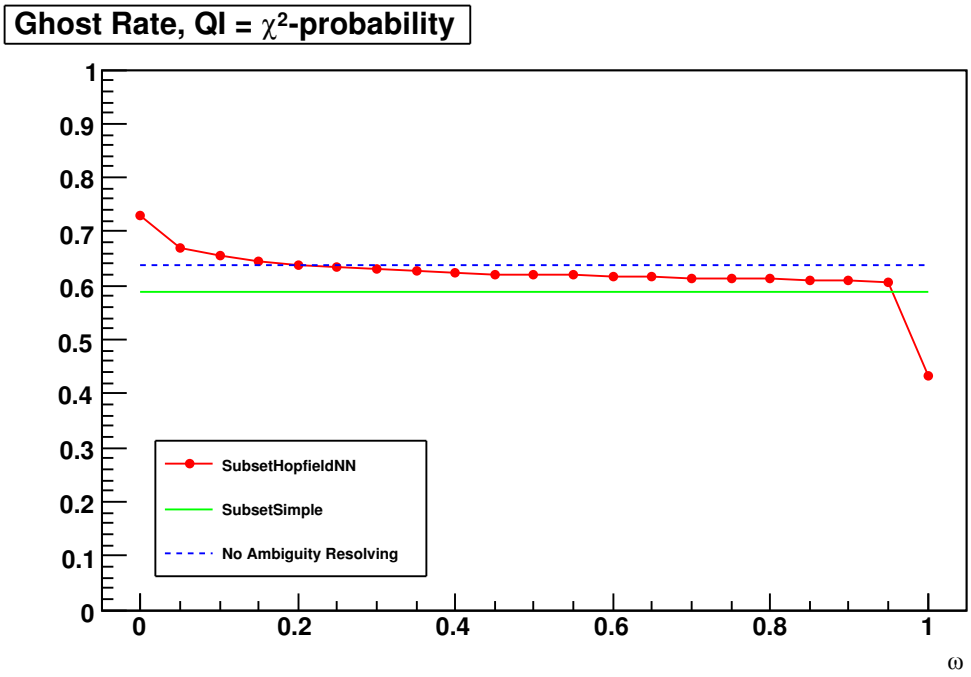


Figure 5.8.: Ghost rate of subset algorithms, χ^2-probability is used as QI

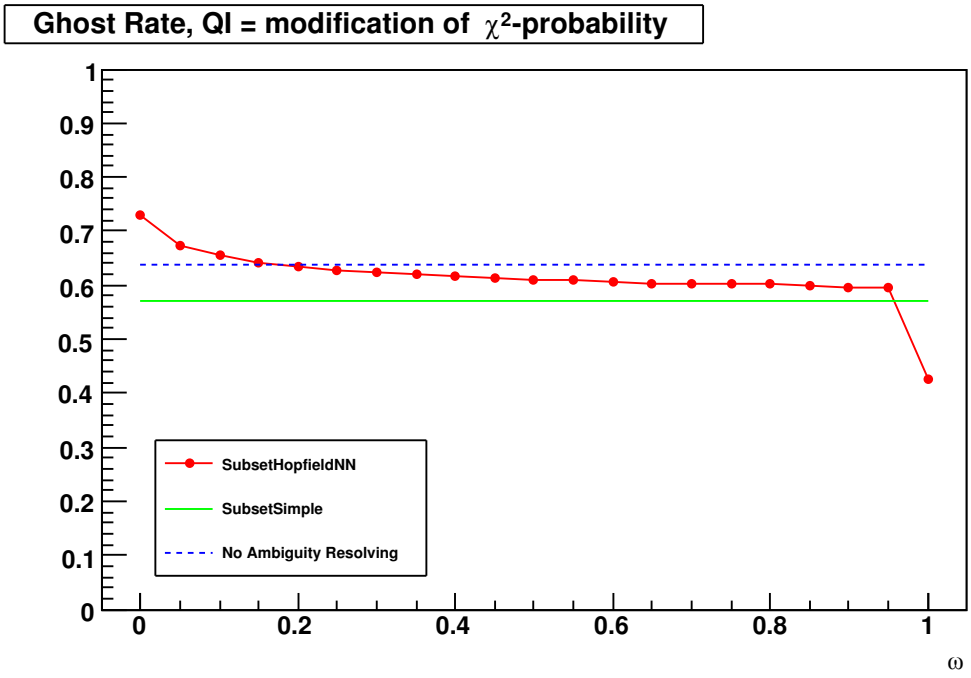


Figure 5.9.: Ghost rate of subset algorithms, a special QI is used based on the χ^2-probability

The other parameters have less effect on efficiency and ghost rate, except if they are far from the optimal value. Investigation has shown that the values recommended in [22] (e.g. $T_\infty = 0.1$, $T_0 = 2.1$) are well suited. For example in Figure 5.10 the modified QI and an ω of 0.9 (which maximizes efficiency) have been used and T_∞ was tuned around the recommended value of 0.1. In contrast to the previous figures here the vertical range is smaller. One can see that this parameter has almost no influence, with 0.08 giving slightly better result.

As the optimal values of the parameters are not independent of each other a mechanism for tuning would probably still improve the results. Maybe the most efficient way would be to try additional quality indicators. But as for now the results are slightly worse than those of the `SubsetSimple`, which is simpler and faster, the `SubsetSimple` is used in standard track reconstruction. The race is however very close and it could well be that with different background or in situations giving higher entanglement the Hopfield Neural Network gives better results than the simple algorithm.

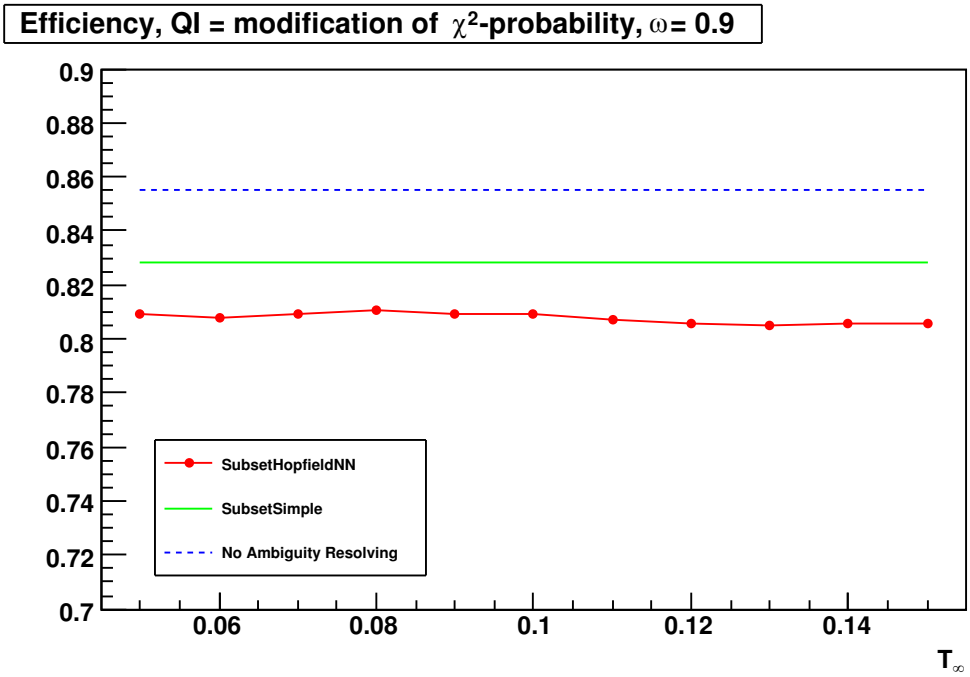


Figure 5.10.: Efficiency of subset algorithms, a special QI is used based on the χ^2-probability and $\omega = 0.9$

5.3. Comparison with SiliconTracking_MarlinTrk

As mentioned in Section 3.1 `SiliconTracking_MarlinTrk` is the processor that has so far been used for track reconstruction for the FTD. `SiliconTracking_MarlinTrk` is already the updated an enhanced version of the older `SiliconTracking`. As the older version is not used anymore the shorter name `SiliconTracking` will here be used to address `SiliconTracking_MarlinTrk`.

As `ForwardTracking` was written specifically to take over the forward tracking, it is expected to show equal or better results than `SiliconTracking`, in addition to being easier to maintain and to read.

The following plots come from a 10000 event sample without background. Figure 5.11 shows the dependence of the efficiency on the transversal momentum of the tracks. Plotting the efficiency (or ghost rate) against the transversal momentum is quite common, as p_T is one of the most important factors determining the path of the track: high transversal momentum tracks also have a high overall momentum[7] and are therefore less easily affected by multiple scattering or energy loss [41]. Also their helices have larger radii, therefore the path inside the detector approaches a straight line for higher p_T. For event reconstruction in the ILD detector the most important tracks are the high energetic ones, therefore they are the most important to be reconstructed.

Figure 5.11 shows that `ForwardTracking` has indeed accomplished a decent reconstruction efficiency, being better than its competitor, but still with room for improvement[8].

Another important parameter that has an effect on the efficiency is the distance of the vertex of a track to the IP[9]. In Figure 5.12 it can be seen that both algorithms lose efficiency with the distance from the IP. As most tracks come from an area close to the IP this drop is no catastrophe, but it should nonetheless be considered and improved in future versions.

Figure 5.13 shows that with increased number of hits in the true track the reconstruction efficiency rises. The drop of `SiliconTracking` for 7-hit-tracks should be further investigated.

Plotting efficiency vs. θ shows that the forward region is covered well, and that for higher θ the efficiency gets worse. This comes from the reduced number of layers of the FTD that are hit for higher θ as well as the θ-dependence of some criteria. The region above $\theta = 20°$ is already well covered by the TPC[10], so forward tracking efficiency is not very important there.

In Figure 5.15 the ghost rate is plotted. One can see that without background a ghost rate of around 10% can be achieved for the most values of p_T.

[7] In the forward region p is roughly ten times as high as p_T.

[8] The efficiency could be even higher, but that would lead to a longer computing time. This trade-off always needs to be carefully considered.

[9] E.g. the cellular automaton uses the IP as an additional hit. While this helps to reduce combinatorics it has a bad effect on reconstruction of tracks not coming from the area of the IP

[10] Already around 70 pad rows of the TPC are hit by tracks with $\theta = 20°$

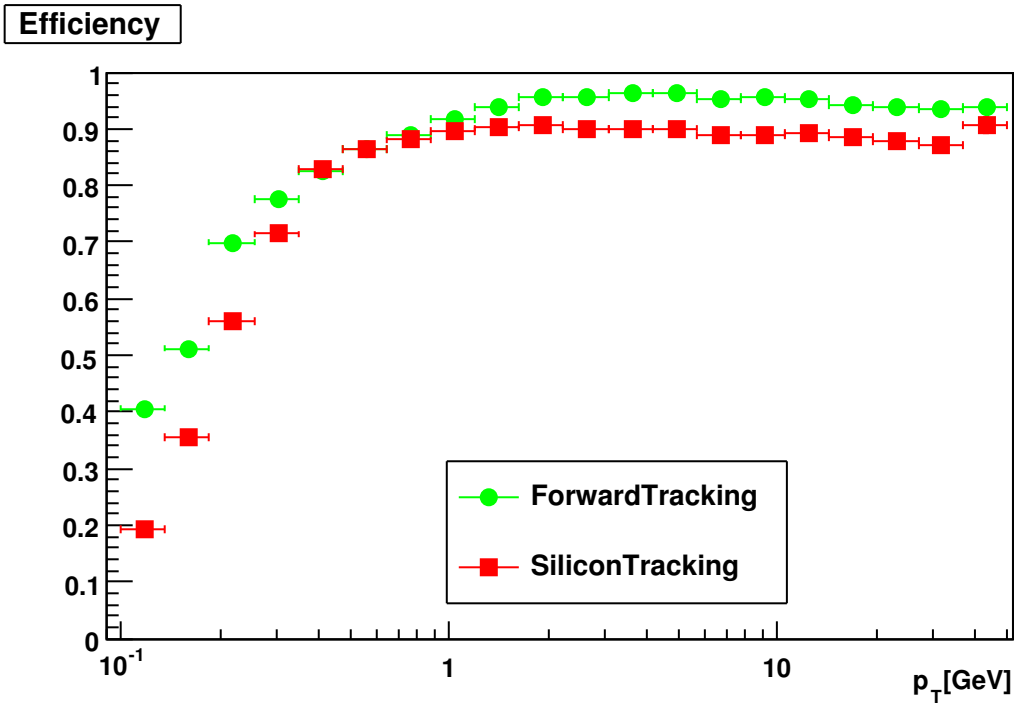


Figure 5.11.: Efficiency vs. p_T

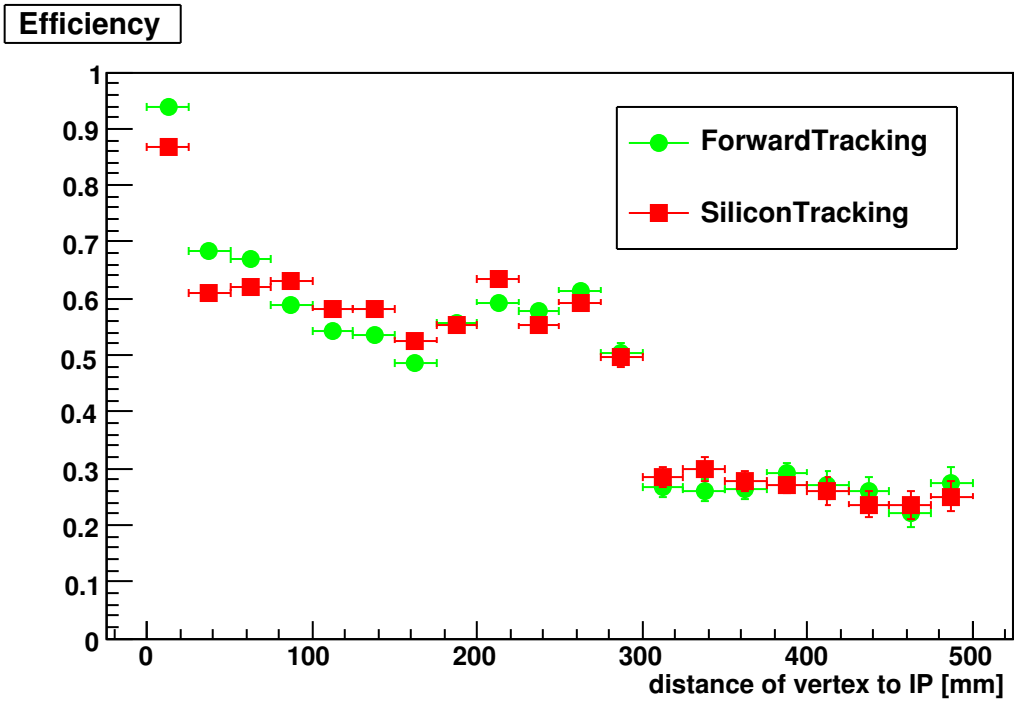


Figure 5.12.: Efficiency vs. distance to vertex

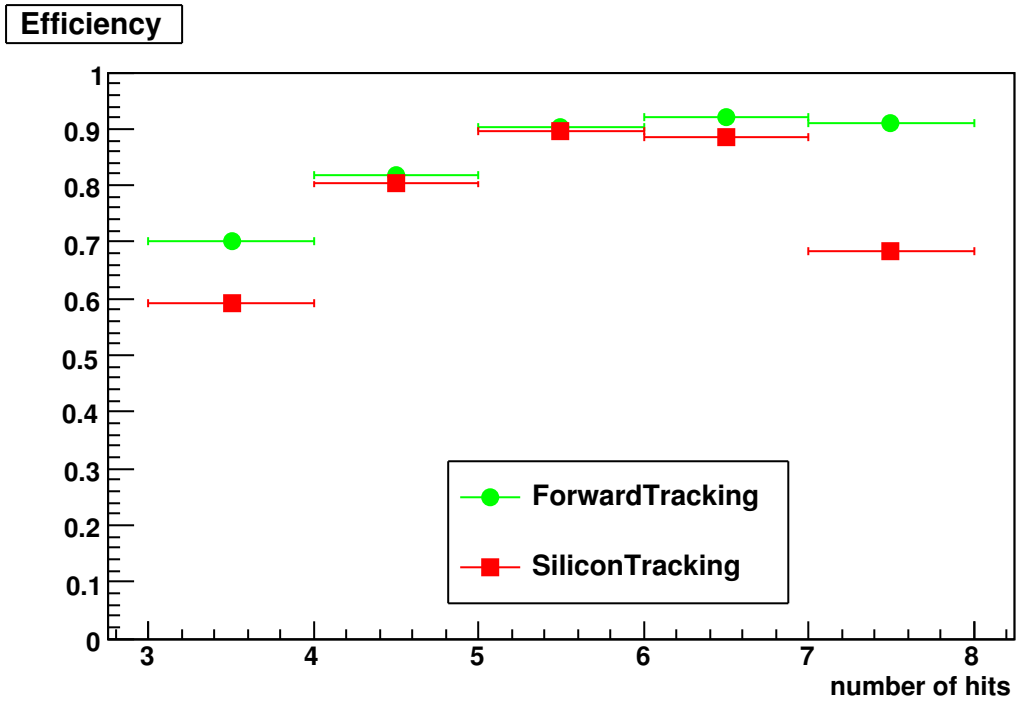


Figure 5.13.: Efficiency vs. number of hits in true track

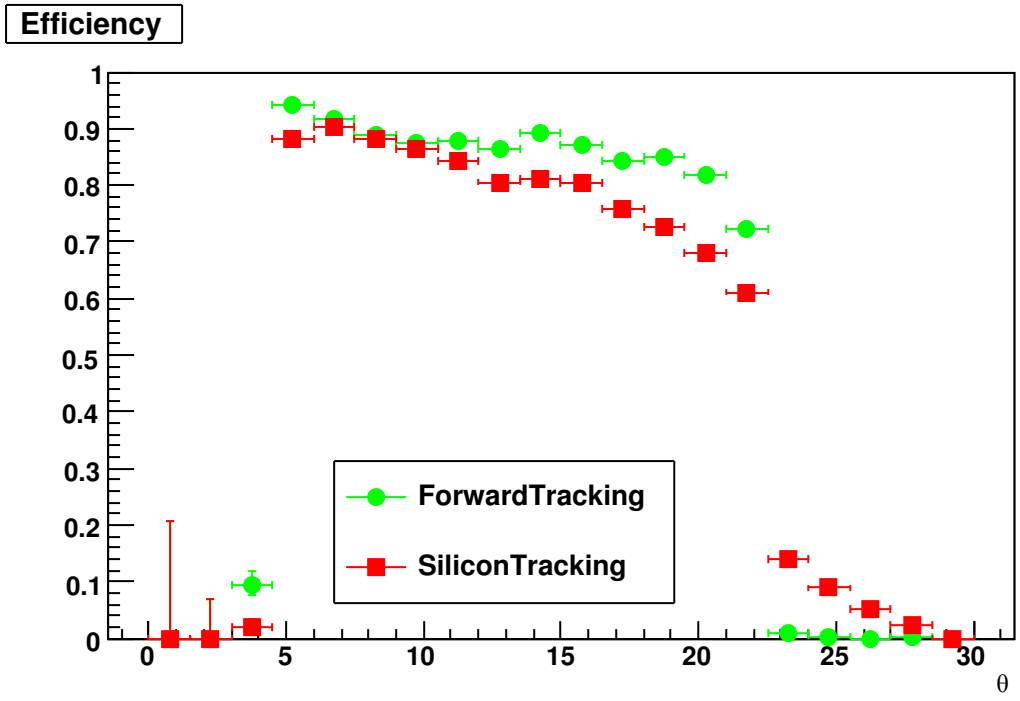


Figure 5.14.: Efficiency vs. θ

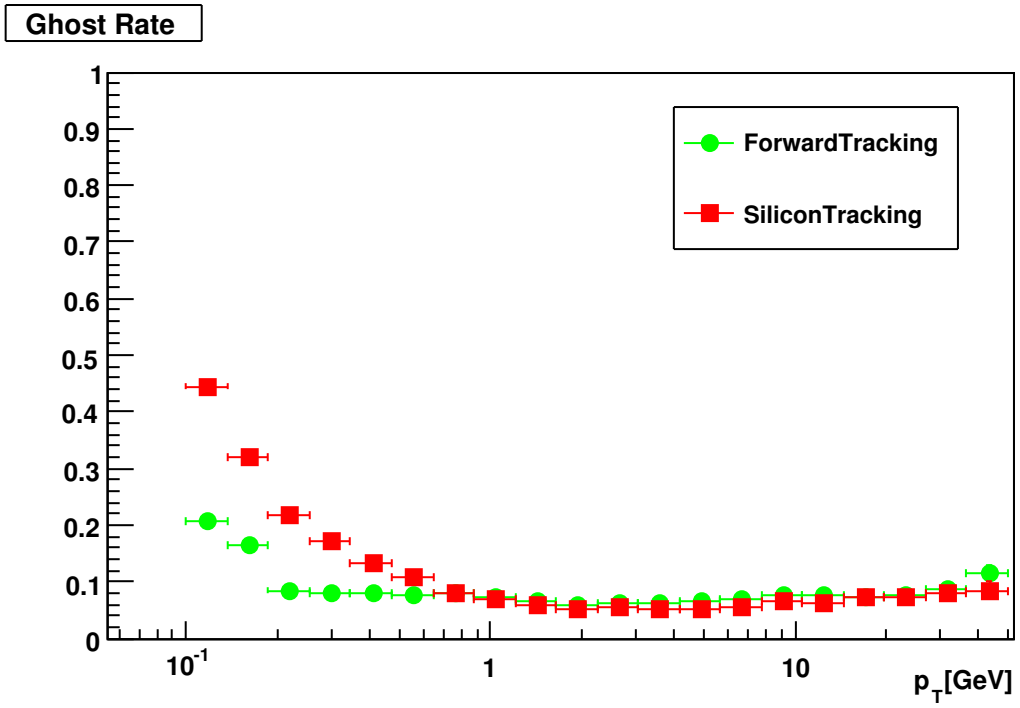


Figure 5.15.: Ghost Rate vs. p_T

5.3.0.1. Behavior under the addition of background

The behavior of the reconstruction algorithms under background addition is of special interest, as in the real application background will definitely be present. The expected levels of background vary with the used technology. For this study background levels as specified in the Letter of Intent of the ILD (LoI) [28] for the FTD at 1 TeV were used. The number of integrated bunch crossings of the pixel detectors were estimated to be 100.

In Figure 5.16–5.18 efficiency, ghost rate and time as a function of the background are plotted. Each measurement was based on 100 simulated events. The background parameter describes a multiplication factor on the LoI background, i.e. a background of 1 means 100% of the LoI background.

The results show that `ForwardTracking` copes with background very well: it loses almost no efficiency, and the ghost rate increases more slowly with background. Also the time consumed is less for high background. Note that the ghost rate is very high for large background – 70% of the reconstructed tracks are then ghost tracks. This could be reduced by applying cuts such the χ^2-probability cut, but it was decided to keep as many tracks as possible until the last stage of tracking `FullLDCTracking`, where the tracking results from the TPC and the silicon detectors are combined.

In Figure 5.19–5.22 the efficiency and ghost rate are plotted against p_T for different values of background in order to see the influence of background on the reconstruction of tracks with different transversal momentum. Efficiency in `ForwardTracking` drops equally for different p_T, while `SiliconTracking` shows especially losses in the high-p_T region under background. The strongest ghost rate contribution for both processors comes from the tracks with low transversal momentum.

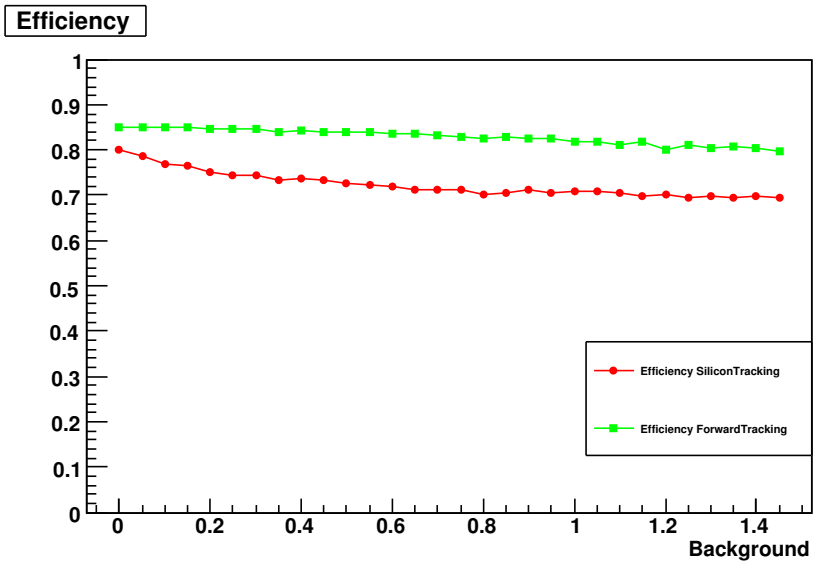


Figure 5.16.: Efficiency vs. background

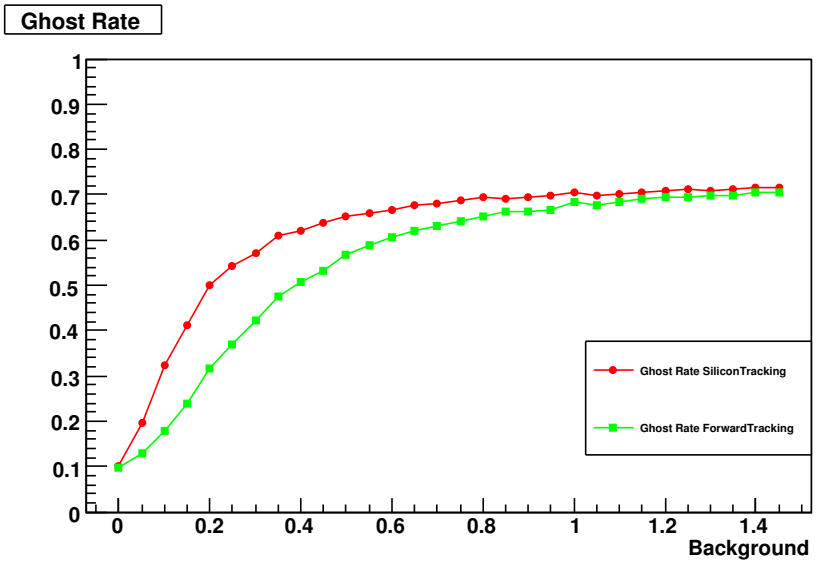


Figure 5.17.: Ghost rate vs. background

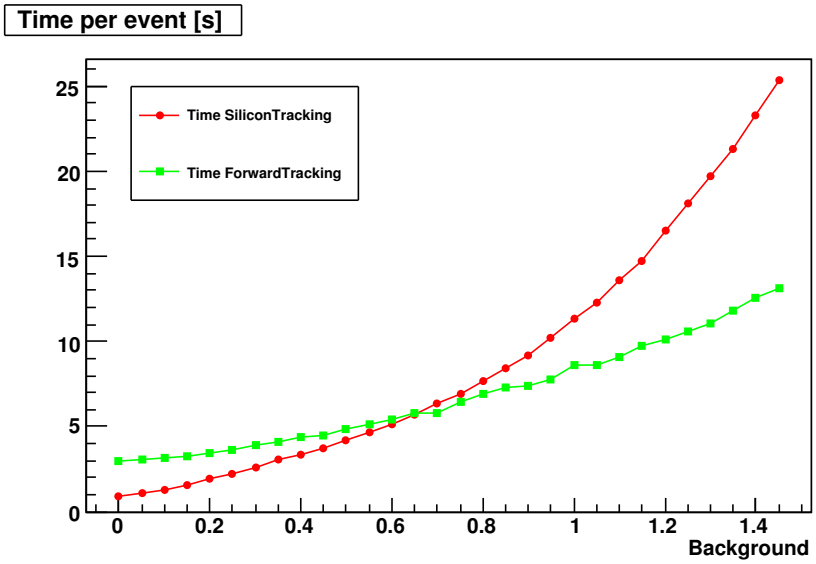


Figure 5.18.: Time vs. background

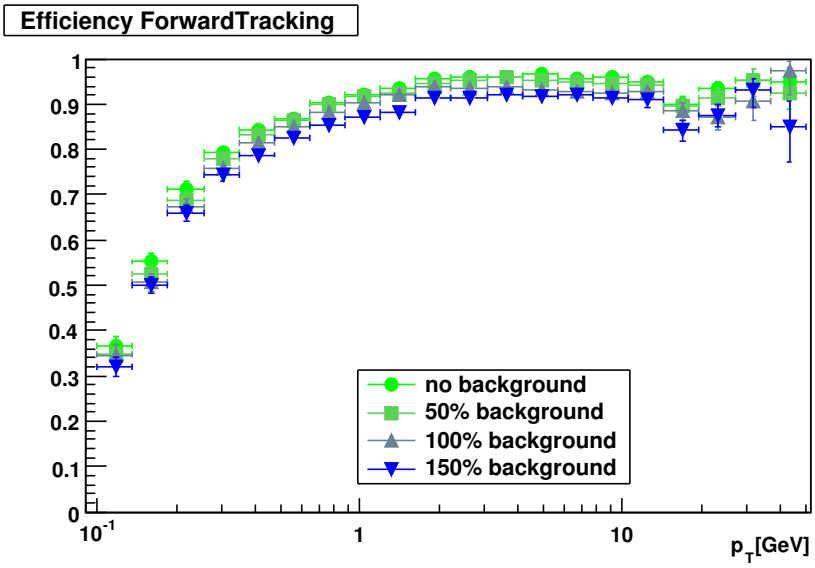


Figure 5.19.: Efficiency of `ForwardTracking` vs. p_T for different backgrounds

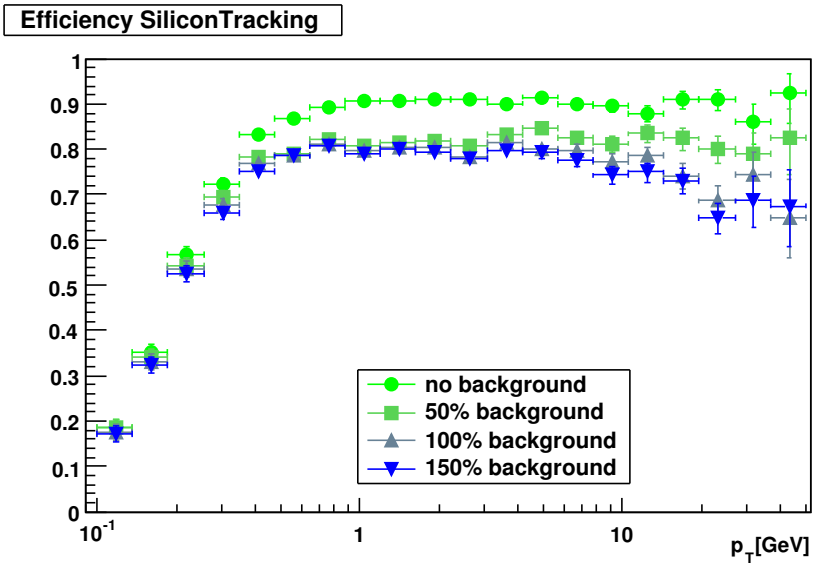


Figure 5.20.: Efficiency of `SiliconTracking` vs. p_T for different backgrounds

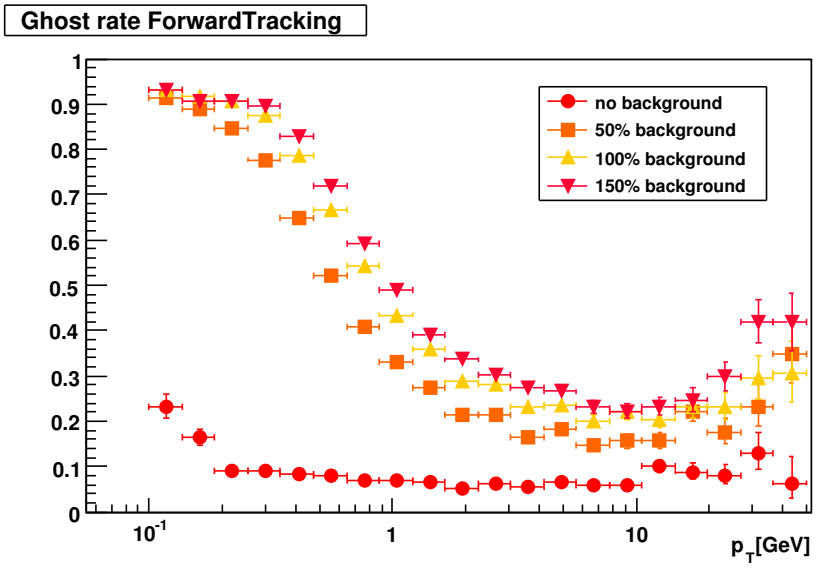


Figure 5.21.: Ghost rate of `ForwardTracking` vs. p_T for different backgrounds

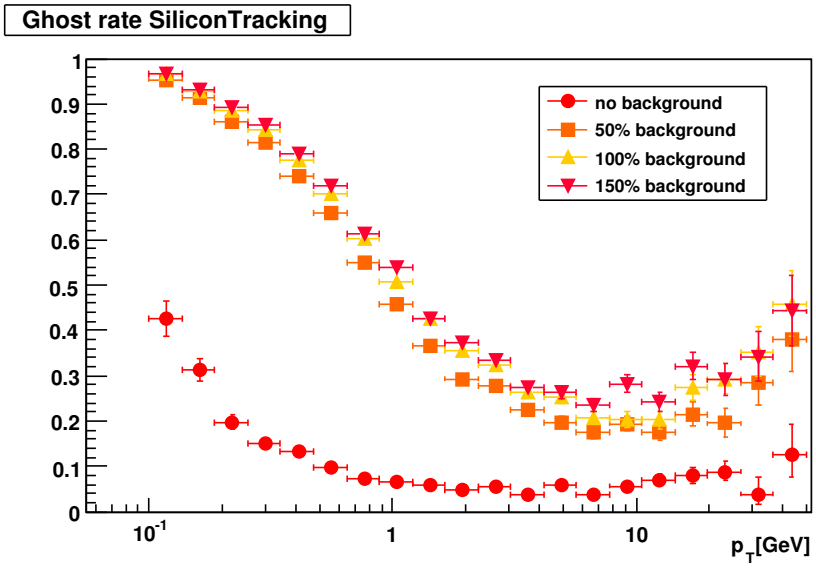


Figure 5.22.: Ghost rate of `SiliconTracking` vs. p_T for different backgrounds

6. Conclusion

For track reconstruction in the forward region of the ILD detector a new approach, differing from the previous track reconstruction has been chosen. It is based on a cellular automaton for track finding, a Kalman filter for track fitting and a Hopfield neural network (or a simpler algorithm) for ambiguity resolving.

The resulting software was split into the 3 distinct packages, named `KiTrack`, `KiTrackMarlin` and `ForwardTracking`, providing a separation of algorithms and abstract base classes, the framework dependent implementation and the actual module for track reconstruction.

The software was written in an object-oriented fashion and allows a high level of flexibility in the execution of the algorithms. The packages have become part of the standard reconstruction chain in the ILD detector reconstruction framework (`Marlin`) which is used for the upcoming Detailed Baseline Design report.

`ForwardTracking` allows for tuning of multiple parameters, in order to optimize especially for efficiency. The tuning shows that a χ^2-probability cut can reduce the ghost rate by a factor of 2 and that the Hopfield neural network was not able to surpass the efficiency of a simpler algorithm for the specific tracking problem at hand, but it still might prove useful for different background and reconstruction situations.

The direct comparison of the new track reconstruction `ForwardTracking` and the previous `SiliconTracking` has shown improved results for the new software. An increase in efficiency was achieved while keeping the ghost rate at the same level. With additional simulation of background on the Forward Tracking Detector, `ForwardTracking` has shown to need less reconstruction time and to achieve lower ghost rates and higher efficiencies.

Both old and new software are probably still operating below their full potential and could be further enhanced with additional, even more sophisticated methods and a thorough analysis of remaining weaknesses. They are both currently run in parallel in order to ensure a maximum of redundancy and thus safety.

7. Outlook

In the next years decisions concerning a future linear collider will be made, giving a clear path on the time line also for the reconstruction software. Currently the Detailed Baseline Design report DBD is prepared by the ILD collaboration and will document the technical features and possibilities of the ILD detector as well as giving a basis for making decisions about the future of the high energy landscape. In this context the performance of the tracking software in its latest version is also included. The DBD is to be finished by the end of 2012.

After the DBD decisions on the precise contents of the AIDA tracking package will be made, and it is possible that the packages `KiTrack` and `ForwardTracking` undergo further changes. Especially `KiTrack` could be unified with tracking software from other sources, for example the one in `Clupatra`.

Until a future linear collider is operational, the reconstruction software can be further enhanced. The current design allows to easily add additional algorithms at each stage (track search, track fit and ambiguity resolving). By adding new methods of track reconstruction and comparing the benefits in different scenarios the capabilities of the software could be further improved.

For the first stage, track finding tools such as the multilayer perceptron, easy back up methods such as Hough transformation or conformal mapping can be added. A combinatorial Kalman filter could be implemented as well, following the successful application at CMS [8]. Algorithms to remove outliers with robust methods, for instance the deterministic annealing filter, could be added to the fitting stage. Regarding fitting itself, different fitting-packages other than `KalTest` (for example `GenFit`) could be used in order to gain a comparison of speed and handling of material effects. A Gaussian-sum filter [4] could improve the fitting of electron tracks with their non-Gaussian errors.

For ambiguity resolving further algorithms could prove useful; however, alternative criteria for track quality determination may be even more important.

A more detailed steering and sector specific cut-offs, could improve the power to reduce combinatorics. Modules to automatically deduce steering values for the different algorithms based on the Monte Carlo truth could reduce ghost rates while keeping efficiencies stable.

The future of the software packages is now dependent on the overall situation of the ILD detector and the decisions of the high energy physics community and politics whether ILC or CLIC will be built and when. Whatever the future course will be, the tracking approach chosen here has shown to be a viable path in track reconstruction for a future linear collider.

A. The cellular automaton for track finding

The basic obstacles in track reconstruction are time and combinatorics and especially their mutual relationship. The task to achieve is easily formulated: to find the true tracks from a few dozen to a few thousand hits. After all, a very small algorithm could solve this problem: take every possible hit combination and use a quality indicator, such as the goodness of a fit, to decide whether it is a possible track. Throwing away the bad results leaves one with track candidates for further use.

One could use methods like the Kalman filter to determine the quality of hit-combinations as possible tracks, as it is one of the best tools to distinguish a real track from combinatorial background. There is however a downside: it takes quite some time.

In offline track reconstruction the time needed by the Kalman filter makes it suited to maybe fit thousands of tracks, but if one were to create track candidates from every possible combination of hits, this would give numbers far beyond that, even when there are only a few dozen hits. If one had for example 7 layers and on each of them were 10 hits, the number of possible track candidates that could be built from these hits is 10^7 if the tracks are restricted to each contain 7 hits.

The key is to greatly reduce the number of track candidates by finding a clever solution that sorts out as much combinatorial background as early as possible. A useful tool to achieve this is the cellular automaton. In this chapter is summarized, how the cellular automaton can be used for track finding in high energy physics. First, Section A.1 gives a short introduction on the task and the environment the author is using the cellular automaton in. In Sections A.2 and A.3 the definition of cellular automata and their application in tracking are presented and illustrated with a simplified toy detector. Finally Section A.4 gives an impression on how the procedure looks like for the Forward Tracking Detector at the ILD detector.

A.1. The environment: Track reconstruction in the Forward Tracking Detector of the ILD

Here some context on the experiment, the author is working for is given. Partly, because later in A.4 an example of how the cellular automaton can work within that environment is shown.

The **I**nternational **L**arge **D**etector ILD[28] is a detector for a future linear collider

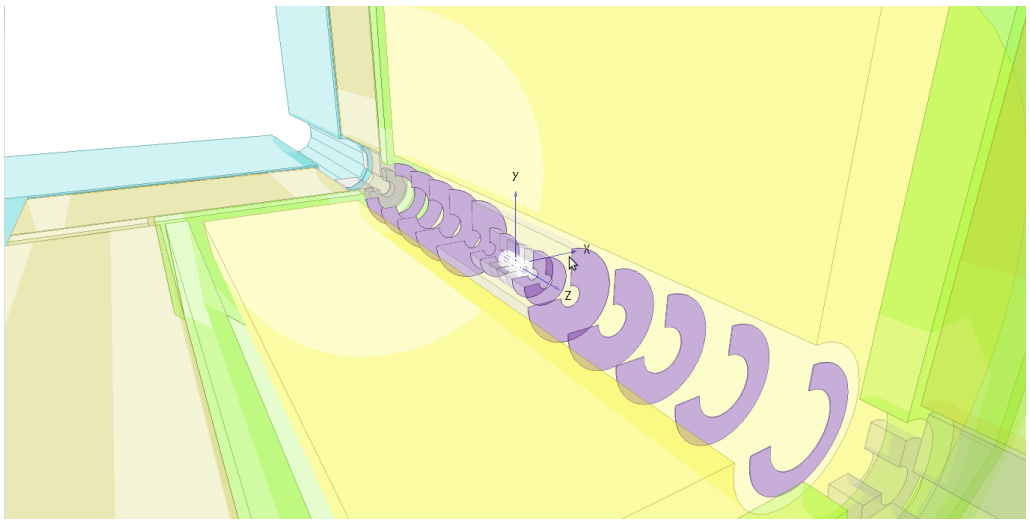


Figure A.1.: The **F**orward **T**racking **D**etector in the ILD (a schematic 270° cut)

(e.g. ILC or CLIC). Most tracking is done in the time projection chamber TPC (yellow in Figure A.1), but for the forward region, the area close around the beam pipe, an additional detector is needed to make the ILD detector as hermetically sealed as possible. This is the task of the Forward Tracking Detector FTD (violet). It consists of each two pixel and 5 back-to-back strip disks in the forward and the backward region. Only the forward region will be referred to, including the backward region as well; for symmetry reasons they can be treated the same way and it is common to address them both with the term "forward region".

The software packages written for track reconstruction in the Forward Tracking Detector are called `KiTrack`, `KiTrackMarlin` and `ForwardTracking` and are part of the ILD event reconstruction software at the moment. They can be accessed via the SVN repository at `https://svnsrv.desy.de/public/marlinreco/`. Also the repository can be explored with a web browser under `https://svnsrv.desy.de/viewvc/marlinreco/`.

A homogeneous magnetic field of 3.5 T pointing into z-direction (the direction of the beam pipe) is applied in the tracking parts of the detector. The charged particles therefore move on helical trajectories. Material effects such as multiple scattering and energy loss can however cause the trajectories to deviate from perfect helices. This is especially true for low momentum particles.

The goal is to use the hits caused by the particles on the silicon sensors to reconstruct their trajectories. The first step to achieve this, is to find which particles belong to the same track. This is called "track finding" or more general "pattern recognition" and is done with a cellular automaton, which will be presented in the next sections.

A.2. What is a cellular automaton

A cellular automaton (CA) essentially is a set of discrete entities with discrete states that can change depending on the local environment. The keywords are "discrete" and "local": the behavior of the entities, the cells, is completely determined by the other cells in their neighborhood. The maybe best example are real cells themselves, which is the original idea behind it. Cells are discrete, because there is always one cell or two or seven cells, but never 2.3 cells. And one can assign them discrete states like: "living" or "dead".

What distinguishes the CA from biological cells is that it also uses discrete time steps. The situation does not change continually, but stepwise. With every iteration the situation evolves following rules that specify the dependence of a cell on its environment.

A.3. The cellular automaton for pattern recognition

So far it sounds as if the cellular automaton would be an interesting tool to simulate cells or population growth, but that is not its limit. For example, it is useful for pattern recognition.

What is a pattern? As hard as it is to define "pattern", one might say that a pattern follows some rules: like the pattern of tiles on a chess board will always follow the rule that there is a black tile next to a white tile.

Cellular automata are based on rules as well: the rules on how the states of the cells change for a given environment. Different rules give different results. If one implements rules in a cellular automaton that resemble the rules of a pattern, one wants to find, and then lets the cells following this pattern survive and others die, at the end the living cells will form the pattern.

A.3.1. Demonstration with a toy detector

The basic principles of the cellular automaton for track reconstruction can best be understood by studying it step by step with a graphic aid. Here the steps of the CA for track finding are explained with the help of the toy detector in Figure A.2.

This is a two dimensional detector, for the sake of simplicity the third dimension is ignored. Still the magnetic field lines point into z-direction, so the paths one expects of charged particles are circles. The blue dot in the middle of the detector is the nominal IP (interaction point).

The event that is to be reconstructed is shown in Figure A.3. These tracks are the patterns one wants to find. The event is unrealistic in the sense that all the tracks are in the upper half, but for didactic purposes it is more comfortable to only show the upper half of the detector, simply because it saves space and there will be enough to look at in the upper half anyway.

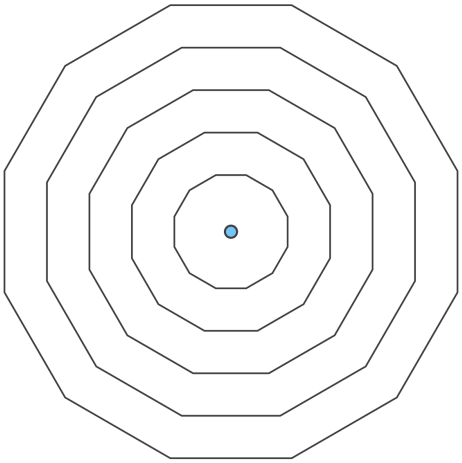


Figure A.2.: 2-dimensional toy detector

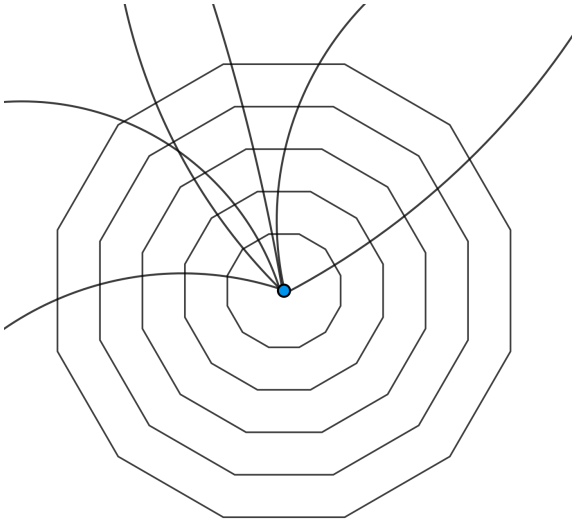


Figure A.3.: True tracks from an event

The particles ionize the material of the detector as they pass through and this results in the measurement of hits: Figure A.4. Of course one does not know the tracks, when reconstructing the event, only the hits are known as in Figure A.5.

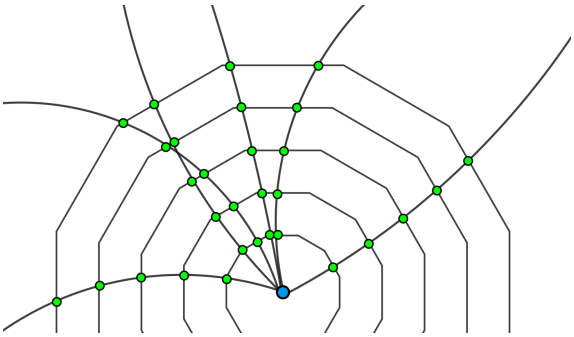


Figure A.4.: The true tracks + hits

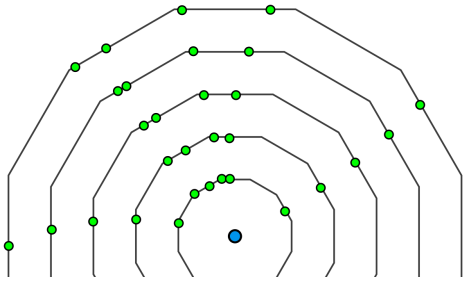


Figure A.5.: The true hits

Also in real applications there is always background[1]. Figure A.6 depicts a more realistic picture of the situation by adding background (red hits).

As one usually does not have the possibility to know what hits come from background, the realistic starting situation for the reconstruction of tracks in the toy detector is Figure A.7, where hits cannot be distinguished anymore.

This already makes it harder to find the tracks simply by looking at the hits. Still it is possible, as this is a simplified 2 dimensional environment and the human brain is very capable of pattern recognition. In a 3-dimensional situation with a large background, however, the task gets much more difficult.

[1] e.g. electron-positron pair production, the slow read out time of pixel sensors causing pile-up of events or the ghost hits on back-to-back strip detectors

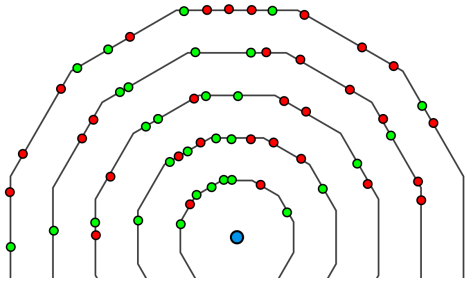


Figure A.6.: True and background hits

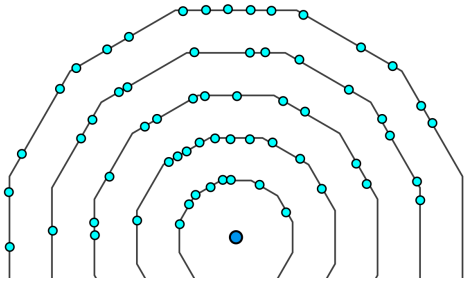
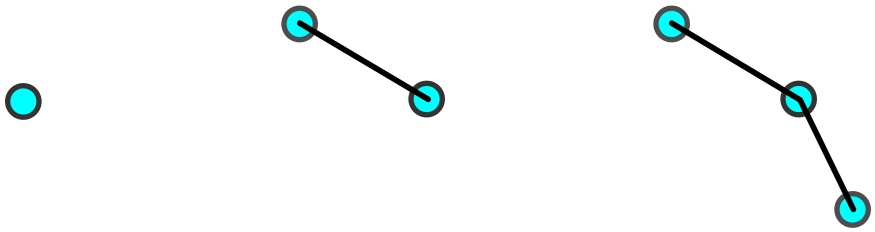


Figure A.7.: All hits, indistinguishable

Next the procedure how the cellular automaton is applied to reconstruct this event is shown. Before it was mentioned that a cellular automaton consists of discrete entities, so-called cells. In a biological simulation these are simply representatives of real cells, but what is a cell in track reconstruction? As one wants to find tracks, parts of them would be a good starting point. These are called segments. Their loosest definition would be: a collection of hits, which, by combining several of them, could form a track.

Therefore the most simple segment is a single hit. It is the smallest part of a track and if hits are combined, this results in a track. In other experiments using the cellular automaton often segments are defined as two-hit combinations, but here this more general concept was chosen.

Segment consisting of exactly 1 hit can be called 1-hit-segment. A segment consisting of two hits is then a 2-hit-segment. These can be illustrated by connecting two hits, for example by a straight line from one hit to the other, which of course does not represent the actual path, which is usually curved. Longer segments can be formulated as well, giving 3-hit-segments, 4-hit-segments and so on. An illustration of segments used in this demonstration is shown in Figure A.8.



(a) 1-hit-segment (b) 2-hit-segment (c) 3-hit-segment

Figure A.8.: Segments of different lengths

Looking back at Figure A.7, one can see that it already contains segments in the form of the single hits, i.e. 1-hit-segments. One could start the cellular automaton algorithm right there, but waiting until the next iteration makes more sense, because for 1-hit-segments there simply is not much gain in doing it. In this example event,

as in many real applications, there would not be much to see. Coming back here once the CA is fully explained and checking out the benefits of the CA for the 1-hit-segments in this example is probably a good idea.

Instead, everything that the automaton does for 1-hit-segments will be shown, while skipping the single characteristic step of the CA there and explaining it afterwards for the 2-hit-segments, as this gives a better example. If one wants to get to 2-hit-segments, one needs to connect the 1-hit-segments first. There are an awful lot of possible connections that can be made and in real events there are even drastically more.

Therefore one needs a criterion that determines, whether it makes sense to connect two hits. If one looks at the true tracks in Figure A.4, one sees that all the hits coming from one true track are not too far apart from one layer to the next. If the layers have a distance of 15 cm, then one could try to connect only hits that have a distance of less than 20 cm. This 20 cm cut-off seems completely arbitrary, but for real applications this value is found by analyzing lots of true tracks and checking in what range these values lie. In order not to lose (too many) genuine tracks, but to dismiss as many wrong ones as possible, careful analysis is crucial. Connecting hits close enough results in Figure A.9. The IP was used as an additional hit, adding information to the cellular automaton. Including it is valid insofar as most tracks come from an area close to the IP[2]

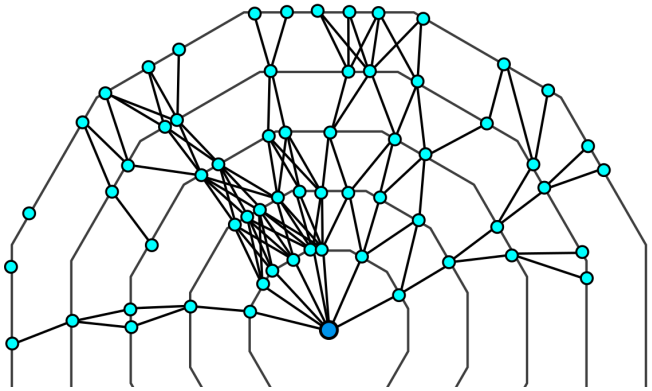


Figure A.9.: Hits are connected if close enough

The straight lines in Figure A.9 are the 2-hit-segments, the cells for the cellular automaton. One can now start the algorithm. One of the key features of the segments (cells) is that they have a state. While a biologist might find states like "living", "dead" or "reproducing" useful, for track reconstruction unsigned integer

[2]To handle secondary vertices farther away there are different possibilities. If for example their rough location is known from other sources, they can be added as additional hits, although this needs to be carefully considered on a technical level (especially concerning the layers of the vertices and the hits).

values can be used. In Figure A.9 all the segments have state 0, which is depicted by their black color. At the beginning the CA always starts with all states being 0[3].

The algorithm is as follows:

1. Every segment is checked if it has a segment connected on the inside with the same state fulfilling some criteria that separate combinatorial background from valid signals as much as possible. These inner segments are called "neighbors".

2. If a segment has at least one neighbor, the state of the segment is raised by 1 at the end of the iteration.

3. This is repeated until the states of all segments are stationary.

The criteria are very important. They allow to sort out as many false connections as possible. A criterion has to be something that can distinguish between real tracks and wrong ones. For this example the following simple criterion will be used: If two segments have an angle smaller than 10° they are compatible, if it is bigger they are incompatible, see Figure A.10.

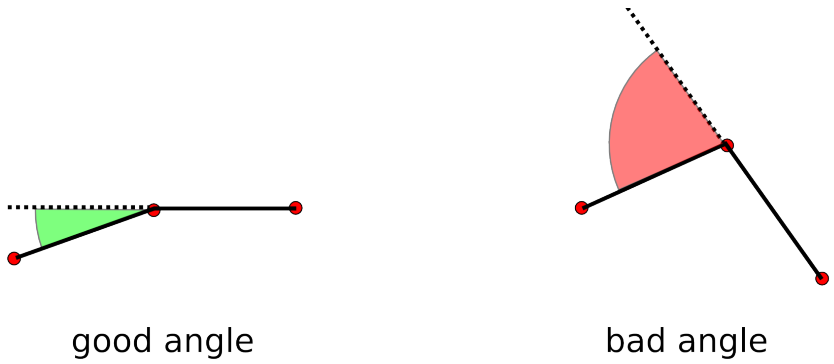


Figure A.10.: The criterion: angle between two 2-hit-segments

This is only one of many possible criteria. It is simple, fast and it works for the true tracks that are searched for[4].

So the first iteration of the automaton reads as follows: Every segment gets checked for neighbors: connected segments on the inside that have an angle below 10° to the segment, and that have the same state. If a segments has at least one neighbor the state of the segment is raised by 1. This results in Figure A.11.

[3] In other experiments a starting state of 1 has been chosen, but it essentially makes no difference as long as one uses it in a consistent fashion throughout the algorithms

[4] This criterion is especially useful for high transversal momentum tracks as they result in rather straight trajectories. For low momentum tracks, the angles between the segments grow bigger. So if one wants to look for low transversal momentum tracks, other criteria are better suited. There is however still a use for this criterion when it comes to low-p_T-tracks if one applies a minimum cut-off. Actually most cut-offs can be formulated with a minimum and maximum cut-off.

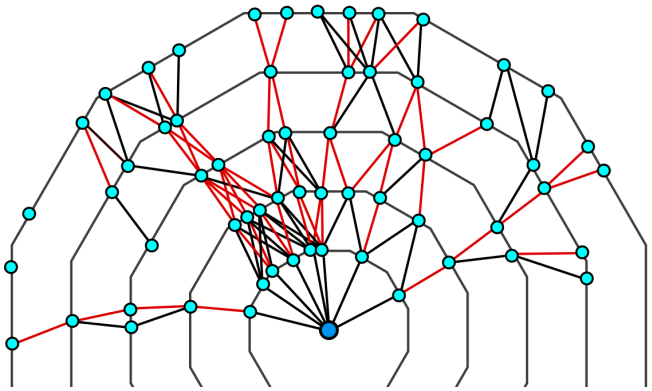


Figure A.11.: First iteration

The red segments have state 1, the black ones are still on state 0. Before further investigating the result, "layers" are introduced for easier addressing the segments. For single hits, layers can easily be assigned. One can say that the IP has layer 0 and the first detector on the inside is layer 1 and so on. One can continue this method for longer segments as well if one assigns every segment the layer of its innermost hit. This results in the 2-hit-segments lying on layers 0 to 4 as shown in Figure A.12.

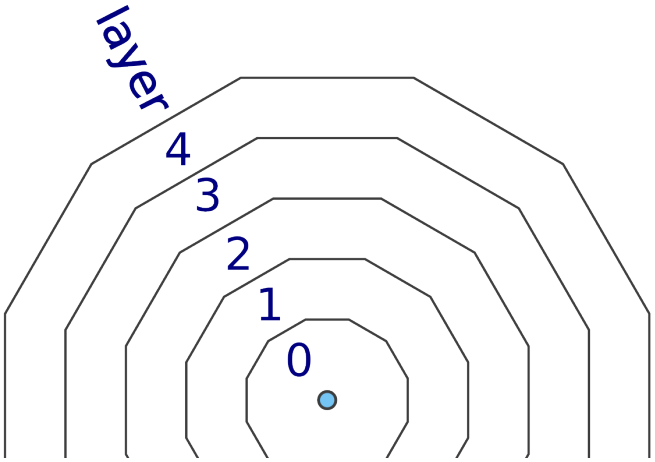


Figure A.12.: Layers of the 2-hit-segments

If one takes a look at the result of the first iteration in Figure A.11, one notices that all the segments on layer 0 (i.e. the innermost segments) are all still in state 0. This is due to the fact that they don't have any segments on the inside, i.e. have no neighbors, thus can't raise their states. Segments on higher layers do have segments on the inside, so in principle it is possible for them to raise their states.

A second iteration leads to Figure A.13. Now there are also segments with state 2 depicted in orange. Of course the segments in layer 0 still stay at state 0. But

also the segments in layer 1 could not reach a state higher than 1. This is for the simple reason that a segment with state 1 needs a segment on the inside with state 1 in order to reach state 2. But as the potential neighbors of the segments on layer 1 are on layer 0 and thus stay at state 0, this never happens. This is a general rule: no segment can have a state higher than its layer and this derives from the segments at layer 0 and the "neighbors must have the same state"-rule[5].

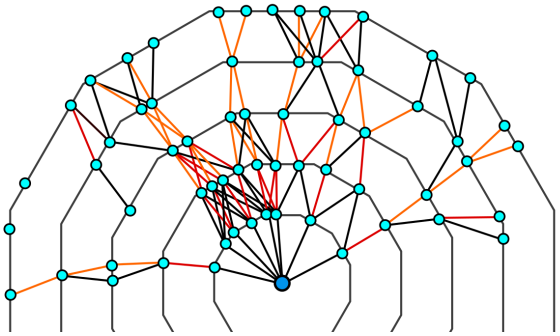


Figure A.13.: Second iteration

In this second iteration the question if potential neighbors have the same state or not became important. It was not important in the first iteration, as there all the segments had the same state: 0. This time many segments could not raise their state due to this rule. This will be important in the final result later on.

Repeating the procedure again two times until there are no changes of states anymore gives Figure A.14 and Figure A.15.

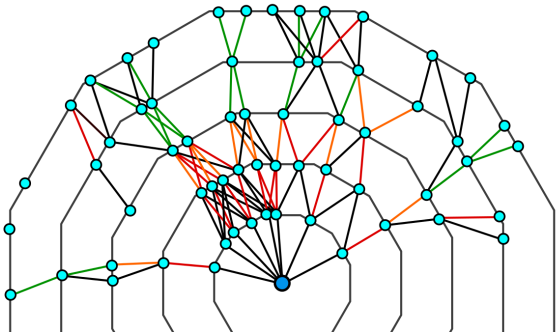


Figure A.14.: Third iteration

[5]This also limits the maximum number of iterations: after n iterations the highest possible state is n, therefore one can stop iterating, if $n = layer_{max}$, because no changes can happen anymore.

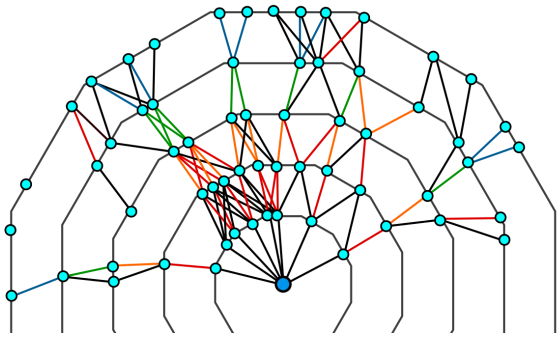


Figure A.15.: Fourth iteration

The possibility to change states in further iterations has now stopped. This means one can now collect the results of the first round. This is where the question arises, what one has gained besides nicely colored lines. One can see the answer in an example: A blue segment in layer 4 has state 4. In order to arrive at state 4, it needed a neighbor with state 3 on layer 3. This one on the other hand needed a neighbor with state 2 on layer 2 and so on; until one arrives at the innermost layer, the IP. It all just comes from the rule, that a segment can only increase its state, when it has a segment on the inside with the same state. This gives a connection all the way from the outermost layer to the IP. This connection fulfills in every step (in every combination of two following segments) the criterion that was applied (the angle between the segments) and gives a possible track candidate.

What if a segment on layer 4 has not state 4? Then it is not connected all the way through to the IP! That is a very useful information, because that identifies these segments as potential combinatorial background[6]. The same applies for all segments on layer 3: if they didn't reach state 3, they are not connected to the IP. So every layer has an allowed state as depicted in Figure A.16. One can now erase all the segments which have a state below their layer-number and thus greatly reduce the combinatorics: Figure A.17.

The combinatorics can be still further improved by repeating the same procedure with longer 3-hit-segments and a new criterion suited for them. First one has to connect the remaining 2-hit-segments in Figure A.18.

[6]Of course not all tracks necessarily come from the IP. But nearly all of them come from the area roughly around it. If one had a situation with a high percentage of tracks coming from different areas, this could be taken into account by keeping all end-segments above a certain state, which guarantees sufficient length, but not a connection to the IP.

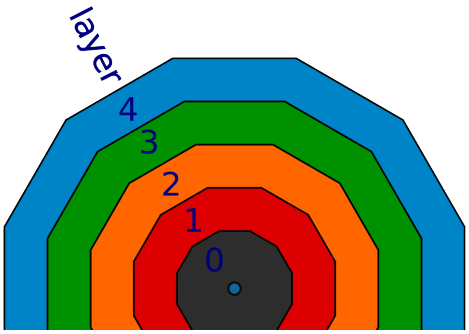


Figure A.16.: The allowed states

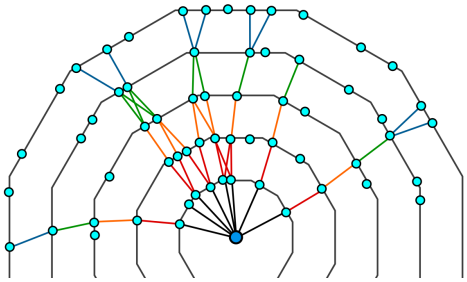


Figure A.17.: Erased bad segments

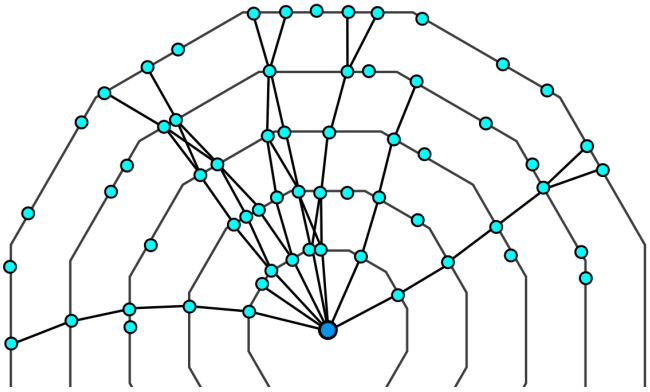


Figure A.18.: The remaining 2-hit-segments

The 2-hit-segments were connected over single hits, i.e. 1-hit-segments. 3-hit-segments are likewise connected by 2-hit-segments. To say it in a different way: two connected 2-hit-segments share exactly one hit: the one in the middle. Two connected 3-hit-segments share exactly 2 hits, the ones in the middle.

This means that there will be quite some overlapping in the next picture, so for reasons of visibility the depicted segments will mostly not be perfectly in place, but shifted a bit sidewards, so that the lines overlap as little as possible: Figure A.19.

Before one can start the cellular automaton again, one has to come up with a new criterion for these longer segments. A quite simple but effective one is to use the outer segment to extrapolate into the direction of the innermost hit of the inner segment and measure the distance from the hit to the extrapolation. Only distances below 1 cm are allowed. Based on this, the first two iterations of the cellular automaton are done like before in Figure A.20 and A.21.

Again every segment was checked, if it had a segment on the inside that fulfilled the criterion and had the same state. If a segment had at least one of those neighbors, it was able to raise its state after the iteration. Repeating this once more, the last

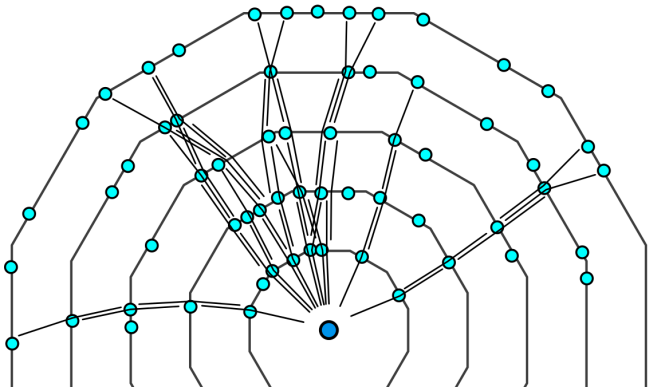


Figure A.19.: The 3-hit-segments

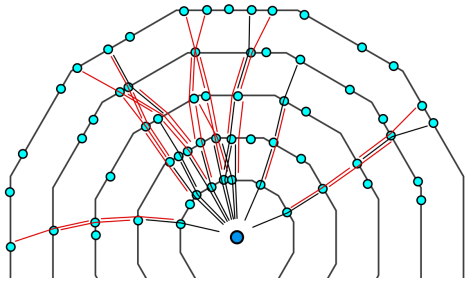


Figure A.20.: First iteration

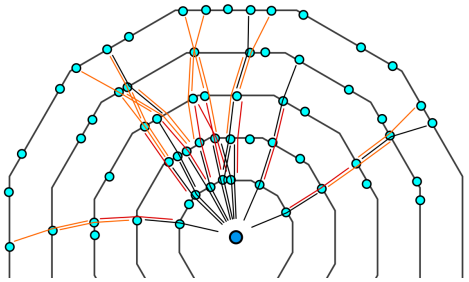


Figure A.21.: Second iteration

iteration[7] results in Figure A.22. Cleaning by erasing the segments with bad states results in Figure A.23.

The remaining segments are the final results of the cellular automaton, the track candidates in Figure A.24. Here the work of the cellular automaton is done. One could still make longer segments and use criteria suited to further decrease the amount of track candidates, but experience has shown that the most decrease of combinatorics happens with the 2-hit-segments, if the cuts for the criteria are set in a good way.

To come back to what was said at the beginning: this is also true for the 1-hit-segments. Almost all single hits are connected all the way through to the IP. Many of the possible connections got sorted out by the cut-offs applied at the beginning, but there are still enough to connect almost every single one all the way through. One could use the CA there, but the benefit is minimal and doesn't pay off.

The 3-hit-segments do of course use refined criteria and are able to distinguish

[7]The layers of the segments are still the layers of their innermost hit, so the highest layer here is layer 3, because a 3-hit-segment starting from layer 4 would reach layer 6 with the outermost hit and there is no layer 6 in the toy detector. This is why the third iteration is already the last one.

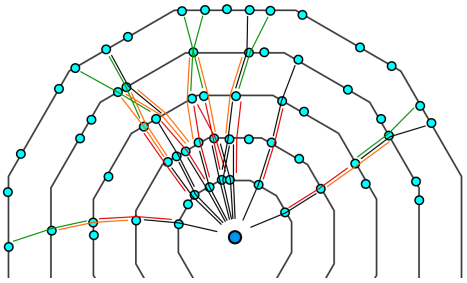


Figure A.22.: Third iteration

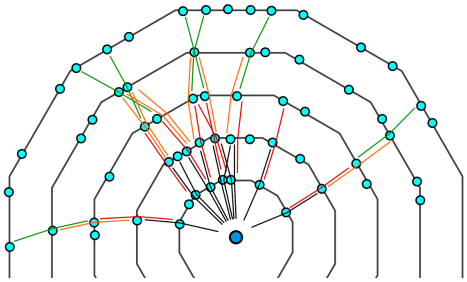


Figure A.23.: Erased segments with bad states

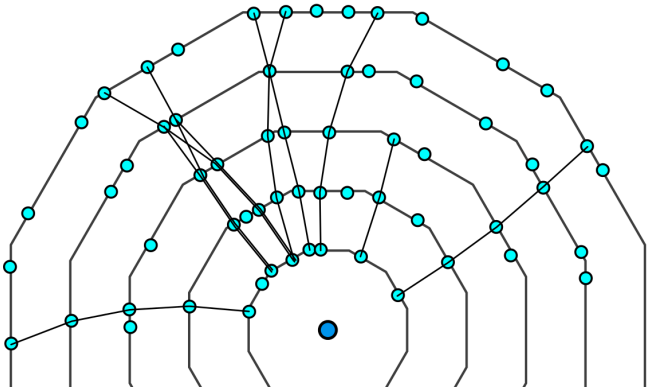


Figure A.24.: Track candidates resulting from the cellular automaton

even better, but their usefulness highly depends on the situation (e.g. are there many hits because of background or because of ghost hits from a jet?).

For all different lengths just one criterion was given each as an example, but usually multiple criteria are utilized for in step. There are many different possible criteria one can come up with, but their power depends strongly on a careful analysis and not all criteria are suited for every tracking situation.

To summarize the procedure in the toy detector: The patterns searched for were tracks. Due to analysis ones knows certain things about those tracks:

- They come from an area around the IP
- The hits caused by them are closer to the next hit in the track than 20 cm
- The 2-hit-segments of the tracks all have angles below 10 degrees
- The extrapolation of 3 hits in the tracks to the next hit is closer to the hit than 1 cm

These criteria define the patterns that can be found. If more criteria had been used, it would have been more effective[8]. The cellular automaton was used in such a way that only cells fulfilling these criteria reached a state (an integer number) that was equivalent to the layer they were on, giving the possibility to identify false segments and erasing them. This way, the combinatorics could be reduced in each step. Finally, when all the criteria have been used, the remaining cells reproduced the pattern that was searched for: tracks.

A.3.1.1. Remaining track reconstruction

The cellular automaton delivered track candidates, but one can see in Figure A.24, that the final tracking is not yet done. There are usually two more steps that are performed. First, the reconstructed track candidates are fitted and a quality cut is applied. This can be done with the before mentioned Kalman filter, which can give a feedback on the goodness of the fit (i.e. the χ^2-probability). The quality cut results in Figure A.25 and Figure A.26.

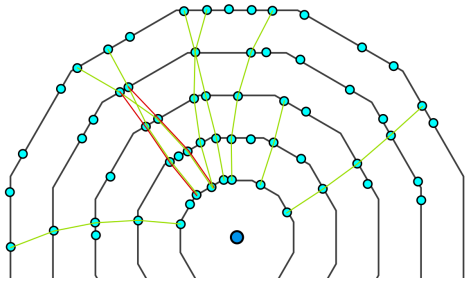


Figure A.25.: Results of a quality cut on the track candidates

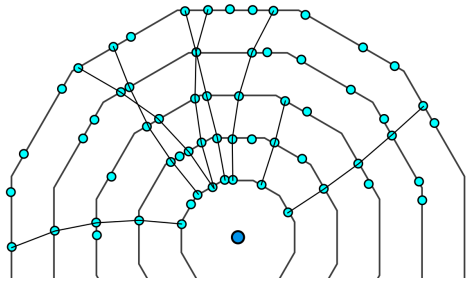


Figure A.26.: After the quality cut has been applied

A final step is the resolution of ambiguities: In Figure A.27 tracks sharing hits are marked in red. These tracks are incompatible and only a subset of those is usually saved, where no tracks share hits. There are various algorithms able to solve this problem, for example a Hopfield Neural Network. The solution could look like in Figure A.28.

With these last steps the tracking is finished, resulting in the final track collection shown in Figure A.29. In this toy detector example the original tracks were all found. Additionally there is a ghost track in the final collection, the short 3-hit track.

[8]On the other hand each criterion needs calculation time, so a few criteria in each step of the cellular automaton are usually sufficient. Sooner or later additional criteria don't give any more benefit.

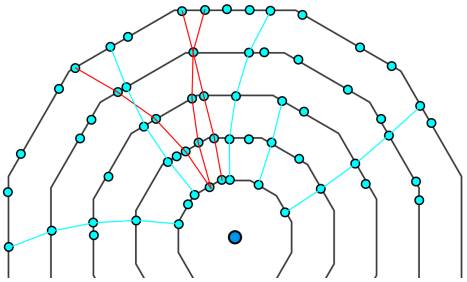


Figure A.27.: Incompatible tracks

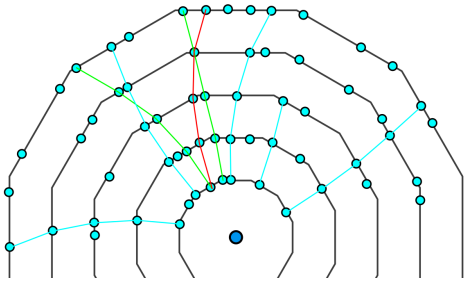


Figure A.28.: After ambiguity resolving

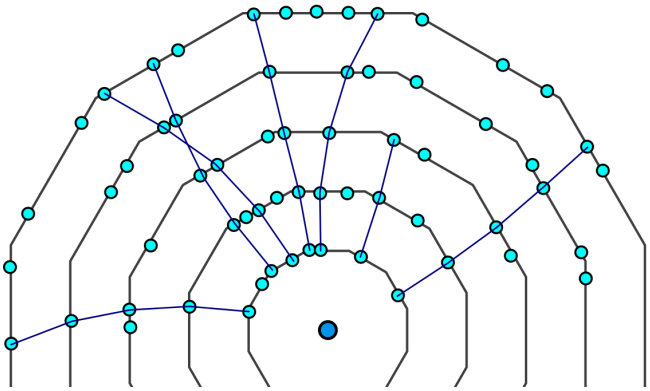


Figure A.29.: The final track collection

A.3.2. All the small things

There are quite a few mistakes that can be done when implementing a cellular automaton. Some of them are listed here, but there surely are far more.

A.3.2.1. Sectors

The use of sectors can save some time or even a lot, depending on how it is applied. At the moment in `ForwardTracking` the sectors are used in a very basic way. There is more power to them, as will be explained later.

Usually when one starts with all the hits, one has to save them in some sort of container anyway, so one might as well store them with different keys (e.g. in a map). For example the layers of the toy detector (Figure A.2) form a dodecagon. One could take every side of them as a single sector. In fact they would for construction reasons possibly even correspond to real single sensors. One then stores the hits in a map under some sector identification key. When afterwards the hits are connected to 2-hit-segments, one only connects hits from sectors that will possibly give meaningful results. So for example it would make sense to only connect two hits on the next

inner layer that are on the same sensor or the adjacent ones: Figure A.30.

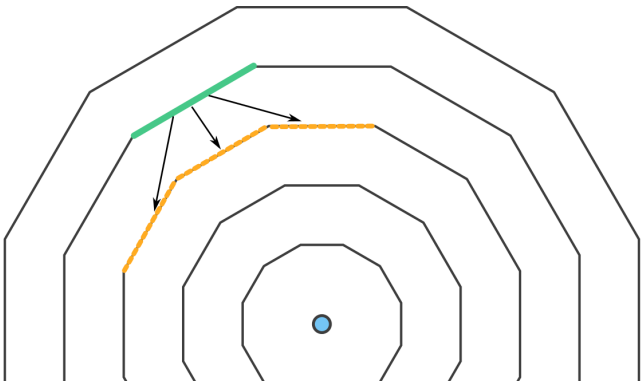


Figure A.30.: Connecting only hits from certain areas

This way a lot of computing steps are saved, because the computer does not need to calculate the criteria for every possible combination. If there is a hit on a sensor, it does not need to check 12 other sensors on the next inner layer, it is enough to check 3 sensors, saving 75% of calculation time.

But there is an even higher power to sectors: the cut-off values used for the criteria can be made sector specific. This might not seem meaningful in the toy-detector, but usual detectors don't have this perfect simplified geometry. There might be overlaps or layers which are not equidistant. In those cases it can be useful to apply rules like: "segments from inner layers might only have an angle of 2 degrees, while the ones further on the outside are allowed larger angles". This, in combination with a good analysis, can help to reduce wrong combinations and especially in asymmetric detectors this can be crucial. For example this has been done in [38].

This is not implemented in forward track reconstruction for the ILD so far for two reasons: First, the detector in the forward region (the FTD) is symmetric in many ways and sector specific cut offs don not seem too important at the moment. And second, because implementing it and doing the corresponding analysis takes quite some time. For example: the sectors of 1-hit-segments are simply the sectors of the hits, but for 2-hit-segments this gets harder. One could define a new "sector" for 2-hit-segments by somehow combining the sectors of the two single hits. This is not trivial as the resulting number of sectors can be huge, for 3-hit-segments even more so. Also for analysis one needs multiple tracks through all those sectors. Of course one could use every symmetry in the detector to combine single sectors into larger ones, but still it remains a non-trivial task.

A.3.2.2. Tracks not from the IP

Tracks coming from secondary vertices far from the IP can also be found. For this, at track-collection all segments with a state above a certain minimum are used as

start for tracks. This way one gets also tracks that are not connected to the IP, but have a minimum length. Of course this requires not to delete segments just because they have a state below their layer at the end of each CA step, thus it reduces the power of dismissing combinatorial background. Another idea would be to use pre-calculated vertices based on other reconstructed tracks if available. For the FTD these could come from the tracking in the TPC. Instead of only using the IP as additional hit for the cellular automaton, one could create an additional hit for every found vertex.

A.3.2.3. When to check the Criteria

Checking the criteria too often is a waste of computing time. When to actually calculate them ideally depends on the implementation of the cellular automaton. If one would use a CA similar to the one described here, the calculation best happens when the segments are combined to longer ones. In these steps the connections to the other now longer segments have to be stored somewhere. If these connections are only made, when the criteria are fulfilled, one can later iterate over every connection without recalculating the criteria.

A.4. Demonstration for the FTD

The goal of this work is track search in the forward direction of the ILD detector, where the Forward Tracking Detector FTD is placed. Here the just presented procedure of the cellular automaton is shown for track reconstruction for the FTD.

It starts with the building of the 2-hit-segments. As mentioned before, two hits only get combined into a 2-hit-segment, if they fulfill certain criteria. For example one criterion used for these pictures is the distance between two hits divided through their z-distance. This value must be below some reasonable cut-off such as 1.1:

$$\frac{R}{\Delta z} < 1.1 \tag{A.1}$$

In Figure A.31 the hits and the 2-hit-segments can be seen. In reality many more segments are present. The reason the fully realistic picture is not yet shown is that one could not see too much. Ten thousands of segments fill the space and are hard to distinguish. Still, this will be shown later on.

On the 2-hit-segments the cellular automaton can be used. In every iteration it checks every segment if it has a neighbor. Criteria of compatibility of two 2-hit-segments here are for example the angle between the segments in 2 and 3 dimensions. After the Automaton was processed one gets Figure A.32, where segments with a low state are bright blue and thin, while segments with higher states become redder and thicker.

In the next step the situation is cleaned up and all segments that don't have the highest possible state in their layer get erased: Figure A.33.

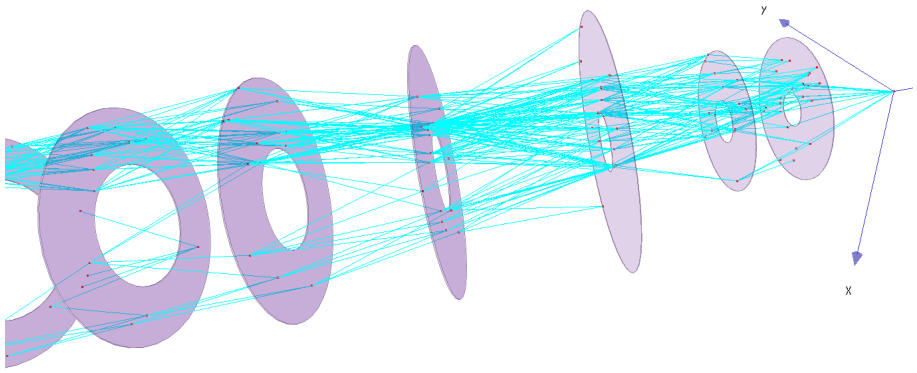


Figure A.31.: Build 2-hit-segments

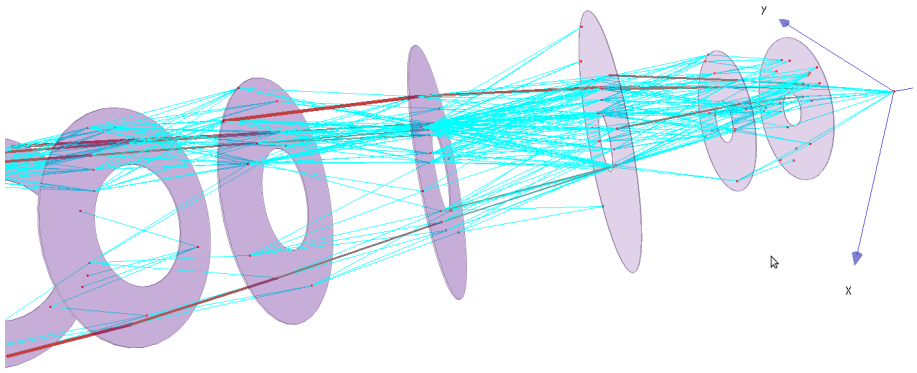


Figure A.32.: After the cellular automaton has performed

Now the 2-hit-segments step is done and it is time to build longer ones: 3-hit-segments. Again the cellular automaton is performed on them using other criteria for 3-hit-segments, for example the compatibility of the radii of the circles they form (which follows from momentum conservation if there are no material effects). After the CA procedure and cleaning, at the end one gets a final track candidate in Figure A.34.

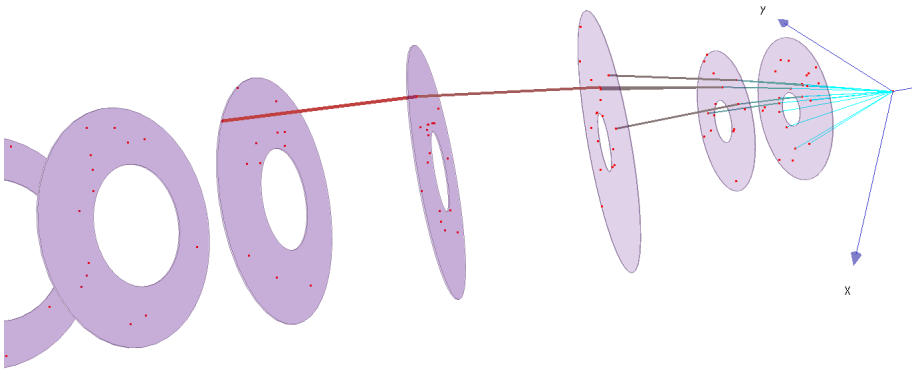


Figure A.33.: After erasing bad segments

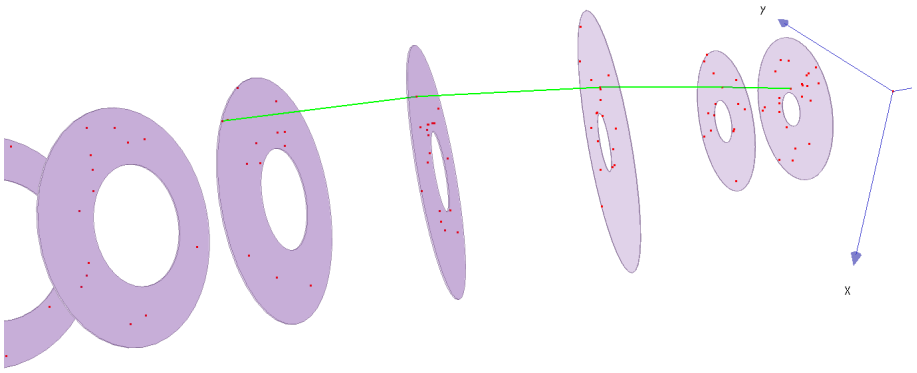


Figure A.34.: Track candidates

In this simplified example there was only low background, no skipped layers etc., which is why it still looked rather clean. Figure A.35 to Figure A.41 depicts the same event including more background and skipping of layers, which gives a denser, but also more realistic example. Also much more tracks can be found if the skipping mechanism is in place.

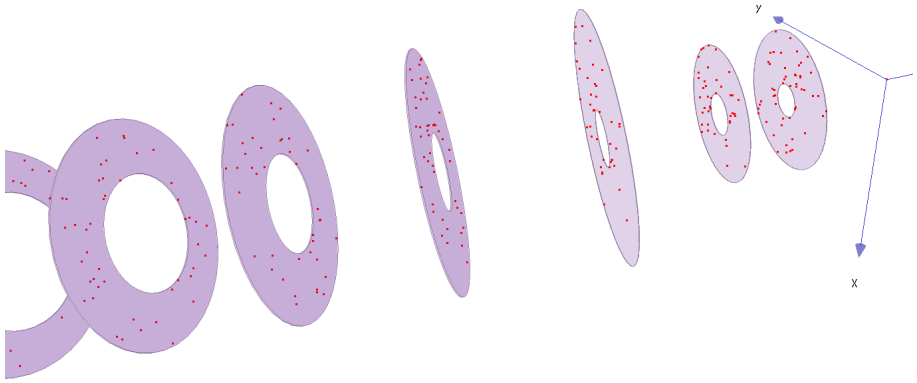


Figure A.35.: The hits

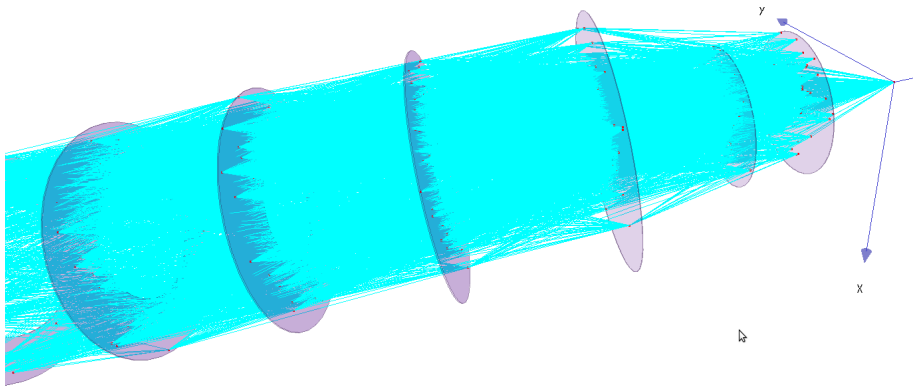


Figure A.36.: Built 2-hit-segments

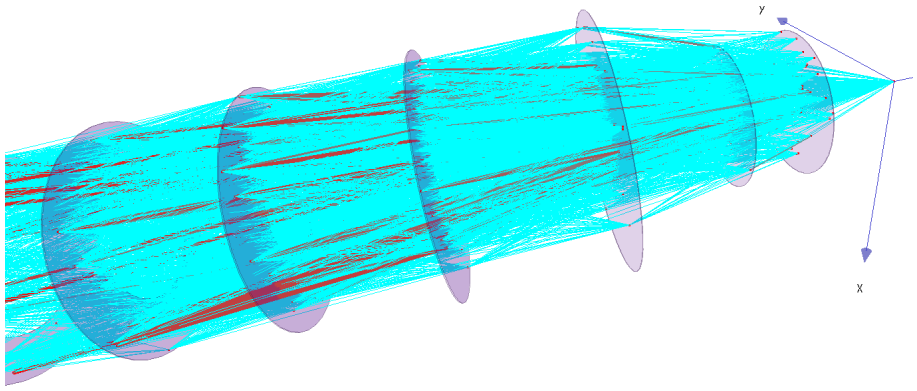


Figure A.37.: The cellular automaton performed with the 2-hit-segments

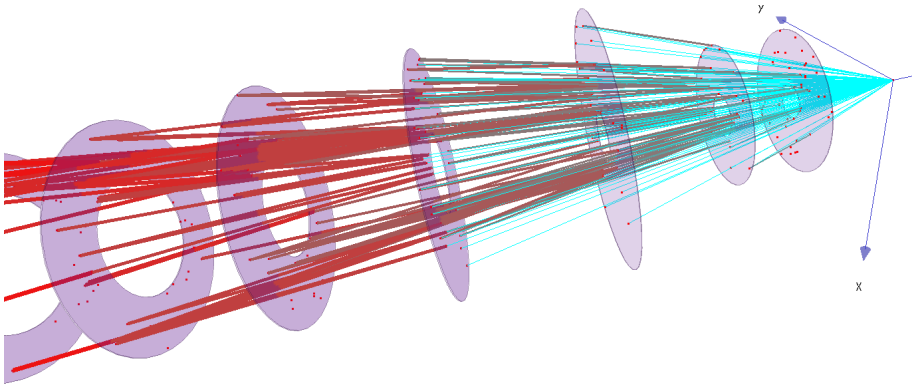


Figure A.38.: After removing segments with bad states

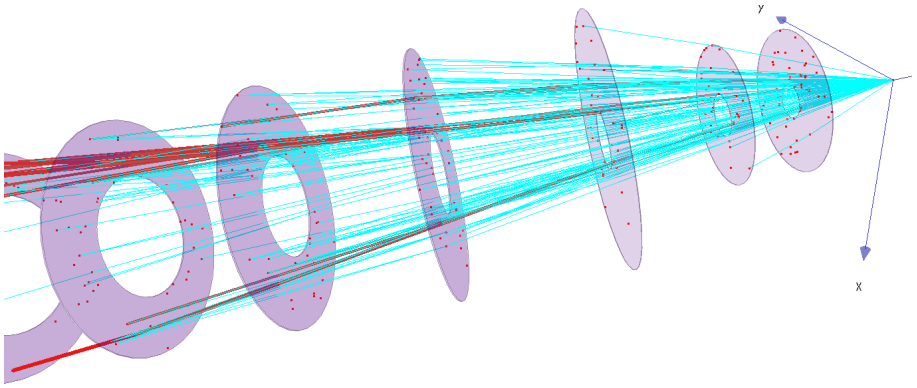


Figure A.39.: The cellular automaton performed with 3-hit-segments

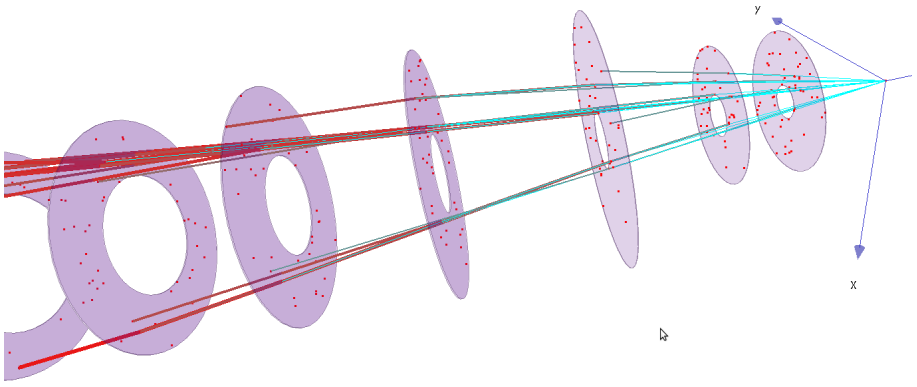


Figure A.40.: After erasing bad states

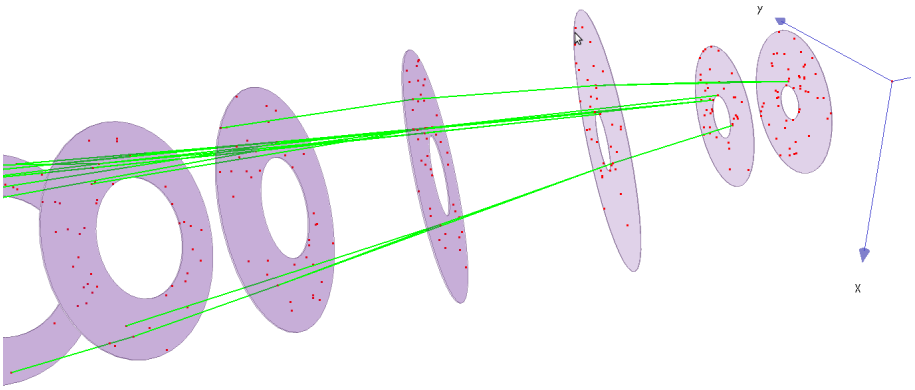


Figure A.41.: The final track candidates

B. Acknowledgements

There are many people I want to thank, without whom writing this thesis would not have been possible:

First I want to thank *Winfried Mitaroff*. He brought me into the subject of high energy physics and put a lot of effort into supervising me. I owe him great thanks for taking so much care of this project and me – his enthusiasm and feedback kept me going.

I also thank *Rudolf Frühwirth* for providing me with great help and insights in algorithms and statistics, for always being a productive and calm supervisor. Thanks to my supervisors, I never felt lost on my way.

Many ideas and great inspiration came from numerous talks with *Jakob Lettenbichler*. I will always remember how fruitful the exchange between different working groups can be.

I want to thank *Felicitas Thorne* and *Moritz Nadler* for all the small things, that help so much – the talks at lunch, the guidance with `ROOT` and the companionship.

And I want to thank all of the High Energy Physics Institute of the Austrian Academy of Sciences – this has been a wonderful year and I've never before worked in such an inspiring atmosphere. Special thanks go to *Christian Fabjan*, *Manfred Krammer* and *Christoph Schwanda* for their administration and financial support.

My gratitudes also go to DESY, which had become my second home for a part of this work, and the people with whom I collaborated there. Especially *Steve Aplin*, with whom I shared the happy and insane moments of writing software algorithms and from whom I learned that life only makes sense backwards while we are under the burden of living it forwards. *Frank Gaede* and *Jan Engels* have been a great help in the ILD collaboration welcoming me so open hearted and supporting me on many frontiers.

Finally I want to mention those who helped me on a personal level. Without my family and their support none of this would have been possible. I deeply thank you.

Last, I want to thank my spouse *Simone* for giving me encouragement and comfort when I needed them, for being patient when I had to leave the country and for giving me a place I was always happy to come back to.

Bibliography

[1] Georges Aad et al. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys.Lett.*, B716:1–29, 2012.

[2] I. Abt, D. Emeliyanov, I. Gorbounov, and I. Kisel. Cellular automaton and Kalman filter based track search in the HERA-B pattern tracker. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 490(3):546–558, 2002.

[3] I. Abt, D. Emeliyanov, I. Kisel, and S. Masciocchi. CATS: a cellular automaton for tracking in silicon for the HERA-B vertex detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 489(1):389–405, 2002.

[4] W. Adam, R. Frühwirth, A. Strandlie, and T. Todorov. Reconstruction of electrons with the Gaussian-sum filter in the CMS tracker at the LHC. *Journal of Physics G: Nuclear and Particle Physics*, 31(9):N9, 2005.

[5] S. Agostinelli et al. Geant4—a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, 2003.

[6] H. Aihara, P. Burrows, and M. Oreglia. Sid letter of intent. *arXiv preprint arXiv:0911.0006*, 2009.

[7] J. Alcaraz. Helicoidal tracks. *L3 Note 1666*, 1995.

[8] P. Azzurri. Track reconstruction performance in CMS. *Nuclear Physics B-Proceedings Supplements*, 197(1):275–278, 2009.

[9] Thomas Bergauer. Lecture: Silicon detectors in high energy physics, 2011.

[10] J. Beringer et al. Review of particle physics. *Physical Review D*, 86(1):010001, 2012.

[11] O. Brüning, P. Collier, P. Lebrun, S. Myers, R. Ostojic, J. Poole, and P. Proudlock. LHC design report. `http://lhc.web.cern.ch/lhc/LHC-DesignReport.html`, 2004.

[12] Karsten Buesser. Why is it desirable to build the ILC in a linear format rather than in a cyclotron format? `http://www.linearcollider.org/about/Why-do-we-need-the-ILC/Ask-a-scientist/`.

[13] Serguei Chatrchyan et al. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys.Lett.*, B716:30–61, 2012.

[14] M.M. Collider. A feasibility study. 1996.

[15] S. Cucciarelli, M. Konecki, D. Kotlinski, and T. Todorov. Track reconstruction, primary vertex finding and seed generation with the pixel detector. *CMS Note*, 2006026, 2006.

[16] ILC Global Design Effort and World Wide Study. International Linear Collider reference design report. `http://www.linearcollider.org/about/Publications/Reference-Design-Report`, 2007.

[17] D. Emeliyanov, I. Gorbounov, and I. Kisel. OTR/ITR-CATS: Tracking based on cellular automaton and Kalman filter. *HERA-B internal note, HERA-B*, pages 01–137, 2001.

[18] The Tevatron Electroweak Working Group for the CDF and DØ Collaborations. Combination of cdf and DØ results on the mass of the top quark using up to 5.8 fb^{-1} of data. `http://tevewwg.fnal.gov/top/TevSum11MtCombo.pdf`, 2011.

[19] 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(2):444–450, 1987.

[20] R. Frühwirth, M. Regler, R.K. Bock, H. Grote, and D. Notz. *Data analysis techniques for high-energy physics*, volume 11. Cambridge University Press, 2000.

[21] Frühwirth and Strandlie. Pattern recognition and reconstruction in Landolt-Börrnstein new series, group I: Detectors for particles and radiation. part 1: Principles and methods vol 21B1 chap 4.3. 2011.

[22] Rudolf Frühwirth. Selection of optimal subsets of tracks with a feed-back neural network. *Computer Physics Communications*, 78:23–38, 1993.

[23] F. Gaede, T. Behnke, N. Graf, and T. Johnson. LCIO-a persistency framework for linear collider simulation studies. *arXiv preprint physics/0306114*, 2003.

[24] F. Gaede and J. Engels. Marlin et al-a software framework for ILC detector R&D. *EUDET-Report-2007-11*, 2007.

[25] Frank Gaede. GEAR - a geometry description toolkit for ILC reconstruction software. `http://ilcsoft.desy.de/portal/software_packages/gear`.

[26] L. Garren. Stdhep 5.05 monte carlo standardization at FNAL. *Fermilab PM0091. Available from http://cepa. fnal. gov/psm/stdhep.*

[27] RL Gluckstern. Uncertainties in track momentum and direction, due to multiple scattering and measurement errors. *Nuclear Instruments and Methods*, 24:381–389, 1963.

[28] ILD Concept Group. The international large detector: Letter of intent. `http://ilcild.org/documents/ild-letter-of-intent/`, 2010.

[29] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA*, 79:2554–2558, 1982.

[30] V. Khachatryan, AM Sirunyan, A. Tumasyan, W. Adam, T. Bergauer, M. Dragicevic, J. Erö, C. Fabjan, M. Friedl, R. Frühwirth, et al. CMS tracking performance results from early LHC operation. *The European Physical Journal C-Particles and Fields*, 70(4):1165–1192, 2010.

[31] W. Kilian, T. Ohl, and J. Reuter. WHIZARD—simulating multi-particle processes at LHC and ILC. *The European Physical Journal C-Particles and Fields*, 71(9):1–29, 2011.

[32] I. Kisel, V. Kovalenko, F. Laplanche, R. Arnold, C. Augier, A. Barabash, D. Blum, V. Brudanin, JE Campagne, D. Dassie, et al. Cellular automaton and elastic net for event reconstruction in the NEMO-2 experiment. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 387(3):433–442, 1997.

[33] IV Kisel and GA Ososkov. An application of cellular automata and neural networks for event reconstruction in discrete detectors. *Cern European Organization for Nuclear Research-Reports*, pages 646–646, 1992.

[34] T. Krämer. Track parameters in LCIO. 2006.

[35] Manfred Krammer and Winfried Mitaroff. Tracking detectors, in "handbook of particle detection and imaging", vol. 1, pp. 265 – 295, springer, berlin-heidelberg. 2012.

[36] P. Lebrun, L. Linssen, A. Lucaci-Timoce, D. Schulte, F. Simon, S. Stapnes, N. Toge, H. Weerts, and J. Wells. The CLIC programme: Towards a staged e+ e-linear collider exploring the terascale: CLIC conceptual design report. *arXiv preprint arXiv:1209.2543*, 2012.

[37] C. Lefevre. Lhc: the guide. `http://cdsweb.cern.ch/record/1092437`, 2008.

[38] J. Lettenbichler, R. Frühwirth, M. Nadler, and R. Glattauer. Low-momentum track finding in Belle II. 2012.

[39] Jakob Lettenbichler. Pattern recognition in the silicon vertex detector of the Belle II experiment. Master's thesis, 2012.

[40] C. Lippmann. Particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 666:148–172, 2012.

[41] R. Mankel. Pattern recognition and event reconstruction in particle physics experiments. *Reports on Progress in Physics*, 67(4):553, 2004.

[42] R. Mankel and A. Spiridonov. Ranger-a pattern recognition algorithm for the HERA-B main tracking system. *Part I: The HERA-B pattern tracker. HERA–B note*, pages 97–082, 1997.

[43] Rainer Mankel and Alexander Spiridonov. The Concurrent track evolution algorithm: Extension for track finding in the inhomogeneous magnetic field of the HERA-B spectrometer. *Nucl.Instrum.Meth.*, A426:268–282, 1999.

[44] G. Moortgat-Pick. News from polarized e-and e+ at the ILC. *arXiv preprint hep-ph/0509099*, 2005.

[45] G. Moortgat-Pick, Abe, et al. Revealing fundamental interactions: the role of polarized positrons and electrons at the linear collider. Technical report, Stanford Linear Accelerator Center (SLAC), 2005.

[46] G. Moortgat-Pick and H. Steiner. Physics opportunities with polarized e- and e+ beams at TESLA. *EPJ direct*, 3(1):1–27, 2001.

[47] S. Myers and E. Picasso. The design, construction and commissioning of the CERN large electron–positron collider. *Contemporary Physics*, 31(6):387–403, 1990.

[48] Bodgan Povh, Klaus Rith, Christoph Scholz, and Frank Zetsche. *Teilchen und Kerne, 7. Auflage.* 2006.

[49] A. Raspereza. LDC tracking package user's manual, 2007.

[50] M. Regler and R. Frühwirth. Generalization of the gluckstern formulas i: Higher orders, alternatives and exact results. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 589(1):109–117, 2008.

[51] M. Regler, M. Valentan, and R. Früwirth. *LiCToy 2.0*, 2008.

[52] T. Schober. Untersuchung der Hough-Transformation als Beispiel eines globalen Algorithmus zur Mustererkennung im HERA-B Spurkammersystem. Master's thesis, Humboldt-Universit. at zu Berlin, 1996.

[53] Yash Shrivastava. Guaranteed convergence in a class of hopfield networks. *IEEE Transactions on Neural Networks*, 3(6):951–961, 1992.

[54] M. Srednicki. *Quantum field theory*. Cambridge University Press, 2007.

[55] A. Strandlie and R. Frühwirth. Track and vertex reconstruction: From classical to adaptive methods. *Reviews of Modern Physics*, 82(2):1419, 2010.

[56] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley Professional, third edition, June 1997.

[57] M.A. Thomson. Particle Flow Calorimetry and the PandoraPFA Algorithm. *Nucl.Instrum.Meth.*, A611:25–40, 2009.

[58] T. Toffoli and N. Margolus. *Cellular automata machines: a new environment for modeling*. MIT press, 1987.

[59] J. Vlissides, R. Helm, R. Johnson, and E. Gamma. *Design patterns: Elements of reusable object-oriented software*. 1995.

[60] G. Weiglein. The LHC and the ILC. *arXiv preprint hep-ph/0508181*, 2005.

Glossary

θ The polar angle w.r.t. the z-axis (origin at center of detector, direction of the beam tube and the magnetic field). 11–13, 78

AIDA **A**dvanced European **I**nfrastructures for **D**etectors at **A**ccelerators, a project funded by the EU's 7^{th} framework programme. 58, 89

ATLAS A detector at the LHC. 25, 51

Belle Detector at the KEKB accelerator. 55

Belle II Detector at the SuperKEKB accelerator, upgrade of Belle. 30, 36, 49, 54–56

C++ An object-oriented programming language, quasi standard in high energy physics. 16, 17, 49, 50, 58–60

CA **C**ellular **A**utomaton, discrete entities evolving in discrete timesteps depending on the local environment. III, 28, 36, 54, 93, 96, 97, 102, 107, 108

CATS Track reconstruction program for the HERA-B vertex detector based on a cellular automaton. 30, 54, 55

CERN European Organization for Nuclear Research, near Geneva, Switzerland/France. 4

CKM **C**abibbo-**K**obayashi-**M**askawa matrix, unitary matrix describing the mixing of quark generations. 2, 55

CLIC **C**ompact **Li**near **C**ollider, project of a linear electron-positron collider. 3, 4, 7, 9, 58, 89, 92

CMS **C**ompact **M**uon **S**olenoid detector, a detector at the LHC. 25, 49, 51, 52, 89

CP Combined charge-conjugation and parity symmetry; its violation expresses an asymmetry between matter and anti-matter. 2, 6, 52

DBD **D**etailed **B**aseline **D**esign, reports being written by the two ILC detector groups. III, IV, 68, 89

DESY **D**eutsches **E**lektronen-**Sy**nchrotron, at Hamburg, Germany. 4, 8, 113

ECAL **E**lectromagnetic **Cal**orimeter. 10, 25

FTD **F**orward **T**racking **D**etector, a silicon sub-detector in the ILD covering the tracking area around the beam pipe. III, IV, 12, 13, 15, 27, 28, 31, 32, 36, 49, 50, 52, 55, 56, 61–64, 68, 70, 71, 78, 82, 92, 106, 107

GEANT4 A toolkit for simulating the passage of particles through matter[5]. 16

GSL **G**NU **S**cientific **L**ibrary. 57

HCAL **H**adronic **Cal**orimeter. 10, 25

HERA **H**adron-**E**lektron-**R**ing-**A**nlage, a proton-electron collider at DESY. 4

HERA-B Spectrometer operated in the halo of the HERA proton beam, used to measure CP-violation. 49, 52

Higgs boson A fundamental spin-0 particle, the last missing part of the SM; possibly discovered 2012 at LHC. III, 2–4, 51

HNN **H**opfield **n**eural **n**etwork, a recurrent neural network with symmetric connections. III, 59

ILC **I**nternational **L**inear **C**ollider, project of a linear electron-positron collider. III, IV, 1, 3–5, 7–10, 27, 58, 89, 92

ILD **I**nternational **L**arge **D**etector, one of two validated detector concepts for the ILC. III, IV, 1, 8, 10–12, 15–17, 19, 20, 25, 28, 30, 31, 36, 49, 51, 52, 54–58, 61, 62, 64, 70, 78, 82, 87, 89, 91, 92, 106, 107, 113

IP **I**nteraction **P**oint, spot where beam particle collisions occur. 30–34, 50, 51, 62, 78, 93, 96, 98, 100, 102, 103, 106, 107

KEK A high energy accelerator research organization in Tsukuba, Japan. 52

KEKB A high luminosity electron positron synchrotron used as B-factory at KEK. 4

KF **K**alman **f**ilter, a recursive and locally linear estimator, equivalent to the global least squares method. III

LCIO **L**inear **C**ollider **I/O**, a data model and persistency system containing classes for describing high energy physics events. 16, 17, 20, 37, 57, 61, 64

LEP **L**arge **E**lectron-**P**ositron Collider, an electron-positron synchrotron at CERN. 4, 7

LHC **L**arge **H**adron **C**ollider, a proton-proton and ion-ion synchrotron at CERN and successor to LEP. III, IV, 3–7, 11

LoI **L**etter **o**f **I**ntent, proposals submitted by the then 3 ILC detector groups in 2009, two of which were eventually validated. 49, 73, 82

Marlin **M**odular **a**nalysis and **r**econstruction for the **Lin**ear collider, framework for event reconstruction at ILD. 16, 17, 57, 61, 62, 87

Mokka Program package for detector simulation based upon GEANT4, used by ILD. 16, 17, 64, 68

MSSM **M**inimal **S**upersymmetric extension of the **S**tandard **M**odel. 3

QCD **Q**uantum **C**hromo**d**ynamics, theory of the strong interaction, part of the SM. 2, 5

QED **Q**uantum **E**lectro**d**ynamics, theory of the electromagnetic interaction; as unified with the weak interaction (electroweak theory) part of the SM. 1

QI **Q**uality **i**ndicator, a test value between 0 and 1, used by the HNN. 46, 74, 77

RF **R**adio **f**requency, a wide range of oscillation frequencies from 30 kHz upwards, originally used by radio; in acceleration cavities, e.g. 1.3 GHz (ILC), 12 GHz (CLIC). 4–9

ROOT A data analysis system developed for and mainly used in high energy physics. 57, 113

SET **S**ilicon **E**xternal **T**racker, a silicon sub-detector in the ILD just outside the TPC. 11, 12

SiD **Si**licon **D**etector, one of two validated detector concept for the ILC description. 8, 10, 11, 51

SIT **S**ilicon **I**nner **T**racker, a silicon sub-detector in the ILD just inside the TPC. 11, 12, 50

SLAC **S**tanford **L**inear **A**ccelerator **C**enter, at Stanford University, CA, USA. 52

SM **S**tandard **M**odel, the present theory of particle physics. III, IV, 1–3

SuperKEKB A high luminosity electron positron synchrotron used as B-factory at KEK, successor to KEKB. 55

SUSY **Su**per**sy**mmetry, a possible theoretical extension of the SM. 3, 6

SVD **S**ilicon **V**ertex **D**etector, the outer part of the silicon tracking system at Belle II, between a pixel vertex detector and a central drift chamber. 49, 54–56

SVN Apache Subversion, a software versioning system. 92

TPC **T**ime **P**rojection **C**hamber, a gaseous tracking detector. III, IV, 8–13, 19, 27, 37, 49, 50, 55, 56, 64, 78, 82, 92, 107

vertex Point in space of the beam interaction (primary vertex), or of the decay of an unstable particle (secondary vertex) - hence the origin of particle trajectories. 9, 12, 16, 17, 19–21, 26, 43, 51, 56, 64, 70, 78, 96, 106, 107

VTX Multi-layer pixel-**V**er**t**e**x** detector, a sub-detector in the ILD, close around the beam tube covering the IP region. 12, 50, 52

Whizard A modern event generator package, for SM physics and beyond. 15, 64