Karlsruhe Institute of Technology

New Software Techniques in Particle Physics and Improved Track Reconstruction for the CMS Experiment

Zur Erlangung des akademischen Grades eines
DOKTORS DER NATURWISSENSCHAFTEN
von der Fakultät für Physik des
Karlsruher Instituts für Technologie (KIT)
genehmigte

DISSERTATION

von

Dipl.-Phys. Thomas Hauth
aus Karlsruhe

Tag der mündlichen Prüfung: 11. Juli 2014

Referent: Prof. Dr. G. Quast
Institut für Experimentelle Kernphysik

Korreferent: Prof. Dr. M. Feindt
Institut für Experimentelle Kernphysik

CERN Supervisor: Dr. A. Pfeiffer
CERN

Chapter 1

Introduction

The history of human discovery of the physical laws was a continuous interplay between theoretical considerations and experimental observations. In some cases during the ages of discovery, meticulous observations were necessary to arrive at theoretical insights and finally to formulate general laws. A particularly good example is the work of danish astronomer Tycho Brahe who observed and recorded the motion of the celestial bodies for years. More than ten years later, Brahe's measurements served Johannes Kepler as an important source to formulate his laws of planetary motion.

In this manner, discovery in modern particle physics is conducted as a constant exchange between theory and experiment and led to the standard model of particle physics. This framework describes the elementary particles and their interactions with each other. Chapter 2 will provide an overview of the standard model of particle physics.

Still today, experimental observations lead to new questions. One example are the measurements of the Planck satellite and previous experiments, which point to the existence of dark matter in the universe [1]. A theoretical description of the nautre of this matter still needs to be found. Theories like supersymmetry have been proposed and need to be validated experimentally.

To address these and other open questions in physics, the Large Hardon Collider (LHC) was built at the CERN research center in Geneva, Switzerland. It is designed to accelerate protons and heavy ions and bring them to collision at four interaction points. An introduction to CERN, the LHC and its research goals will be given in chapter 3.

The Compact Muon Solenoid (CMS) is one of the experiments recording the particle collisions and thus allows to perform observations, which can be compared with theoretical predictions. To arrive at a full understanding of the measured collisions, an event reconstruction is performed by software. The detector hardware and the computing and software systems involved in this process will be

described in chapters 4, 5, 6 and 7.

In order to analyze the signals of Higgs boson decays and investigate possible supersymmetric models, as many collision events as possible need to be recorded by the CMS detector and reconstructed by software. To achieve this, the capacity of existing computing resources need to be fully exploited.

Chapter 8 describes the computing challenges faced by high energy physics and chapter 9 presents an overview of modern computing architectures and their capabilities. Chapter 10 will present the work done as part of this thesis, to optimize simulation and reconstruction software to better exploit the vector capabilities of modern CPUs.

To cope with the increased amount of recorded data expected during the upcoming years of the LHC's operation, the parallel computing power of Graphics Processing Units (GPU) are a promising hardware option. Chapter 11 describes the specifics of GPUs, how event reconstruction algorithms can be adapted to this kind of hardware and the performance gains which can be achieved.

While the increase of computing performance is necessary, also an improvement of the reconstruction accuracy is a desirable property to allow for high-precision measurements of physics processes with the CMS detector. Chapter 12 will describe how a detailed geometrical model of the CMS detector can improve the reconstruction quality of particle tracks. Chapter 13 will demonstrate this novel technique by reconstructing the mass of a short-lived Kaon in Monte Carlo simulation and measured events. The conclusion in chapter 14 will summarize the findings of this work and outline possible next steps.

Contents

Chapter 2

Theoretical Principles

2.1. The Standard Model of Particle Physics

Even in the early days of human development, people wanted to understand the nature surrounding them. One important element of this endeavour is to discover and study the most basic building blocks of nature.

After 2500 years of human progress, the Standard Model of Particle Physics [2] forms a sophisticated framework which describes the basic building blocks of nature and the forces between them. The particles specified by the Standard Model can be sorted into three categories: leptons and quarks, which act as the basic building blocks, and vector bosons which mediate forces.

2.2. Elementary Particles

Lepton and quark particles occur grouped in three generations, wherein each particle of a higher generation exhibits a larger mass than its corresponding previous generation particle. Apart from the flavour quantum number, all quantum numbers are identical between the different generations of the same type of particle. All matter observed in our everyday life is made up of particles belonging to the first generation. Particles of the second and third generation can for example be observed in high energy physics (HEP) experiments like the Large Hadron Collider [3](LHC) and cosmic rays hitting the earth's surface.

All kinds of leptons have a spin of $\frac{1}{2}$ in common. In contrast, one half of all kinds of leptons have an electrical charge of one elementary unit, while the other half is uncharged. Therefore, one kind of uncharged lepton, named neutrino, can be assigned to every type of charged lepton. These charged leptons of each generation are named electron, muon and tau. In total, this results in six kinds of leptons and their six corresponding anti-particles. The full list of all lepton types and

some of their most important properties can be found in table 2.1.

Table 2.1.: List of the three lepton generations and some of their most important properties [4].

Generation	Name	Symbol	Mass	El.Charge [e]
First	Electron	e	0.51 MeV	-1
Generation	Electron Neutrino	ν_e	$< 2\,\text{eV}$	0
Second	Muon	μ	105.66 MeV	-1
Generation	Muon Neutrino	ν_μ	$< 2\,\text{eV}$	0
Third	Tau	τ	1776.82 MeV	-1
Generation	Tau Neutrino	ν_τ	$< 2\,\text{eV}$	0

In a similar manner to leptons, six flavours of quarks and their corresponding anti-particles exist. While leptons have an integer electrical charge, quarks have a charge of either $-\frac{1}{3}e$ or $\frac{2}{3}e$. A full listing of all quarks and their most important properties can be found in table 2.2.

Table 2.2.: List of the three quark generations and some of their most important properties [4][5].

Generation	Name	Symbol	Mass	El. Charge [e]
First	Down	d	$4.8^{+0.07}_{-0.03}\,\text{MeV}$	$-\frac{1}{3}$
Generation	Up	u	$2.3^{+0.07}_{-0.05}\,\text{MeV}$	$\frac{2}{3}$
Second	Strange	s	$95 \pm 5\,\text{MeV}$	$-\frac{1}{3}$
Generation	Charm	c	$1.275 \pm 0.025\,\text{GeV}$	$\frac{2}{3}$
Third	Bottom	b	$4.18 \pm 0.03\,\text{GeV}$	$-\frac{1}{3}$
Generation	Top	t	$173.34 \pm 0.27 \pm 0.71\,\text{GeV}$	$\frac{2}{3}$

The Pauli exclusion principle also applies to quarks, as they are spin-$\frac{1}{2}$ particles and therefore fermions. This law states that the same quantum state can not be held by two fermions at the same time. Nevertheless, various particles comprised of the same quark types, like the Δ^{++} particle consisting of three up quarks, have been observed [2].

This discovery made the introduction of an additional quantum number for quarks necessary: the colour charge. This quantum number can either take the colours “red”, “blue” or “green” but should only be understood as a way to express a quantum property and is not related to any visual perception. Using the

colour charge, every up quark constituent of the Δ^{++} particle can be assigned a different colour quantum number and thereby satisfying the Pauli exclusion principle. Furthermore, the colour charge can be used to describe how quarks can form composite particles by demanding that only colour-neutral composite particles are occurring in nature [2]. One way to create colour-neutral composite particles made up of quarks is to have each quark carry a different colour. Analogues to the theory of colours, overlaying "red", "blue" and "green" will result in "white" and therefor a colour-neutral particle. Composite particles formed of three quarks are named baryons and make up the bulk of the matter we observe in our daily lives. Two prominent baryons are the proton [6], comprised of two up and one down quark and the neutron [6], formed by two down and one up quark.

Another possibility to form composite particles is to combine only two quarks, wherein the second quark is the anti-quark of the first one. In this configuration, the anti-quark will have the anti-colour of the first quark and therefor the composite particle has an overall neutral colour charge. These types of configurations containing only two quarks are named Mesons. Both Baryons and Mesons can be grouped in the more general term Hadron, which are composite particles formed by quarks.

2.3. Mathematical Formulation of the Standard Model

The mathematical framework used to formulate the Standard Model is a relativistic quantum field theory. In this theory, observable particles are mathematically described by excited states of underlying quantum fields. The mathematical is the Lagrangian formulation of classical mechanics. Therein, the Lagrange function is defined by [2]

$$L = T - U \tag{2.1}$$

where T stands for the kinetic and U for the potential energy. This expression is a function of the generalized coordinates q_i and their time-derivatives $\dot{q}_i$. In this formulation the fundamental law of motion is expressed by the Euler-Lagrange equation [2]:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) = \frac{\partial L}{\partial q_i} \tag{2.2}$$

2.3.1. Quantum Field Theory

The formulation of the quantum field theory uses the field variables ϕ_i and depends on the four vector of a particle. This four vector representation does not only contain three space-like components, like in classical mechanics, but also one time-like component. The Lagrangian density $\mathcal{L}$, which depends on the field variables

ϕ_i, can be plugged into a modified Euler-Lagrange equation [2]:

$$\partial_\mu \left(\frac{\partial \mathcal{L}}{\partial \left(\partial \phi_i \right)} \right) = \frac{\partial \mathcal{L}}{\partial \phi_i} \tag{2.3}$$

Different to the classical Euler-Lagrange equation, the time derivative has been replaced by the derivatives of all vector components ∂_μ. As a relativistic theory, both space-like and time-like coordinates must be treated equally [2].

With this Lagrangian formulation, the Dirac equation can be derived. It is a relativistic quantum mechanical wave equation for spin-$\frac{1}{2}$ particles. To this end, the four spinor, which holds four components, is defined as:

$$\psi = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{pmatrix} \tag{2.4}$$

Using this spinor, the Lagrangian density can now be written as [2]

$$\mathcal{L} = i\bar{\psi}\gamma^\mu \partial_\mu \psi - m\bar{\psi}\psi \tag{2.5}$$

wherein γ^μ are the gamma matrices $\gamma^0, \gamma^1, \gamma^2, \gamma^3$ and m is the mass of the particle.

In this formulation, the spinor ψ and its adjoint spinor $\bar{\psi}$ are considered as separate field variables and the Euler-Lagrange equation 2.3 can be applied [2]. In case of the adjoint spinor $\bar{\psi}$ the use of the Euler-Lagrange equation leads to the Dirac equation:

$$i\gamma^\mu \partial_\mu \psi - m\psi = 0 \tag{2.6}$$

Accordingly, the application of the Euler-Lagrange equation on the spinor ψ leads to the adjoint version of the Dirac equation.

2.3.2. Local Gauge Invariance

An important component of the mathematical formulation of the Standard Model is the gauge invariance. It gives rise to the fields and therefore the force meditating particles associated with them. One application of gauge invariance, and the results this implies, will be given in the following using the U(1) symmetry group [7] and $e^{i\theta}$ as one specific member of this group.

In case of global gauge invariance, the phase factor θ is required to be the same for all points in space time. In the case of local gauge invariance, the phase factor is depending on the concrete location in space-time and it is expressed as $\theta(x)$.

After evaluation of a global gauge transformation on the Dirac Lagrangian density 2.5, it becomes obvious that it is invariant under the transformation:

$$\psi \rightarrow e^{i\theta}\psi \tag{2.7}$$

However, this does not hold when a local gauge transformation is applied:

$$\psi \rightarrow e^{i\theta(x)}\psi \tag{2.8}$$

This transformation will result in an additional term which gets added to the Lagrange density

$$\mathcal{L} \rightarrow \mathcal{L} - \partial_\mu \theta \bar{\psi} \gamma^\mu \psi \tag{2.9}$$

In order to arrive at a local gauge invariant Lagrange density again, an additional *gauge field* A_μ needs to be integrated [2] in order to cancel out the extra term in equation 2.9.

$$A_\mu \rightarrow A_\mu - \partial_\mu \frac{1}{q}\theta\,(x) \tag{2.10}$$

wherein q denotes the particle charge.

Utilizing the extra gauge field, the Lagrange density can now be expressed [2] as a locally gauge invariant version

$$\mathcal{L} = \left[i\bar{\psi}\gamma^\mu \partial_\mu \psi - m\bar{\psi}\psi\right] - \left[\frac{-1}{16\pi} F^{\mu\nu}F_{\mu\nu}\right] - \left[\left(q\bar{\psi}\gamma^\mu\psi\right) A_\mu\right] \tag{2.11}$$

where $F^{\mu\nu} = \partial^\mu A^\nu - \partial^\nu A^\mu$. One necessary condition to enable this local gauge invariant formulation to work is the requirement for the gauge field to be massless ($m_A = 0$). The new field introduced by this reformulation can be identified as the electromagtnic field, whose force is meditated via massless photons.

2.4. Elementary Interactions

As outlined in the previous section with an example of the U(1) symmetry group, group theory is used to describe all the elementary interactions formulated by the Standard Model. For the strong interaction, the SU(3) symmetry group [7] is used and the SU(2)×U(1) symmetry group is utilized for the formulation of the electroweak interactions.

2.4.1. Electromagnetic Interaction

As illustrated in the previous example, the U(1) symmetry group is used to formulate the electromagnetic interactions and the work in this area is summarised under the name Quantum Electrodynamics [2] (QED). The gauge boson mediating the electromagnetic interactions is the massless photon (γ). It can propagate freely in space at the speed of light and therefore the range of electromagnetic interaction is infinite.

Table 2.3.: List of gauge bosons which mediate elementary forces and some of their most important properties [4].

Gauge Boson	Symbol	El. Charge [e]	Mass [GeV]	Interaction
Photon	γ	0	0	electromagnetic
Gluon	g	0	0	strong
Z boson	Z^0	0	91.1876 ± 0.0021	(neutral) weak
W boson	$W^{\pm}$	± 1	80.399 ± 0.023	(charged) weak

2.4.2. Weak Interaction and Electroweak Unification

The weak force is not meditated by one, but by three gauge bosons, the electrically neutral Z boson and two charged W bosons. As visible in table 2.3, these bosons have a significant mass which stands contrast to the massless bosons of the electromagnetic and strong interactions. This specific feature of the Z and W bosons is due to their stark coupling to the Higgs field.

The weak interaction can either be meditated through a charged-current via the W bosons or as a neutral current via the Z boson. Addtionally, the range of weak interactions is limited due to the short lifetime of both the Z and W bosons.

The Glashow, Weinberg, and Salam (GWS) theory [7] allows for a unification of both the electromagnetic and weak interactions. In this theory, both interactions stem from a common source, the electroweak force which is mediated by the four massless gauge-bosons W^+, W^-, W^0 and B^0. The Higgs mechanism [2] leads to a spontaneously broken symmetry which gives rise to the specific electromagnetic and weak bosons described before.

2.4.3. Strong Interaction

Quantum Chromodynamics (QCD) is the theory used to formulate the strong interaction. It uses the SU(3) symmetry group which results in eight gauge bosons, named gluons, which mediate the strong force.

One possible representation [6] of these eight gluons is

$$r\bar{g}, r\bar{b}, g\bar{b}, g\bar{r}, b\bar{r}, b\bar{g}, \frac{1}{\sqrt{2}}\left(r\bar{r} - g\bar{g}\right), \frac{1}{\sqrt{6}}\left(r\bar{r} + g\bar{g} - 2b\bar{b}\right) \tag{2.12}$$

The ninth gluon resluting from using the SU(3) symmetry group is colour-neutral: all contained colours are cancelled by their respective anti-colours

$$\frac{1}{\sqrt{3}}\left(r\bar{r} + g\bar{g} + b\bar{b}\right) \tag{2.13}$$

The configuration of this colour-neutral gluon has not been observed in nature [2] and is therefore not included in the list of gauge bosons of the strong force.

As one gluon interacts with other colour-charged particles, like quarks and also gluons themselves, an important property of QCD becomes visible: the self-coupling of gluons and the asymptotic freedom it entails. While the strong force is relatively weak at very short distances, its strength increases for larger distances. This leads to an effect named color confinement, which prevents single quarks or gluons to be observed directly. In case the distance between a quark anti-quark pair is increased, it becomes energetically more favorable to produce a new quark anti-quark pair from the vacuum and recombine with the already existing pair. This process of recombination of color-charged quarks and gluons to colour neutral particles is called hadronisation. Only color-neutral particles, like mesons and baryons, can propagate freely in space and can in turn be observed by particle detectors.

2.5. Cross Section and Luminosity

In particle physics experiments, the cross section represents a hypothetical area that expresses the probability of a particle interacting. Cross section are commonly given in the unit barn, which can be converted into the metric system with $1\,\text{barn} = 10^{-28} m^2$.

In the case of two particles colliding, the cross section of a physics process can be determined by counting the occurrence of the interaction. To arrive at experiment-independent results, this quantity must be normalized using the instantaneous luminosity L which expresses the number of particles per unit time and unit area [2].

$$\mathrm{L} = \frac{\text{number of particles}}{\text{unit area} \times \text{unit time}} \tag{2.14}$$

Now, the integrated luminosity $\mathcal{L}$ can be defined as the instantaneous luminosity integrated over time:

$$\mathcal{L} = \int L dt \tag{2.15}$$

With the integrated luminosity $\mathcal{L}$ used for normalization, the specific cross section of a physics process in a particle collider can be expressed as [2]:

$$\sigma = \frac{N_{\text{inter}}}{\mathcal{L}} \tag{2.16}$$

here N_{inter} is the total number of observed interactions during the measurement period and $\mathcal{L}$ is the corresponding integrated luminosity during this time.

The computed cross sections of various physics processes in proton-(anti)proton particle colliders as a function of the center-of-mass energy can be seen in figure 2.1. Both the maximum design center-of-mass energy of the Tevatron and LHC colliders are marked in the diagram. The figure also shows, that the cross section of possible Higgs mass scenarios is many magnitudes smaller than the particle jet production cross sections center-of-mass energy. To study these rare events, particle colliders with a large center-of-mass energy and a high collision rate are necessary in order to provide sufficient Higgs processes. The next chapters will discuss some of the technical challenges posed by these requirements and present possible solutions.

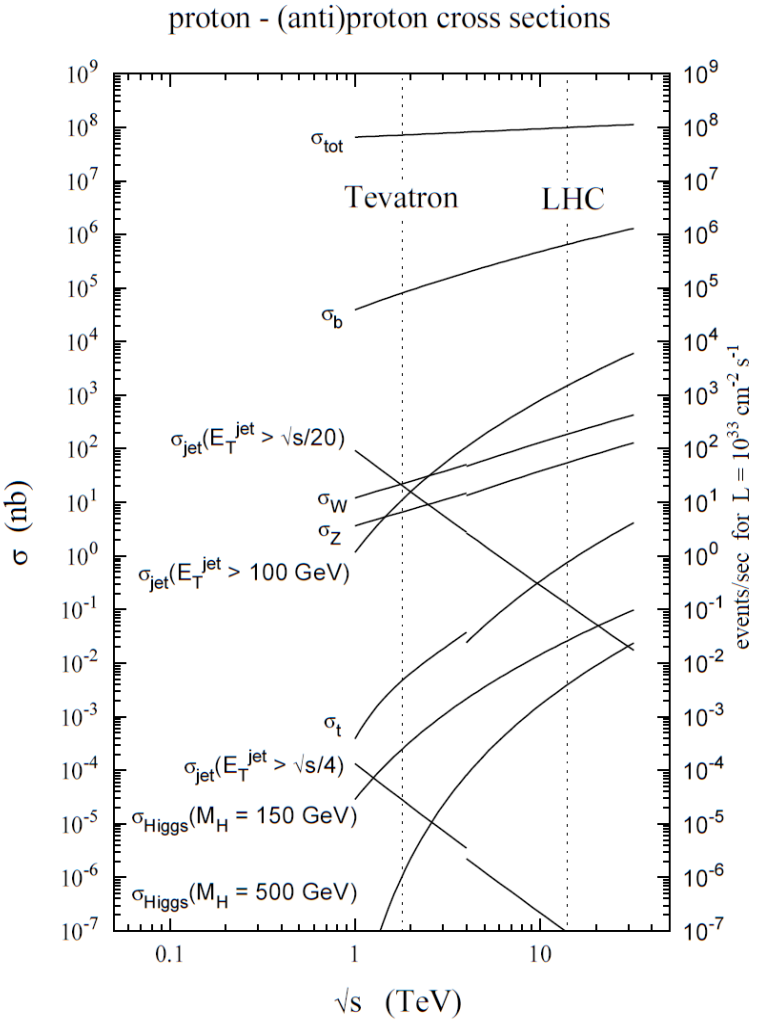


Figure 2.1.: The computed cross sections of various physics processes as a function of the center-of-mass energy [8].

2.6. Particles in Matter

The knowledge about the interaction of energetic particles with matter is an important ingredient to most of today's particle physics experiment. Whether the particles passing matter are the actual focus of the experiment or particle-matter interaction is used in elaborate measurement devices: a deep understanding and mathematical modelling of these processes is of particular importance.

This introduction will focus on high-energetic muons and pions to highlight the processes involved and detail how they can be described by mathematical models. A detailed discussion of the material interaction of photons, electrons, protons and heavier ions can be found in [9].

2.6.1. Electronic and Radiative Contributions to the Energy Loss

The mean energy loss per unit path length of muons can be separated into an electronic and a radiative part [10]:

$$\left\langle -\frac{dE}{dx} \right\rangle = a(E) + b(E)E \tag{2.17}$$

where E is the total energy, $a(E)$ the electronic part and $b(E)E$ the radiative contribution.

The radiative processes can be further spilt up into Bremsstrahlung, pair production and photo-nuclear interactions [10]:

$$b \equiv b_{brems} + b_{pair} + b_{photo} \tag{2.18}$$

For most materials and $E \leq 100\,\mathrm{GeV}$, $b(E)E$ is less than 1% [10]. For this reason, this introduction will focus on the electronic contribution to the energy loss, which is the dominant effect on the charged particles treated in this thesis.

Figure 2.2 shows the stopping power of copper, defined as $S = -dE/dx$, over a wide range of muon energies. The vertical grey bars indicate the boundary region between various approximations. Of these, only the Bethe method will be discussed in the following, as it is able to provide an approximation for the energy range used in this thesis. More details on the other approximations can be found in [4].

2.6.2. Bethe Formula

The electronic contribution to the energy loss experienced by muons is originating from interactions between the interaction of the traversing muon and the electrons

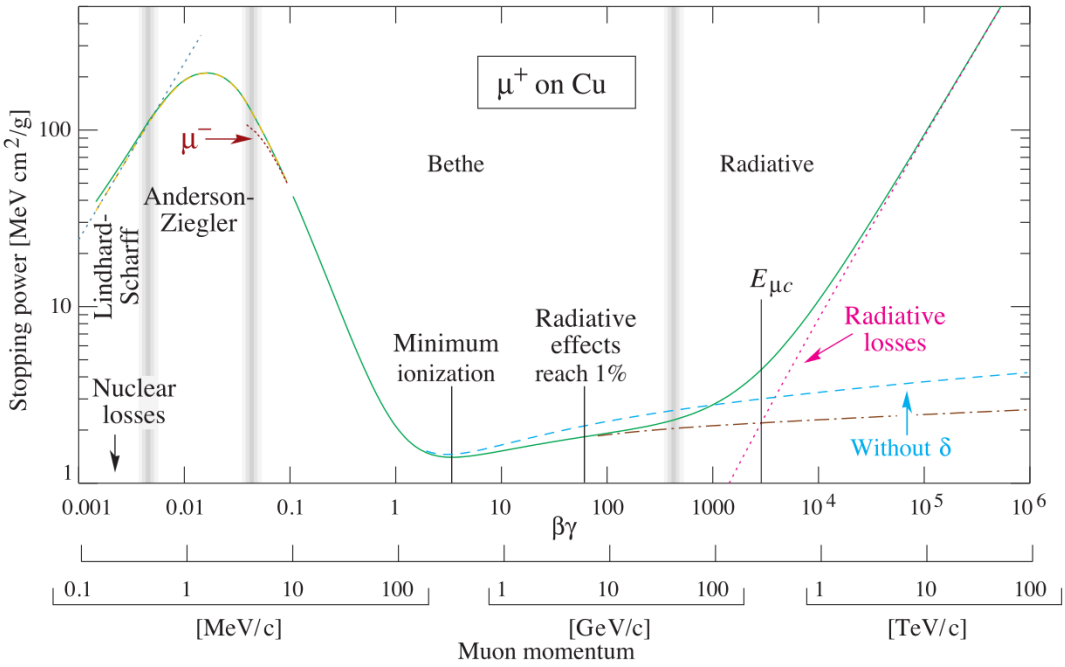


Figure 2.2.: Stopping power for positive muons in copper. The solid line indicates the total stopping power and the vertical grey bars indicate the boundary region between various approximations. [4].

Table 2.4.: Description of the variables used in the Bethe formula 2.19

Symbol	Unit	Description
c	$\frac{m}{s}$	Speed of light in vacuum
A	$g\,mol^{-1}$	Atomic mass of observer
m_e	MeV	Eletcron mass
N_a	mol^{-1}	Avogadro constant
r_e	fm	Classical electron radius
$\frac{K}{A}$	$2.2 \cdot 10^{35}$	$4\pi N_a r_e^2 m_e c^2/A$
z	1	charge, z = -1 for muons
β	1	$\beta = \frac{v}{c}$
γ	1	Lorentz factor
T_{max}	MeV	Maximum possible energy transfer on a single electron in a single collision
I	eV	Mean excitation energy
$\delta(\beta\gamma)$	1	Density correction

of the rest atoms. These electrons are either excited to a higher state of the atom or completely removed, resulting in ionization.

The mean energy loss expected from this effect is described by the Bethe formula for relativistic particles .

$$\langle -\frac{dE}{dx} \rangle_{electronic} = Kz^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} ln \frac{2m_e c^2 \beta^2 \gamma^2 T_{max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right] \tag{2.19}$$

One important correction applied to the Bethe formula is to account for the so-called *density effect* [4]. The term $\frac{\delta(\beta\gamma)}{2}$ subtraced in formula 2.19 from the average energy loss accounts for this effect.

Especially important for particle physics applications, where thin layers of detector materials are used as means to measure passing particles are statistical fluctuations of the energy loss, as described by Landau [4]. In this model, the energy loss in thin layers is described by a gaussian distribution with a long tail generated by rare, but very large energy losses. In mathematical terms, this behaviour is expressed as a probability density function following the Landau distribution.

The energy loss following this Landau distribution can be quantified by the most probable value (MPV) in a certain material and layer thickness. In theory, the mean value cannot be computed, as the tail of the Landau distribution extends to infinity. However, in real experiments, the number of measurements is finite, the actual energy transfer is limited and the mean can be computed.

The left plot in figure 2.3 illustrates the energy loss distribution of pions with an

energy of 500 MeV in a silicon layer of thicknesses ranging from 60 µm to 640 µm. Both the mean energy loss rate and the most probable value $\Delta p/x$ for a silicon layer thickness of 640 µm are marked in the plot.

The right plot in figure 2.3 shows the ratio between the most probable energy loss and the mean energy loss at minimum ionization for various detector thicknesses.

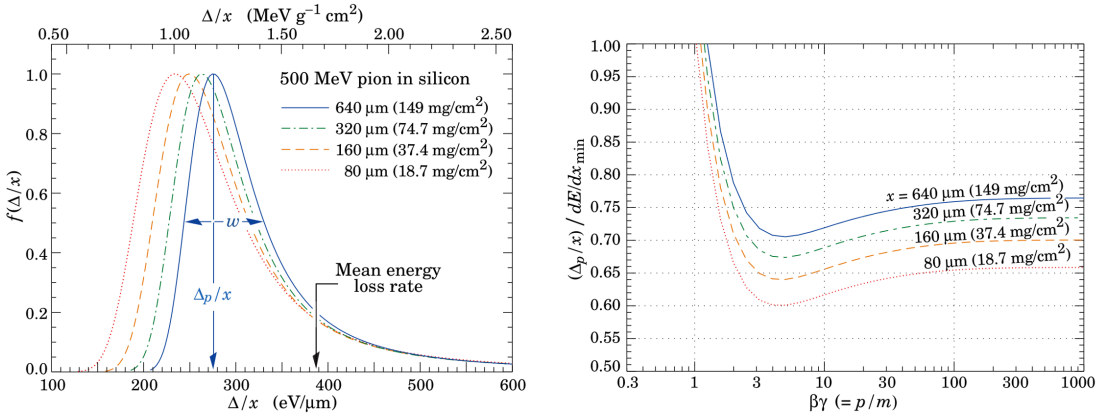


Figure 2.3.: Left plot: energy loss distribution of pions with an energy of 500 MeV in a silicon layer of thicknesses ranging from 60 µm to 640 µm.
Right plot: ratio between the most probable energy loss and the mean energy loss at minimum ionization for various detector thicknesses [4].

Chapter 3

The Large Hadron Collider

CERN is the European organization for nuclear research and has been established in 1954 to promote peaceful research in the field of nuclear physics among its member states. In 2014, 20 countries located on the European continent are member states of CERN. The total number of CERN member states is 21, with Israel joining CERN in 2013 as a full member state. Other countries hold an observer status, for example the United States of America and the Russian Federation, and are involved in experimental work at CERN, as well.

The Large Hadron Collider (LHC) [3] [11] is a particle accelerator complex which is located at CERN near Geneva and spans across the French and Suisse border.

The LHC accelerates two beams of protons in opposite directions along the accelerator ring and focuses these beams on to each other in the so-called interaction points. In its previous operational phases, the LHC was running with a center-of-mass energy of $\sqrt{s} = 7\,\text{TeV}$ and $\sqrt{s} = 8\,\text{TeV}$ and will be increased to the design value of $\sqrt{s} = 14\,\text{TeV}$ in the coming years. Already with these energies, the LHC machine passed all previous collider experiments in terms of the maximally achieved center-of-mass energy. Before the LHC, the Tevatron [13] proton-antiproton collider at the Fermi National Accelerator Laboratory near Chicago achieved the highest center-of-mass energy with $\sqrt{s} = 1.96\,\text{TeV}$.

The Large Electron-Positron Collider(LEP) [14] was operated from 1989 to 2000 at CERN and the machine and the scientific team provided many important scientific contributions. One of the most prominent is the measurement of the Z boson mass with an unparalleled accuracy [15]. As part of the LEP's construction, a 26.7 km long underground tunnel was excavated in a depth ranging from 56m to 170m. This tunnel was reused for the LHC once the LEP collider was decommissioned in 2000.

One important feature of the LHC design is to employ superconducting magnets operated at a temperature below 2 K which can generate a magnetic field of up

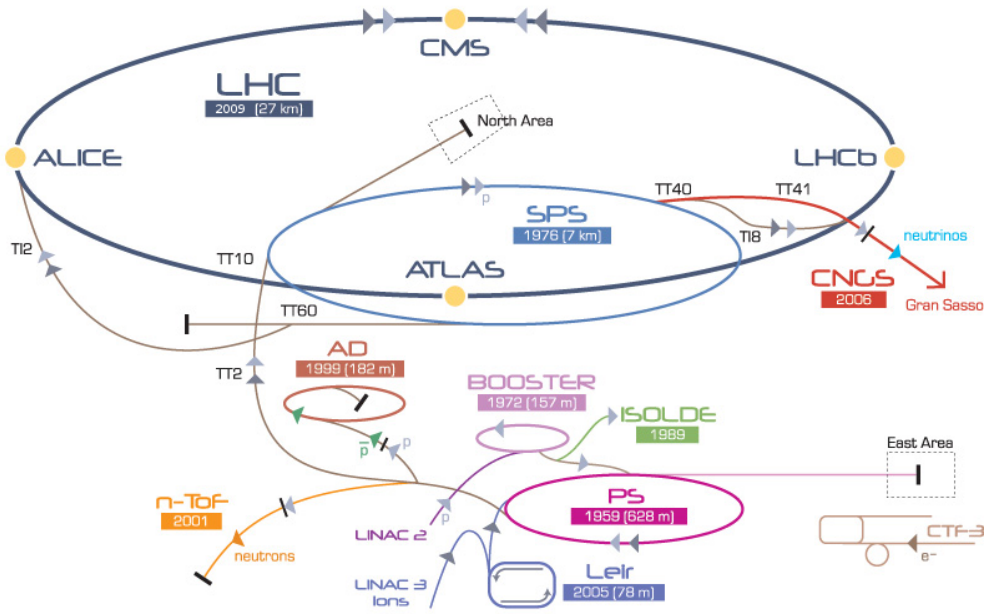


Figure 3.1.: Schematic of the LHC accelerator complex with the four main experiments and the PS and SPS pre-accelerators. Taken from [12].

to 8 T. Such a strong field is necessary to hold the beam of charged particles on their circular path.

Before protons are injected into the 26.7 km long LHC accelerator itself, they are pre-accelerated by a set of smaller accelerators. By using this technique, the protons are injected into the LHC with an energy of 450 GeV and are further accelerated using radio frequency cavities to their final energy. Furthermore, the LHC accelerator complex has been designed to also allow the acceleration of heavy ions, in particular lead ions. In place of protons, they can also to be injected, accelerated and collided inside the LHC machine.

The LHC machine provides beam collisions at four interaction points, where these experiments are located:

- **Compact Muon Solenoid (CMS)** This machine was built as a general-purpose detector for a wide variety of physics studies, among others the search for possible Higgs particles.

- **A Large Ion Collider Experiment(ALICE)** This detector was specifically desgined to record the collisions during the heavy ion operation mode of the LHC.

- **A Toroidal LHC ApparatuS (ATLAS)** Similar to CMS, ATLAS was desgined as a general-purpose machine with a broad physics programme.

- **Large Hadron Collider beauty (LHCb)** This machine is optimized to cover the forward region of particle collisions and focuses of the measurement of b-mesons and their decays.

The particle beams in the LHC machine are not continuous, but subdivided in so-called bunches. The minimum time between the collision of these bunches inside the four experiments is 25 ns in the current LHC design.

3.1. Scientific Goals

The scientific program of the LHC and its experiments is very ambitious and spans a wide range of open and current questions in particle physics and cosmology. In the following, some of the major fields of interest will be outlined.

3.1.1. Higgs Boson

The standard model Higgs boson was postulated in 1964 by independent groups of scientists [16] [17] as part of the Higgs mechanism which is responsible for giving mass to other elementary particles.

The four collaborations at the LEP experiments combined their measurement data and were able to establish a lower bound of $114.4\,\mathrm{GeV/c^2}$, at the 95% confidence level, on the mass of the Standard Model Higgs boson [18].

One important research goal of the LHC and its experiments is the search for the Higgs boson in a wide mass range and, if found, the examination of its properties.

In July 2012, both the CMS and ATLAS experiments announced the discovery of a new particle which is compatible with the standard model Higgs boson [19][20]. The CMS experiment reported a mass of $125.3 \pm 0.4\,(stat) \pm 0.5\,(syst)\,\mathrm{GeV}$ and the ATLAS experiment published their results with a mass of $126.0 \pm 0.4\,(stat) \pm 0.4\,(syst)\,\mathrm{GeV}$.

In the upcoming LHC run period, this new particle will be further studied and its parameters like spin will be measured in detail.

3.1.2. Quark-Gluon-Plasma

The Quark-Gluon-Plasma (QCP) is a proposed state of matter, in which the confinement of quarks and gluons is non-existent, due to the high temperature. Running the LHC in the lead ion mode, allows to create high enough temperatures to potentially recreate the QCP state and therefore study its properties directly.

3.1.3. Precision Measurement of the Standard Model

The standard model of particle physics is incredibly successful in describing the processes in the energy ranges which have been probed so far.

With the new energy range provided by the LHC machine, the predictions made by the standard model can be tested in this areas and the free parameters of the model can be adapted to this new energy range.

3.1.4. CP Violation

The LHCb experiment has been specifically designed to study the CP violation to help understand the matter-antimatter asymmetry we observe in the universe today. Six key measurements of B-decays have been selected to be performed at the LHCb experiment [21].

3.1.5. Physics Beyond the Standard Model

Observed phenomena, like the dark matter and dark energy content of the universe, cannot be easily explained with the elementary particles contained in the standard model.

One theoretical class of models are the supersymmetric extensions, in which each elementary particle from the standard model gets assigned a superpartner. The LHC experiments look for signatures of these SUSY particles, mostly by analysing the missing transverse energy ($\vec{\not{E}}_T$) of events.

3.2. LHC Machine Parameters

As outlined in the previous theory chapter, the luminosity is an important property of a particle collider. The number of observed collisions is

$$N_{events} = L \cdot \sigma_{event} \tag{3.1}$$

here L is the luminosity of the collider and σ_{event} is the cross section of a specific physics process. Under the assumption of a Gaussian beam distribution, the luminosity L of the LHC can be calculated using:

$$L = \frac{N_b^2 \, k \, f_{rev} \, \gamma_r}{4\pi\epsilon_n\beta^\star} \, F \tag{3.2}$$

wherein N_b is the number of particles per bunch, k is the number of overall bunches in the machine, f_{rev} is the revolution frequency of the beam, γ_r is the relativistic gamma factor, ϵ_n the normalized transverse beam emittance and $\beta^\star$ the beta function at the collision point. F is a geometric luminosity reduction factor necessary due to deviation from π of the crossing angle at the interaction point.

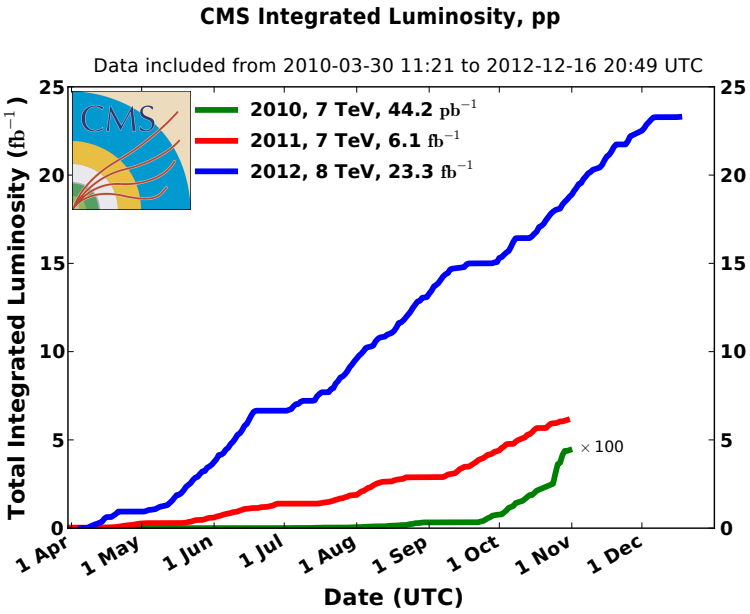


Figure 3.2.: The integrated luminosity delivered to CMS by the LHC machine in the years 2010, 2011, 2012. The graphs shown here are for proton-proton collisions only. Taken from [22].

The specified luminosity for proton-proton operation mode for the ATLAS and CMS experiments is $L = 10^{34} cm^{-2} s^{-1}$ [23]. As outlined in the theory chapter, such a large luminosity is required to also be able to study rare processes which have a very small cross section. At the design luminosity of the LHC machine, around 20 inelastic events occur at the same time as the physics process of interest. This large background poses a significant challenge to both the detector hardware and software systems.

Table 3.1.: Machine parameters for the LHC [24][25] and the proposed HL-LHC machine [26].

Parameter	LHC Run 2012 (at CMS)	HL-LHC (est.)
Peak L [cm$^{-2}s^{-1}$]	$7.9 \cdot 10^{33}$	$2.2 \cdot 10^{35}$
Integrated L [fb^{-1}]	23.3	3000
$\sqrt{s}$ [TeV]	8	14
number of bunches [1]	up to 1400	2808
bunch spacing [ns]	50	25
β^*[m]	0.6	0.15

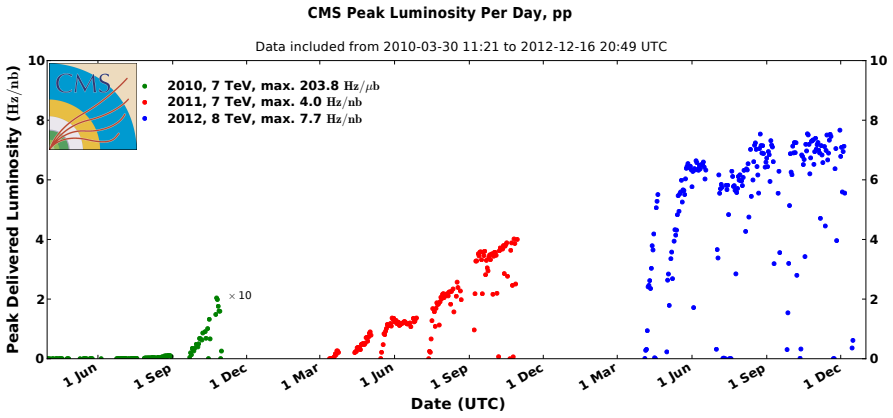


Figure 3.3.: The peak luminosity per day delivered to CMS by the LHC machine in the years 2010, 2011, 2012. The graphs shown here are for proton-proton collisions only. Taken from [22].

3.3. Possible Scenarios for an LHC Upgrade

While the LHC machine is still in operation, studies are already under way for possible machines to follow up on the discoveries made by the LHC. One of the most prominent proposals is the High Luminosity LHC (HL-LHC) [26]. At the core of this proposal is an upgrade to the current LHC machine over the next years leading to an HL-LHC by the year 2020. The design proposes to increase the collision data that can be recorded per year to $250 fb^{-1}$ which leads to an accumulated amount of $3000 fb^{-1}$ for an estimated 12 years of operation [26]. In its current design state, the proposed machine will be able to deliver $2.2 * 10^{34} \mathrm{cm}^{-2} s^{-1}$ of peak luminosity which is 10 times more then the current LHC is capable of. The plan is to achieve this increase in luminosity by using a strong focus scheme to reach very low values of the β function at the collision points [26].

With this increased luminosity, the HI-LHC will be able to increase the measurement accuracy on the properties of particles discovered at the LHC. It will furthermore allow to observe very rare processes which can not be accessed by the LHC, due to its smaller luminosity.

Chapter 4

CMS Experiment

More than 3000 scientist and engineers from 38 countries form the CMS Collaboration and have designed, built and are operating the CMS detector at the LHC accelerator complex. During the design phase of the CMS detector, a set of requirements were defined to guide the development process [23]:

- **Muon Reconstruction** Good muon identification and a di-muon mass resolution of $\approx 1\%$ at a muon energy of 100 GeV.
- **Charged-particle momentum resolution** Design of a tracking system which is able to distinguish particle trajectories close to the interaction point.
- **Electromagnetic Resolution** An electro-magnetic calorimeter which is able to achieve a di-photon and di-electron mass resolution of $\approx 1\%$ at an particle energy of 100 GeV.
- **Missing Transverse Energy Measurement** The $\not{E}_T$ measurement is an important experimental method to spot possible new physics processes. Therefore, the detector should be as hermetic as possible to enable $\not{E}_T$ measurements.

4.1. Coordinate System

The Cartesian coordinate system definition used within the CMS Collaboration places the origin at the beam interaction point, which is at the very center of the CMS detector. Furthermore, the x-axis is defined to point at the center of the LHC accelerator ring and the y-axis points upwards. Finally, the z-axis is oriented along the beam pipe in the direction of the Jura mountain range.

To take advantage of the symmetrical nature around z-axis, also a polar coordinate system can be defined. The azimuthal angle Φ starts from the x-axis in the

x-y-plane. Furthermore, the polar angle θ is measured starting from the z-axis. Figure 4.1 displays a graphical representation of these two coordinate systems and their location and orientation within the CMS detector.

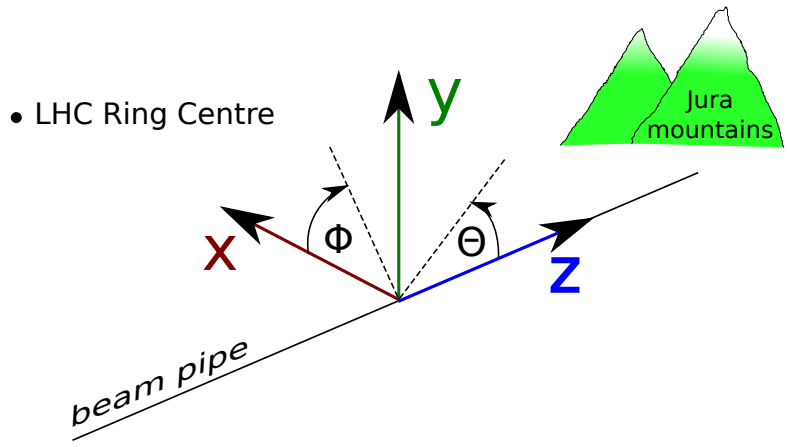


Figure 4.1.: The coordinate system defined by the CMS experiment.

Furthermore, the pseudorapidity is defined as [27]:

$$\eta = -ln\, tan\,(\theta/2) \tag{4.1}$$

4.2. Detector Structure and Magnet System

Figure 4.2 displays the detector components, which are symmetrically arranged around the beam pipe with the interaction point in the very centre. The components parallel to the beam pipe are called the barrel region, the ones closing the detector at both ends are called the endcaps. The complete detector is 21.6 m long, has a diameter of 14.6 m at a total weight of 12,500 t [29].

The superconducting solenoid magnet is a central element of the detector design. It is capable of reaching a magnetic field of 4.0 T in its contained volume. To achieve such a strong field, the alloy NbTi is employed and 2.6 GJ of energy are stored at full current [29]. The critical temperature for NbTi is $T_c = 9.25$ K [29] at zero magnetic field. During regular measurement mode, the magnet is cooled to 4.5 K in order to operate the NbTi material in its superconducting regime.

Contained within the magnetic solenoid are the tracking system, the electromagnetic as well as the hadronic calorimeter. The outermost layer of the CMS detector is comprised of an iron yoke with a mass of 10,000 t, which focuses the magnetic flux of the solenoid into the center of the detector. The muon detection system is interlaced with the the iron slabs of the return yoke and therefor located in the outermost region of the detector.

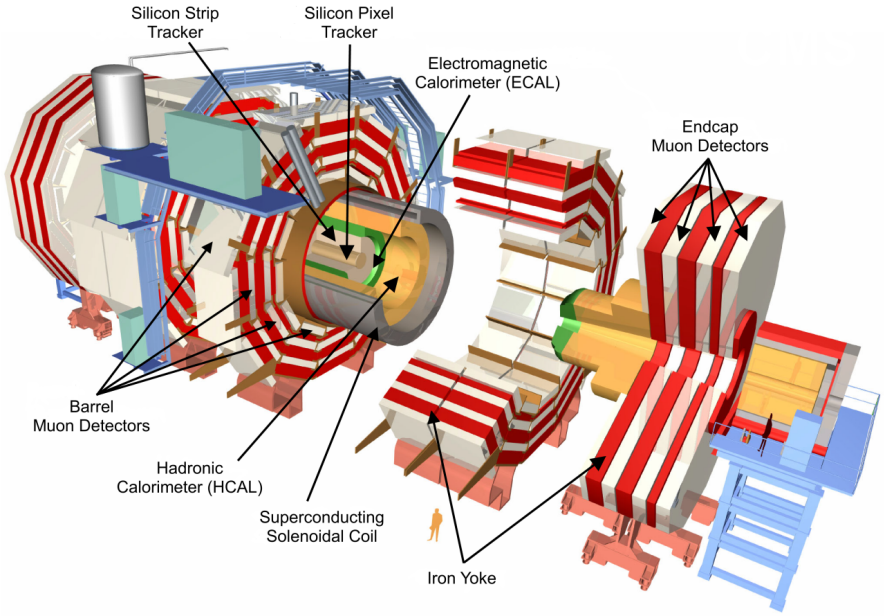


Figure 4.2.: Schematic representation of the components comprising the CMS Detector. Parts of the detector have been cut-away to expose the inner-most components in this artistic rendering [28].

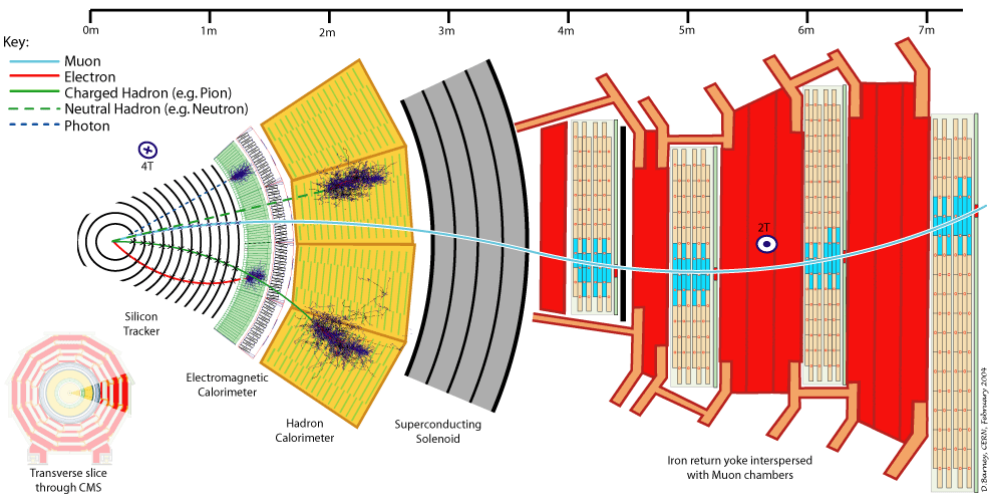


Figure 4.3.: Cross section view of the CMS detector. Various types of particles traverse the detector components starting from the interaction point on the left [30].

4.3. Inner Tracking System

The tracking system, located at the inner-most part of the CMS detector, is essential to detect and reconstruct trajectories of charged particles close to the interaction point. The entire tracking system is based on silicon technology and has an overall length of 5.8 m and a diameter of 2.5 m.

Closest to the interaction point are three layers of silicon pixel detectors in the barrel and two layers in the endcap region. The high position resolution provided by pixel detectors is required to be able to separate the many particles originating at the interaction point. In total, the pixel detector elements cover an area of about $1\,m^2$ and have 66 million pixels combined.

Following the pixel elements, are 10 layer of silicon strip detector elements in the barrel and 11 layers in the endcap region. Using this approach, the tracking system provides a full coverage in ϕ and offers a tracker acceptance of up to $|\eta| < 2.5$.

Chapter 5 will go into more detail on the tracking system's design, material budget and operational performance.

4.4. Electromagnetic Calorimeter

The detector component directly following the tracking system is the electromagnetic calorimeter (ECAL) which is made up of lead tungstate ($PbWO_4$) crystals. In total, 61 200 crystals are integrated into the barrel region and 7 324 crystals are located in each of the two endcap regions [29]. One advantage of the $PbWO_4$-crystals is that about 80% of all scintillation photons are emitted within the 25 ns bunch crossing time of the LHC machine [29].

The ECAL systems covers the whole ϕ range. Furthermore, the barrel section of the ECAL (named EB) extends over the pseudorapidity range of $|\eta| < 1.479$ while the endcap ECAL (named EE) extends from $1.479 < |\eta| < 3.0$.

Two different types of read-out systems are used for the barrel and endcap regions of the ECAL due to the different radiation profiles and magnetic field setups. The barrel region ECAL uses avalanche photodiodes while the endcap region uses vacuum phototriodes.

4.5. Hadronic Calorimeter

As displayed in figure 4.4, the hadronic calorimeter of the CMS experiment is constituted by four individual elements. The barrel HCAL (named HB) is located in the central region of the CMS detector and covers an region of up to $|\eta| < 1.3$ while the two endcap HCALs (named HE) are located at both ends of the detector and cover the range of $1.3 < |\eta| < 3.0$ [29]. Both the HB and HE parts of

the hadronic calorimeter are fully located within the 4 T strong magnetic field generated by the solenoid magnet.

Located outside of the magnetic solenoid, the outer hadronic calorimiter (named HO) is installed in the central region of the CMS detector and covers the region of $|\eta| < 1.3$. Its purpose is to measure particles which have not been fully stopped within the electromagnetic EB and hadronic calorimeter HB in the barrel region.

The hadronic calorimeter setup is completed by the forward HCAL (named HF). It is located in the forward region of the detector, 11.2 m away from the interaction point and extends the calorimeter coverage to $|\eta| = 5.2$.

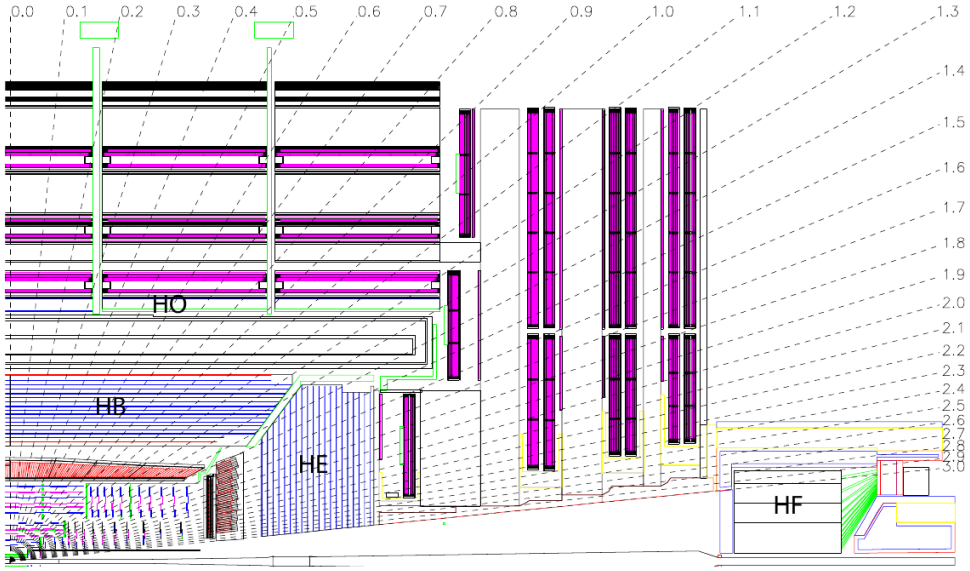


Figure 4.4.: The location of the four elements constituting the hadronic calorimeter of the CMS detector. The muon system is highlited in purple [23].

The design of the hadronic calorimeter in the barrel region will be showcased in the following. It is constituted of brass plates with a thickness of 50.5 mm which are interlaced with plastic scintillator material with a thickness of 3.7 mm. Photons, which are emitted in the scintillating material, are sampled with wavelength shifting fibers and are guided to hybrid photodiodes where a corresponding electronic signal is generated.

4.6. Muon System

This sub-system of the CMS detector provides high-precision measurements of muon tracks. In order to facilitate this, the muon system must identify particles as muons and perform a precise measurement of their charge and momentum.

The muon system of CMS covers an overall area of 25 000 m^2 [29] and is therefor the detector element with the largest area within the CMS detector. It is installed in between the iron slabs of the magnetic return yoke and uses drift tube technology in the central barrel region to cover the area of $|\eta| < 1.2$. A gas mixture constituted of 85% Ar and 15% CO_2 is contained within the tubes and in total 172 000 sensing wires are installed to detect the ionization of the gas mixture by passing high-energetic particles.

In contrast, the two endcap regions use cathode strip chambers (CSC) technology to detect passing particles. This part of the muon detection system covers the region of $0.9 < |\eta| < 2.4$.

4.7. Trigger System

The LHC machine achieves a beam crossing every 25ns, which results in a crossing frequency of 40 MHz [29]. At this high rate, it becomes technically impossible to store all collision information with a justifiable amount of resources. Therefore, a data reduction scheme needs to be adopted which filters undesired events and reduces the storage size of interesting events. These events are stored to disk and can be processed and analyzed in more detail later.

Level-1 Trigger

The CMS collaboration opted to install multiple levels of triggers to achieve a data reduction along the various stages of the trigger. The first stage is the Level-1 trigger system which is located close to the actual detector. This logic is implemented on custom electronic boards which link directly to trigger systems in the tracker, calorimeter and muon system hardware. These systems can flag a measured event, for example if a certain threshold energy was exceeded or a muon was identified. The trigger decisions of all sub-detectors are reported back to the central trigger system which then discards the event or passes it on to the High Level Trigger (HLT) for further evaluation.

Using this technique, the Level-1 trigger can achieve a reduction of the event rate from 40 MHz to 100 kHz. In this process, around 99.75 % of all measured events are excluded from further processing [23].

High Level Trigger System

All events which have been accepted by the Level-1 trigger system are forwarded to the High Level Trigger (HLT). For the first time, all data gathered by the various sub-detectors is combined to achieve a complete picture of the event. This allows to reconstruct the trajectory, momentum and charge of particles traversing the detector. Fine-grained trigger decisions can now be performed using the

information available to the HLT. A comprehensive list of trigger criteria, named Trigger Paths [31], is compiled by the teams of physicists interested in a specific event topology. For example, a trigger criteria can be defined which stores the event only if at least two muons have been reconstructed with sufficient quality. Another Trigger Path can be configured to store events which contain a tau decay candidate.

Furthermore, a pre-scale value can be assigned to each Trigger Path. Using these technique, only a fraction (defined by the pre-scale value) of all triggered events is actually stored on disk, while the rest is discarded. This allows to reduce the necessary data rate of Trigger Paths which have a high acceptance rate and still retain the possibility to study these types on events later.

The HLT-logic is implemented as a computing farm comprised of 10.000 commodity server x86 CPUs. The necessary event reconstruction is a software application running in a highly-distributed fashion on these machines. Therein, the same or very similar algorithms are used as in the offline event reconstruction, which is described in more detail in the chapters 6 and 7. This allows for a flexible setup of the HLT and improvements in the event reconstruction algorithms also benefit the online event selection.

During the 2012 run, the HLT was configured to achieve an output frequency of 300 Hz while the input stream of events was provided by the level-1 trigger at a rate of 100 kHz. All events accepted by the HLT system are transferred from the CMS Detector's location at Point 5 to CERN's central data center and stored for later offline reconstruction.

Chapter 5

CMS Tracker

Robust and reliable track and vertex reconstruction play an important role in the high-luminosity environment provided by the LHC machine. Achieving a good particle identification and measurement resolution with more than 20 pile up collisions overlaying the actual primary one is a challenge to be addressed by the tracker hardware design and reconstruction software.

This chapter will outline the hardware design of the tracking system while chapter 7 will detail the track reconstruction algorithms.

5.1. Design Goals of the Tracking System

5.1.1. Momentum Resolution

The decay of the W and Z gauge boson plays an important role in the processes observed at the LHC machine. Especially their decays into leptons provide clean signals for analysis [32]. Therefore, the tracker must provide a good momentum resolution in a wide momentum range to infer the energy carried by the these particles.

5.1.2. Isolation of Particles

The detection of isolated leptons in the tracker, in particular electrons, plays an important role in assigning energy clusters measured in the electromagnetic calorimeter to decay vertices. This allows to suppress background and helps to measure decays like $H \rightarrow ZZ \rightarrow 4l^{\pm}$ in a cleaner way.[32]

To achieve this, an effective reconstruction of all tracks down to 1 GeV is necessary.[32]

5.1.3. B-tagging Abilities

The ability to reconstruct and identify jets resulting from a beauty quark decay is very important, as they are considered important signals for new physics, can help to give more insight into the CP violation and are an essential tag for top quark physics [32].

To be able to achieve this, the decay vertex must be identified properly and low p_{T} tracks must be correctly reconstructed and assigned.

5.1.4. Vertexing and Decay Chains

With around 20 pile-up interactions overlaying the primary collision, a reliable and efficient vertex reconstruction must be deployed. It is responsible for detecting collision points (vertices) and assigning reconstructed tracks to them. As the vertices are located inside the high-vacuum beam line, they can not be measured directly, but can only be inferred from the reconstructed tracks.

Therefore, the track resolution close to the collision area must be high enough to allow for a reliable vertex reconstruction and track assignment.

Furthermore, particle decay in flight within the tracking detector must be detected, resulting tracks reconstructed and properly assigned to the decay location. These locations are named secondary vertices.

Secondary vertex reconstruction is essential for observing the decay of the short-lived Kaon K_s^0 which will be described in chapter 7.7.

5.1.5. High Particle Multiplicities

The LHC offers the unique possibility to also produce high-energetic heavy ion collisions and can potentially produce a Quark-Gluon-Plasma. These events are also recorded by CMS and have a very high multiplicity of up to 25,000 particles [32] and the tracking system has to be able to cope and operate in such a high-occupancy environment.

5.2. Measurement Principle

The physical principal behind Silicion-based tracking systems is that of semi-conductivity. In terms of their electrical conductivity, different type of matter can be separated into the three classes conductors, semi-conductors and isolators. The property which allows this separation is the so-called bandgap E_g between the conduction band and the valence band. The band-gap can be computed as follows:

$$E_g = E_c - E_v$$

E_v is the highest possible electron energy state in the valence band and E_c the lowest possible electron energy state in the conduction band.

While the valence band electrons are bound to their corresponding atoms, in the conduction band they are loose and can move quasi-free in the material. Metals are conductors because they have no band-gap and only very little energy is required to allow for electrons to move freely in the lattice structure of the metal. On the other hand, with insulators, the band gap is significant and the energy necessary to excite electrons to move from valence band to the conduction band is high.

Semi-conductors are localised between these two poles and are defined as having a significant band-gap energy, but still smaller then $E_g < 5eV$.For example, the bandgap of Silicon at room temperature is $E_g = 1.11eV$ [33].

This results in semi-conductors being isolators if their electrons are all in the lowest possible energy state. Only very little is energy is necessary to move electrons into the conduction band and make them act as quasi-metals. This extraordinary property of semi-conductors allows to use them in a wide range of applications from transistors in electronics, digital cameras to HEP tracking systems.

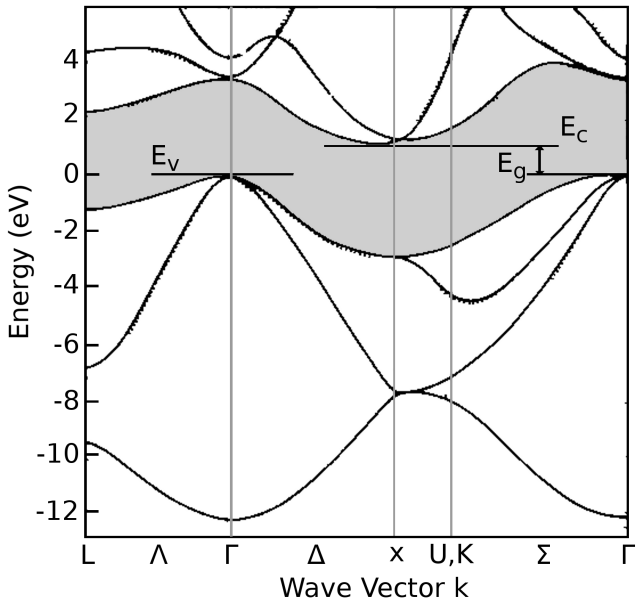


Figure 5.1.: Band structure of Silicon. The band gap region is marked with a grey background. Own work based on [34].

The transition between the valence and conduction bands can be separated into two types. In a direct band gap, only energy needs to be transferred in order to bridge the gap. The momentum between the two states is equal and needs not to be changed. In an indirect band gap, the crystal momentum, defined by the wave

vector k, between the highest energy state E_v in the valence band and the lowest energy state E_c in the conduction band is different and therefore an additional momentum transfer is necessary to perform the transition. For the case of Silicon, having an indirect band gap, a phonon in the lattice structure of the material is carrying the momentum in a transition process. [35].

Figure 5.1 displays the band structure of Silicon and illustrates the indirect transition between the valence and conduction band.

5.2.1. Doping of Semi-Conductive Materials

To change the properties of semi-conductors, foreign atoms can be introduced into their lattice structure. In the case of Silicon, which holds four electrons in its outer, weakly bounded valence band, atoms with three or five electrons in their outer band are introduced. When using elements from the third main group of the periodic table as so-called dopants, the missing negative charge behaves like a positive one and is called *hole* in the structure of the semi-conductor. Materials treated in this manner are called p-type (for positive) semi-conductors.

In the same way, doping Silicon with elements from the fifth main group introduce an additional electron and thus charge carrier. This increases the electron density and materials treated in this manner are called n-type (for negative) semi-conductors.

Using these doping techniques, the size of the bandgap can be modified and one can decide whether a semiconductor will use holes (with p-doped material) or electrons (with n-doped material) as charge carriers.

5.2.2. Structure and Functionality of the CMS Silicon Chips

The Silicon tracking chips of CMS are designed with a bulk layer of high dose n-doped Silicon, followed by a layer of p-doped Silicon thus creating a pn-junction [35]. On the backside of the chips, a highly p-doped layer of Silicon is located. As particles traverse these layers of semi-conductive material, they deposit a part of their energy via the processes described in chapter 2.6.

The deposited energy will lift electrons from the valence into the conduction band and thus create electron-hole pairs, which can move quasi-freely along the lattice structure of the semi-conductor. In this process, the created number of electron-hole pairs n is proportional to the energy $E_{deposit}$ deposited in material divided by the ionization energy necessary to produce electron-hole pairs E_{ion}.

$$n = \frac{E_{deposit}}{E_{ion}}$$

For Silicon, the average energy which is necessary to produce a electron-hole pair is $E_{ion} = 3.6\text{eV}$. This is higher as the band-gap energy as Silicon is an

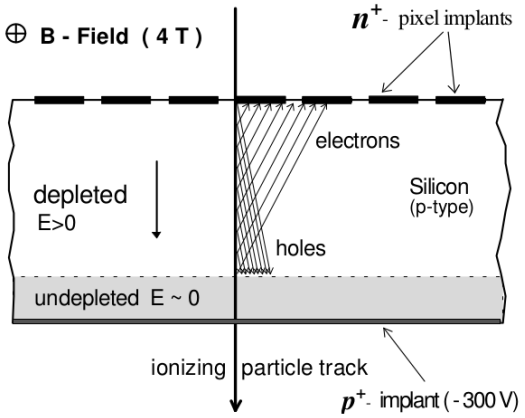


Figure 5.2.: Schematic view of the doped Silicon layer used in a CMS Tracker chip [32].

indirect semi-conductor, as part of the energy is necessary to produce a phonon in the lattice structure of the material [35].

5.2.3. Bias Voltage and Signal Read-Out

Without any external electric field, the electron-hole pairs created by either thermal effects or passing particles vanish in a process of recombination and the separation of positive and negative charges in the semiconductor arrive again at a state of equilibrium [35].

In order to read-out the number of created electron-hole pairs, a so-called *bias voltage* [32] is applied to the pn-junction. This voltage will prevent the electron-hole pairs from recombining and allows to collect and count the electrons on the n-type side of the chip.

Depending on the type of chip, either a one-dimensional array for strip chips or a two-dimensional array for pixel chips collects the electrons. This electronic is called chip read-out and can either provide an analog or digital signal to the DAQ system.

The digital read-out type provides a binary signal which is *false* by default, but is switched to *true* if the collected amount of charge for one read-out channel is larger then a pre-configured threshold.

The CMS read-out electronics provides an analog readout to the DAQ. Therein, the integer number provided is proportional to the collected charge. As a particle passing through the Silicon chip leaves a signal for more then one channel, the analog signal allows for a more detailed determination of the intersection point between the particle trajectory and the detector element surface.

5.2.4. Momentum Measurement

While the direction of a particle can be directly measured using the energy deposits, so-called *hits*, in the Silicon detectors, the momentum must be derived indirectly.

As the whole CMS Tracker is embedded in the 3.8 T strong magnetic field produced by the CMS magnet, all charged particles are affected by the Lorentz force. Assuming there is no electric field, this force can be written as [36]:

$$\vec{F_L} = q\vec{v} \times \vec{B}$$

Here, q is the charge of the particle, $\vec{B}$ is the strength of the magnetic field and m is the mass of the particle.

As the magnetic field of CMS is oriented along the z direction, the particle tracks will describe a helix along the z direction [36]. Only the transverse component p_{T} of the particle trajectory is affected by the magnetic field. By determining the radius R of the helix, the transverse momentum p_{T} can be derived. This is done by the track reconstruction software by using consecutive tracker hits to determine p_{T}. Furthermore, also the overall momentum can be reconstructed by combining the p_{T} component of the particle momentum and the knowledge about the direction of the particle. This procedure is described in more detail in section 7.5.

5.3. Tracker Components

The tracker system is made up by 16588 individual detector elements [37] which consist of a Silicon sensor, read-out electronics and energy supply.

These parts are organized in stacked layers in the barrel and endcap region in order to cover all the required areas.

The tracking system can be separated into two main components. Although these components have a lot in common, like the general measurement principle, and share some of the infrastructure components like energy supply and cooling, they offer different performance in terms of measurement resolution and volume they cover.

The innermost three barrel detection layers and the first two layers in the endcap region consist of 1440 Silicon-based detection elements and are able to provide a two-dimensional measurement of the traversing particle by using a 2d-grid of pixel cells. All of these measurement layers are referred to as the *Pixel Tracker*.

The remaining layers in the barrel endcap region are populated by 15148 elements of Silicon-based strip detectors. These elements can provide a one dimensional hit position perpendicular to their strip direction.

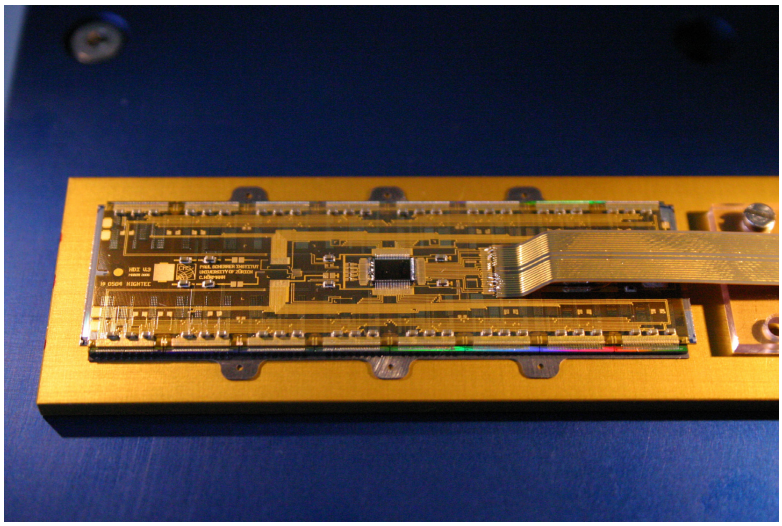


Figure 5.3.: Photo of a pixel module [38].

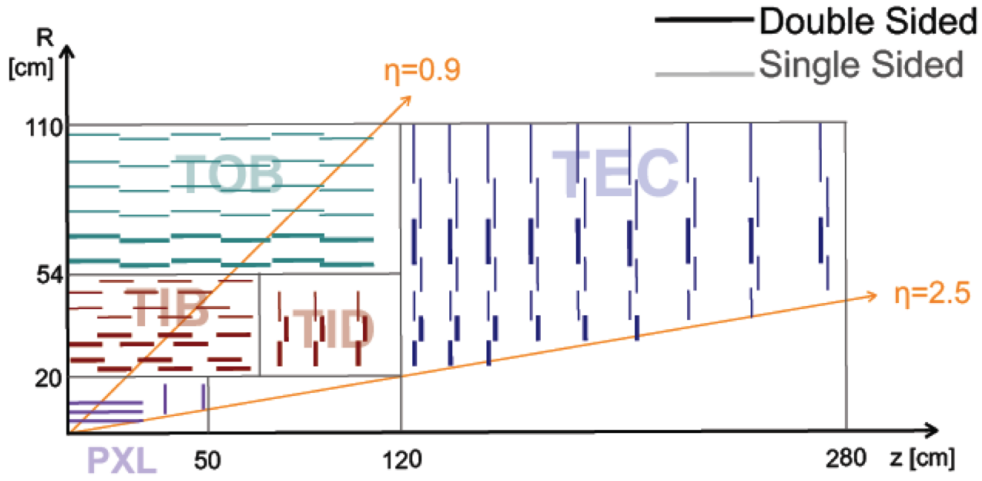


Figure 5.4.: R-Z view of one quarter of the CMS Tracker showing pixel and strip silicon layers. Double sided strip modules are highlighted with thick lines [39].

In the R-Z view of the Tracker in Figure 5.4, one can see how the pixel and strip elements have been arranged in order to cover the range of up to $|\eta| \leq 2.5$. Furthermore, the detector elements cover the full ϕ space.

5.3.1. Inner Pixel Tracker

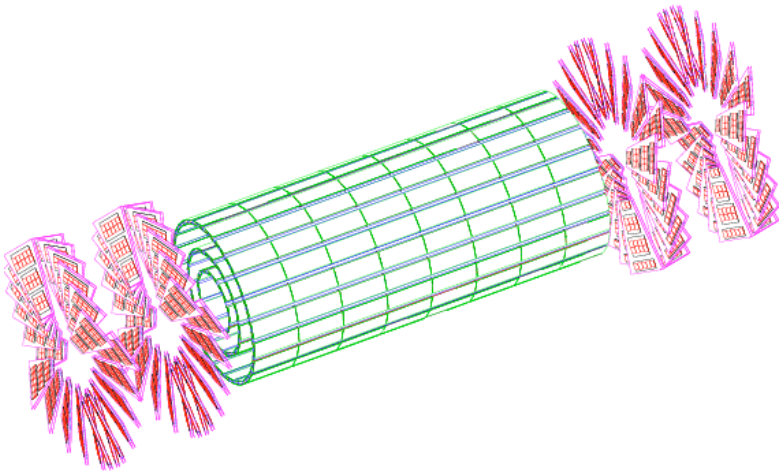


Figure 5.5.: Schematic view of the pixel-based inner tracking system. The central three layers of barrel modules (green) are visible and the two endcap pixel layers (red), which exhibit a fan-like structure, are visible on both ends [38].

The three layers of the barrel pixel system are placed at a mean distance of 4.4, 7.3 and 10.2 cm from the primary interaction vertex [38] and are therefore the closest detector elements to the collision point. The pixel endcap elements are arranged in two layers in a fan-like structure on both ends of the barrel pixel layers. They are 6 to 15 cm in radius, and are located with a distance of $\pm 34.5\,cm$ and $\pm 46.5\,cm$ from the primary interaction point. Figure 5.5 shows a schematic view of the pixel tracker and highlights the barrel part in green and the endcap part in red.

As these pixel modules are the closest detection modules to the interaction point, they have to cope with a high particle rate and still need to be able to provide sufficient resolution to perform vertex recontstruction and b-tagging in a reliable way.

For this reason, these silicon detectors have the highest resolution of the tracking system with an individual pixel size of $100 \cdot 150\,\mu m^2$. However, this resolution can be greatly improved by measuring the charge deposit as an analog value. By combining multiple charge distribution across all pixels a particle passed, the

resolution can be greatly improved to $\approx 10\,\mu m$, . The algorithm used here will be described in detail in chapter 7.

5.3.2. Outer Strip Tracker

The strip part of the tracking system can be further separated into the TIB (Tracker Inner Barrel) and TOB (Track Outer Barrel) components for the barrel region and into the TID (Tracker Inner Disc) and the TEC (Tracker End Cap) components for the endcap region. This naming scheme is best illustrated by figure 5.4.

The TIB is made up by four layers with a minimum strip size of $10cm \cdot 80\mu m$ and the first two layers are stereo layers, with two strip elements very close to each other. Having stereo layers allows to combine the two individual strip measurements of a single passing particle and derive a two dimensional hit. This enables a better measurement of the $r - \phi$ and z components of the track. Furthermore, by having these precise hit informations, stereo layers can also be used to search for secondary vertices and conversion finding. This will be explained in more detail in chapter 7 which is dedicated to track reconstruction .

Due to the significant drop in occupancy in the outer layers of the tracker, the seven layers of the TOB were designed with a maximum strip size of $25cm \cdot 180\mu m$. Again, the first two layers of the TOB are equipped with stereo modules.

The TID is comprised of three radial discs which fill the gap between the TIB and the TEC. The lower two rings of the TID sensors are stereo modules, too.

The largest part of the tracker endcap region is filled by the nine radial discs of the TEC. As visible in figure 5.4, selected rings of the TEC discs are equipped with stereo modules, too.

5.4. Material Budget

The so-called *material budget* is an important quantity in every detector design and refers to the amount and type of matter which incoming particles have to cross while inside the detector volume. Sensitive measurement elements in a detector are denoted *active material* and are, in the case of a Silicon-based tracking system the Silicon sensors. They can be read out to determine the charge deposited by a passing particle. Support structures, cooling, cabling and even air and other gas mixtures are denoted as *passive material*, as energy deposited in these materials can not be measured.

Whether a significant amount of active and passive material is favourable depends on the detector design and the particles which should be measured. For example, the design of the CMS hadronic calorimeter, described in section 4.5, utilizes a big amount of brass plates as passive material. They cause interactions

Component Category	Mass [kg]	Fraction [%]
Carbon Fiber	1144.5	27.6
Copper and copper alloys	644.5	15.6
Aluminium	595.0	14.4
Organic materials	472.1	11.4
Coolant (C 6 F 14)	258.9	6.3
Silicon active	225.8	5.5
Fiber-glass laminated	213.4	5.2
Other mechanical structures	191.7	4.6
Inorganic oxides	153.2	3.7
Other metals	141.9	3.4
Glues and resins	75.3	1.8
Electronic component	25.7	0.6

Table 5.1.: Mass distribution across various types of material extracted from the GEANT4 detector model of the CMS Tracker. [37].

with the incoming particles to enable the measurement of resulting secondary particles in the active material behind each brass plate.

As described in section 5.2, the CMS tracker was build to measure the energy and direction of passing particles via their curvature in the magnetic field. At the same time, particles like electrons, charged and uncharged hadrons should pass the tracker without much energy loss, so their energy can be determined in the electromagnetic and hadronic calorimeters.

To facilitate this, the CMS tracking system was built with very light materials like carbon fibre composite as support structures. Table 5.1 lists the mass distribution derived from the tracker geometry as modelled in the GEANT4 software package. The estimated total mass of the Tracker amounts to about 4150 kg [37] distributed within a total tracker volume of about $23.5\,m^3$.

One can see from table 5.1, that the active Silicon material amounts to around 5% of the overall mass of the tracking system.

Furthermore, the mass of active and passive material is concentrated at the detector layers, their support structures, cooling and wiring. Figure 5.6 shows the mass distribution in radiation length for the each tracker subcomponent and for the pixel tracker only. Both plots have been generated from the tracker geometry modelled in the GEANT4 software package. The active measurement area of the tracking system is limited to $|\eta| \leq 2.5$ and material plotted above $|\eta| > 2.5$ will not affect the track measurements, but detectors further away from the interaction point (for example the forward hadronic calorimeter (HF)).

The left plot illustrates that the lowest radiation length x/X_0 in the design of the tracker is in the central part $\eta \approx 0$ while the largest are located at around

$|\eta| \approx 1,5$ in the so-called *transition region*, where the barrel and endcap parts of the tracker overlap. For $|\eta| > 1.5$, the radiation length x/X_0 decreases again, but is still significantly larger at $|\eta| = 2.5$ as in the central barrel region.

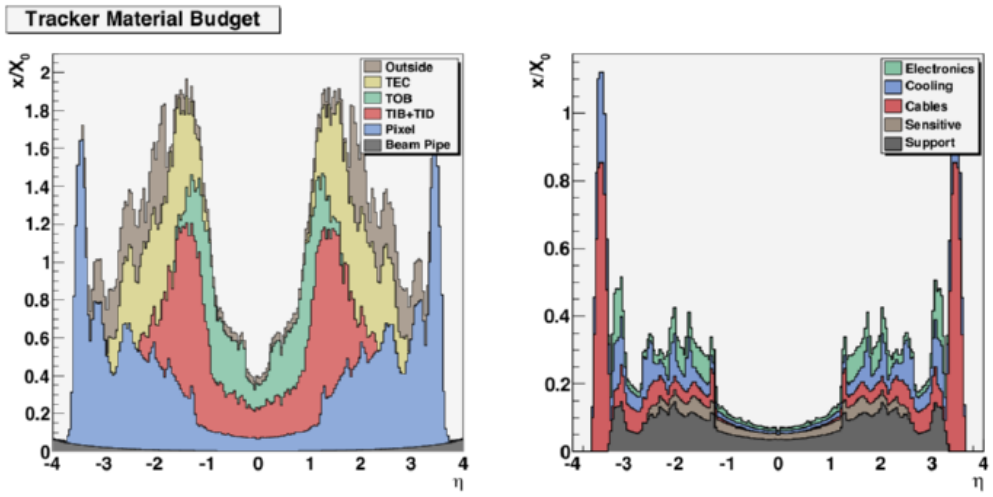


Figure 5.6.: The total integrated material budget in terms of radiation length x/X_0 of the full tracker (left) and the pixel tracker only (right). The various contributions of the sub-components are stacked. These plots have been generated from the tracker geometry modeled in the GEANT4 software package [40].

Although the tracker model in the GEANT4 package has been created with great care and accuracy (more than 350.000 volumes [37]), it can only represent the material amount and distribution of the real-world tracker to a certain extend. Slight variations in weight and composition of the used materials, production tolerances and limitations in the accuracy of the model can result in differences.

To quantify possible differences, multiple methods can be employed. Photon conversions, wherein one photon converts to an e^+e^- pair ($\gamma \rightarrow e^+e^-$) and nuclear interactions, wherein charged pions interact with the tracker material, are especially useful to probe the tracker material in-situ. With these techniques, one uses the fact, that the rate of photon conversions and nuclear interactions is proportional to the radiation length resp. the nuclear interaction length in the crossed material [40].

Figure 5.7 shows the material distribution derived from nuclear interactions over the radial value of the reconstructed vertex. The lower section of the plot shows the inverse nuclear interaction length directly derived from the Material in the GEANT4 simulation. The upper part of the plot shows the material distribution derived from Monte Carlo and data using the nuclear interaction method.

The same kind of plot can be seen for photon conversions in figure 5.8.

Using this technique, the agreement between the GEANT4 model of the tracker and the actual amount and distribution of material in the real-world tracker was estimated to be in the range of $\approx 10\%$ [40].

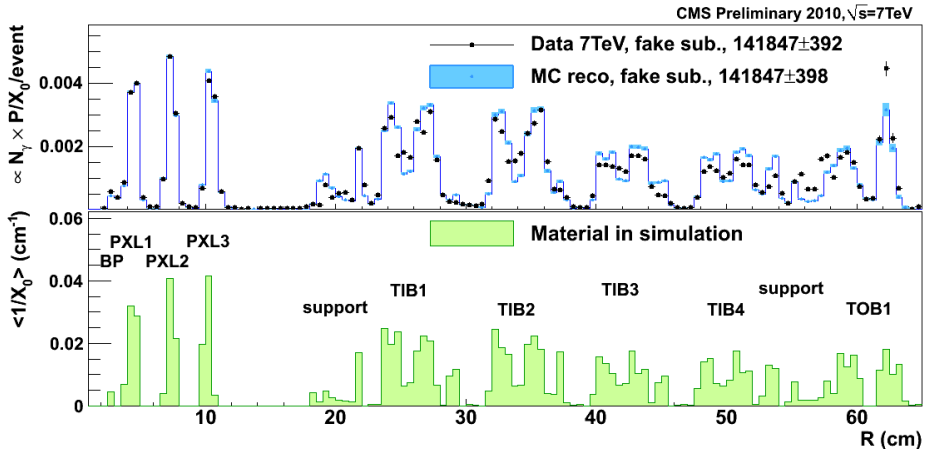


Figure 5.7.: Material distribution derived from nuclear interactions plotted over the radial value of the reconstructed conversion vertex. The lower section of the plot shows the inverse nuclear interaction length directly derived from the Material in the GEANT4 simulation. The upper part of the plot shows the material distribution derived from Monte Carlo and data using the nuclear interaction method. Taken from [41].

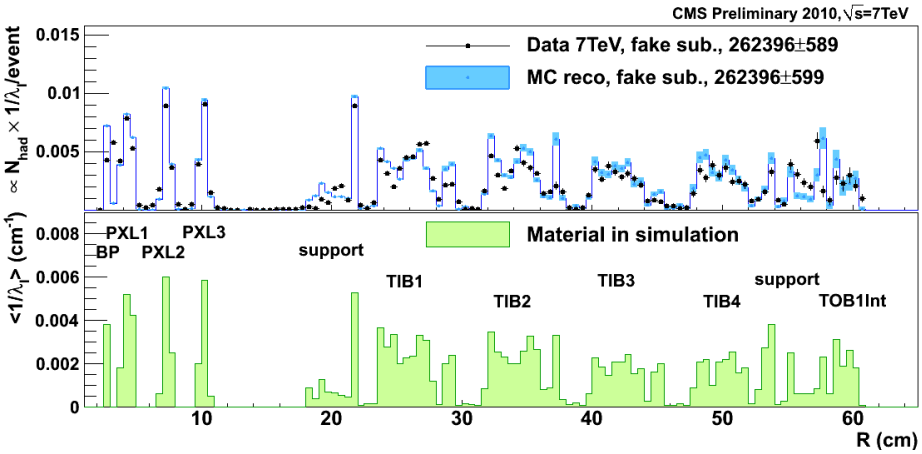


Figure 5.8.: Material distribution derived from photon conversions plotted over the radial value of the reconstructed conversion vertex. The lower section of the plot shows the inverse radiation length directly derived from the Material in the GEANT4 simulation. The upper part of the plot shows the material distribution derived from Monte Carlo and data using the photon conversion method. Taken from [41].

Chapter 6

CMS Software and Computing

To further process and analyze the data delivered by the CMS Detector, a whole range of software processing steps are necessary. This includes the unpacking of the raw data stream from the detector's read-out electronics, applying calibration constants, reconstructing particle trajectories up to selecting events with a specific topology. The CMS Software Framework, short CMSSW [42], is a C++ application which is used to implement all these functionalities in a modular manner. It allows to configure the individual modules and define the data and control-flow among modules. In the following, the main building blocks of the CMS Software Framework will be outlined and examples for specific modules will be given.

6.1. CMS Software Framework

One central concept employed by CMSSW is the Event Data Model (EDM) exchange technique [38]. All raw measurement data and derived products, like reconstructed particle tracks and energy deposits, are stored in a central container. Software modules retrieve their input data from this container and store the result of their processing back to this container. Data items stored in the container can be identified either by their data type or name. This concept decreases the dependence among the modules to a minimum and allows for a flexible configuration of the module execution order.

Modules are arranged in so-called Paths which define the execution order, and therefore the processing hierarchy, of the setup. Furthermore, multiple Paths can be defined and filled with equal modules but different configurations. This allows to process the same set of input data with different configurations, for example calibration constants.

It is possible to implement a CMSSW module in one of the following categories:

- **Source**

  Event data can either be read from files or via network streams. In case of offline processing, previously stored events are loaded from the disk mass storage system. In the HLT setup, event data is streamed via a network connection directly into CMSSW using a special source module.

- **Producer**

  Producer modules read data items from the central data store and add a newly created product, which contains the result of the module's computation, to the data store. The CMS event reconstruction is implemented as Producer modules which are connected by multiples Paths. One of these modules is the track finding step, which loads the tracker hits from the central store, applies the track finding algorithms and outputs a list of found track candidates to the central data store.

- **Filter**

  In contrast to the Producer, no output data object is created by the Filter. These modules can access all data items produced so far and decide whether the execution of a Path is continued or stopped. This allows to quit the processing of a event which does not suffice the criteria of the topology of interest, for example no two muons were measured.

- **Analyzer**

  The data items created by the Producer module can be read by a specific Analyzer module. In contrast to the Producer, Analyzer modules don't save their results back to the data store but store them on the disk in the form of data tables, histograms and graphs.

- **Output**

  Output modules store the current content of the data store to the disk. The saved file can be later used to restore the state of the data store, continue processing or apply further analysis steps.

6.2. ROOT Data Analysis Framework

Many high-energy physics experiments share the same software requirements in terms of data storage, statistical analysis and visualization and plotting needs. The ROOT [43] framework has been developed as a comprehensive software library and application to address these areas. It is maintained by a core development team mostly located at CERN and used by all major high-energy physics

experiments today. The largest part of ROOT's functionalities are implemented using the C++ language, but also python can be used to create programs using classes and functions offered by the ROOT library.

One of ROOT's most prominent features is its ability to store the content of C++ classes to disk and load them again at a later stage. Nearly all HEP experiments use this functionality to save their recorded event data and to store a reduced set of values for analysis purposes, so-called n-tuples. Furthermore, due to ROOT's standardized data format, the files created can also be used to exchange data within one experiment's working groups or even across experiments.

6.3. Event Simulation

To compare the predictions of various theoretical models with the observed events in the CMS Detector, a sophisticated event simulation is employed. Starting with the proton-proton collision in the beam pipe, a chain of consecutive simulation steps is performed. The initial collision products resulting from the parton-parton-interaction can be generated by so-called Monte Carlo event generators. Various generators are available including PYTHIA [44], Herwig [45], Alpgen [46], MadGraph [47] or Sherpa [48] and many of them can be interfaced by the CMSSW framework.

Using Monte-Carlo based methods, an event generator can compute the final state of various physics processes. The decay, propagation and their interaction with the material of the CMS detector of these particles is simulated via a detector simulation step. Energy deposits resulting from these particles are recorded and used as input for the event reconstruction to understand the detector response and measurement resolution for specific event topologies.

Furthermore, the information of the event generator is retained and can be compared to the result of the event reconstruction. This allows to estimate the reconstruction quality and particle finding efficiency of the hardware and software elements of the detector.

6.3.1. Event Generators

Many event generators are in active development and differ in the amount and type of physics processes they have implemented. Among the most popular are PYTHIA [44], Herwig [45] and Sherpa [48] which can be directly used from withing the CMSSW framework.

The PYTHIA event generator has been used to produce events for this thesis. Therefore, this software package will be introduced in more depth in the following. It implements more than 300 leading order calculations of physics processes of the Standard Model, Minimal Supersymmetric extension of the Standard Model and

non-standard physics [44].

A model-based process can be applied to particles subjected to QCD confinement to reach a colourless final state. This process is called fragmentation, as the coloured particles result in a shower of colour neutral particles. By default, PYTHIA uses the Lund model to calculate the final hadronic state of the event. The interaction between partons is modelled by a virtual string which is stretched as the particles move apart. If a critical tension in the string is reached, it rips and new particles are produced at the ends of the two new strings. This process is performed until only non-coloured particles are left. This showers of hadronic particles are not bound by QCD confinement and can freely propagate [11].

Additionally, PYTHIA simulates the decay of the hadronic particles created by the collision, so only quasi-stable particles are left in the final state [11].

PYTHIA is also able to simulate the underlying event, which consists of multiple parton interaction in the beam pipe and is an important background signal for many investigated physics processes. Also the initial and final state radiation is taken into consideration by the PYTHIA simulation run. The implementation of this features are based on phenomenological models and are depending on the centre-of-mass-energy used at the detector. So-called *Tunes* contain a set of parameters which have been optimised to fit the behaviour observed in actual measurements[49] [11].

6.3.2. Detector Simulation

To simulate the interaction of the detector components with passing particles, the geometrical layout and magnetic field of the CMS detector have been modeled in the software package Geant4 [50]. Using this technique, Geant4 is able to track the particles trajectory within the detector volume, apply energy loss effects and, for charged particles, changes in the path introduced by CMS's magnetic field [51].

Furthermore, energy deposits in each of the active detector components like the silicon tracker and the calorimeter cells are counted and used as an input to the reconstruction algorithms. This readout process of the detection elements is called *Digitisation*[51].

6.4. Track Reconstruction

More details about the track reconstruction can be found in the dedicated chapter 7 on this topic.

6.5. LHC Computing Grid

Events which have been selected by the high level trigger are transferred to the primary CERN data center and their reconstruction is started within the next 48 hours [52]. This procedure is called prompt reconstruction.

Apart from the prompt reconstruction step, the CMS Collaboration requires a lot more computing capabilities. Among them are the production of large Monte-Carlo data sets, the re-reconstruction of collisions with updated calibration constants and the final analysis of the recorded and simulated events. Furthermore, storage space must be available to store the recorded and simulated events in a redundant fashion to minimize the risk of data loss.

To offer these services at the required scale, the decision was taken to employ a tiered and distributed computing model for the CMS Collaboration. In this scheme, one Tier-0 center, located at CERN, performs the prompt reconstruction and serves as one of the primary storage facilities for recorded data sets. Tier-1 centers are connected via dedicated glass fiber links and store redundant copies of the recorded data. Furthermore, Tier-1 centers provide computing resources to produce Monte-Carlo events or rerun the reconstruction of recorded events. Tier-2 centers are also participating in Monte-Carlo production campaigns and offer resources for scientists to perform analysis of the recorded and simulated data sets. Tier-3 centers provide additional computing resources, mostly only to the local university users, and are not used for the official storage of data sets.

In the year 2013, the CMS Collaboration employed seven Tier-1 centers and 52 Tier-2 centers [52]. Figure 6.1 illustrates the tiered layout of the CMS computing model.

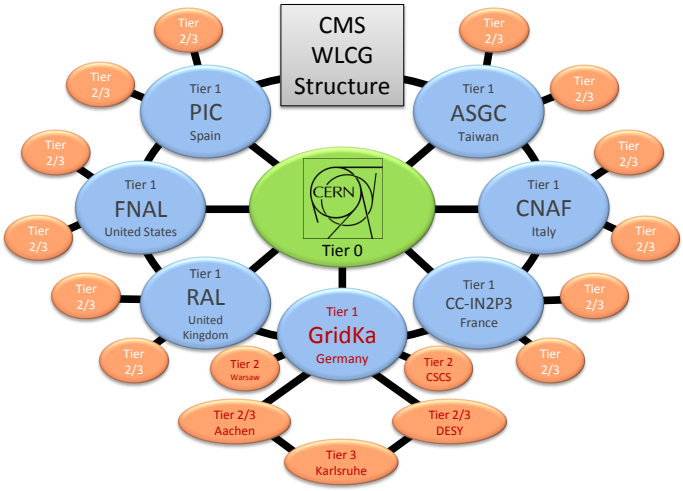


Figure 6.1.: The multi-tiered architecture of the Worldwide LHC Computing Grid [53].

Chapter 7

CMS Track Reconstruction

While the CMS Tracker described in chapter 5 is able to provide measurements of traversing particles, called *hits*, it does not provide information about the direction or momentum of passing particles. To determine this information, a complex procedure named *Track Reconstruction* needs to be performed.

This procedure can be separated in two logical steps, track finding and track fitting. Depending on the actual procedure used, both may also be performed by the same algorithm.

Track finding is the process of identifying possible particle tracks from the three-dimensional point cloud of tracker hits. The decision needs to be made, which hits belong to which track.

Track finding is using the information hit assignment to tracks and performs a fit of the track model to these hits. The result of the fit is the determination of the track direction and momentum at each hit position of the track.

7.1. Reconstruction Quality Criteria

$$E = \frac{N_{\textbf{valid found}}}{N_{\textbf{gen}}} \qquad F = \frac{N_{\textbf{fakes found}}}{N_{\textbf{found}}} \tag{7.1}$$

To quantify the reconstruction performance of the employed algorithms, efficiency E and the fake rate F can be defined. This quantities can be determined with simulated events, where the number of generated particles $N_{\textbf{gen}}$ is known. The efficiency is defined as the ratio between the valid particles found by the reconstruction software $N_{\textbf{valid found}}$ and the number of generated ones $N_{\textbf{gen}}$. An efficiency close to 1.0 is desired.

The fake rate quantifies how many combinations of hits have been misidentified as belonging to one track and therefore reconstructing a track which has not been generated in the first place. The fake rate can be computed by taking the ratio

between the number of fake combinations $N_{\textbf{fakes found}}$ and the number of the overall found tracks $N_{\textbf{found}}$. A fake rate close to zero is desired in order to only have valid tracks in the final selection.

7.2. Requirements on the Track Reconstruction

The requirements on the track reconstruction are many fold. Some requirements are independent of each other, while others are closely linked.

- **Track finding**

  The ideal track reconstruction is able to find all tracks which are present in the event. This requirement is expressed in form of the track finding efficiency, which is one of the main quality criteria in track reconstructions validations.

- **Low fake rate**

  While one intention is to achieve a high efficiency, it is also desirable to have low fake rate. An elevated fake rate can occur, when the algorithm collects non-related hits to tracks and is not able distinguish them from actual tracks. As fake tracks have no correspondence to the measured physical process and are merely an artifact of the reconstruction, they can be a source of measurement uncertainty.

- **Correct assignment**

  The tracking software will assign hits to its track hypothesis to form a final track. It will incorporate all hits assigned and use them for the final track fit. In this respect, only hits which result from one particles traversal in the detector should be assigned to one track. Failing to do so may result in a quality degradation of the final track fit.

  Furthermore, the reconstruction software needs to assign found tracks to the vertices where they originate. Again, the best possible assignment quality is desired here.

- **Best-estimation of the track parameters**

  Once the correct assginment of hits to tracks has been made, the direction and momentum of the track need to be computed. During this fit procedure, the measurement uncertainties on the hits, the magnetic field and tracker material must be modeled correctly.

- **Sufficient runtime performance**

  The track reconstruction using hits from the tracking system poses a big computational challenge due to high combinatorical freedom in combining hits to tracks. A track reconstruction software can therefor only be called successful it is able to fulfill the before mentioned requirements and stays within the runtime and memory consumption budget available. To facilitate this, comprises may need to be taken in terms of algorithmic complexity.

7.3. Kalmann Filter Basics

The Kalman filter technique was initially developed and utilized in the areas of measurement and control systems. Among other applications, it was used to estimate the trajectory of the Apollo spacecraft on its way to the moon [54].

In its linear form, the Kalman filter is the optimal recursive estimator of the state of a linear dynamic system. In such a system, the evolution of the state is described by a linear transformation plus a random disturbance w, which is the process noise [55].

The system equation describes the relation between the current state of the system x_k at step k and the previous state at step x_{k-1}:

$$x_k = F_{k-1}x_{k-1} + w_{k-1}$$

Here, F_{k-1} is a transformation which transforms a state from step $k-1$ to k. w_{k-1} is the so-called process noise which is introduced by the transformation of states.

Trajectories in a homogeneous magnetic field can be described by five parameters [36] which define the helical path of a charged particle and can be used as the state of the system, the so-called *track state*. When applying the Kalman method to track reconstruction, x_k is the track state after k hits have been processed. If n hits are available for one trajectory, the Kalman filter will start at the first hit $k = 1$ and be finished once the state $k = n$ has been reached.

In the case of track reconstruction, the transformation F is performing the track state propagation from one detector surface to the next one. The process noise w is the random disturbance of the track along the propagation path, mostly due to multiple scattering effects in the crossed material [55].

In the track reconstruction procedure with input hits from the CMS tracker, the complete track state cannot be derived from one hit alone. Multiple measurements along the track path need to be combined to determine the momentum and charge of the particle. The measurement equation describes the relation between track state and available hit measurement:

$$m_k = H_k x_k + \epsilon_k$$

Herein, m_k are the quantities that can be measured by the detector at step k and ϵ_k is the measurement noise which is overlaying the measured values.

Additionally, the Kalman filter also includes the full covariance matrix associated to the track state. For the sake of compactness, their definition and transformations have been omitted here, but the interested reader can find the full set of equations here [55].

Prediction step

Using the known track state of the step x_{k-1}, the prediction step transforms this previous state to the current state k. x_k^{k-1} contains the best estimate of the state at step k using only the information from the previous track state and its associated error: no new measurement has been integrated, yet.

$$x_k^{k-1} = F_{k-1} x_{k-1}$$

In the case of track reconstruction, this transformation propagates the track state from the detector layer where kit $k-1$ was located to the detector layer where the next hit k is located.

Filter step

During this procedure, the previous knowledge about the track state x_k^{k-1} is combined with the next hit measurement k.

$$x_k = x_k^{k-1} + K_k(m_k - H_k x_k^{k-1})$$

Herein, K_k is the Kalman gain matrix, which is computed from the covariances of the predicted state x_k^{k-1} and the measurement noise ϵ_k. The complete definition of K_k can be found in paper [55].

Depending on the uncertainty of the measured values with respect to the uncertainty of the predicted state, the final combined state x_k will include more contribution from one or the other.

The prediction and filter step are performed one after another, until all hits n have been integrated into the final track state estimation x_n.

7.4. Track Finding

The CMS Track Reconstruction can be separated in two main areas of concern: The track finding part is responsible for assigning measured tracker hits to track candidates. The track fitting part is employed to achieve a best estimate of the particle parameters at various positions in the tracking system.

For both tasks, the Kalman filter method is applied to refine the knowledge about the track candidates in the tracker.

The track finding procedure consists of multiple, iterative stages [56]. Early stages are tailored to find high momentum, centrally produced tracks. These tracks have a low curvature and are only weakly affected by material effects. For this reason, the requirements on possible track candidates can be quite strict. Once hits have been assigned to a track in these early stages, they are masked for the following stages which limits the combinatorical complexity and allows for later stages to search for hard to find low-momentum tracks and tracks originating from displaced vertixes.

7.4.1. Seeding

Each tracking iteration starts with an initial seeding step to initialize the track state for the Kalman-based track building. An initial idea of the direction and momentum and the associated errors of a possible track is necessary to allow the track building to make informed decisions which hits might belong to a track.

The various tracking iterations of CMS use compatible hit pairs, requiring two hits, or hit triplets, requiring three hits, in the inner layers of the tracker.

7.4.2. Track Building

The track building process takes the inner, coarse estimate resulting from the seeding stage and adds compatible hits on following tracker layers using the Kalman filter method. By using this principle, the errors on the track parameters can be decreased and the assignment probably of a hit to a track candidate can be evaluated.

Starting from the track parameters and its corresponding errors, the trajectory is extrapolated to adjacent layers to gather a list of sensor modules where the possible next hit could be located and on which sensor location the hit is expected to be found [56].

During this extrapolation, material effects on the trajectory are included to account for energy loss and multiple scattering as the particles pass through the tracker volume.

All hits found within a configurable tolerance of the estimated hit location will be used to form a new track candidate. Furthermore, a so-called invalid hit is added at the extrapolated hit location. This allows to find trajectories, where one or more hits are missing due to detector inefficiencies. Only a limited number of these invalid hits is accepted per track candidate, before the candidate is dropped.

One important, final step is to combine the extrapolated track state with the new-found hits, wherein one new track candidate is created for each compatible hit. This is done using the Kalman filter method and thus taking into account the size of the errors associated with each track parameter when performing the combination.

This procedure is now repeated with the new track candidates to look for hits in the following tracker layers. Fake combination of hits, which have been miss-idenfied as track candidates, will most likely not continue to be followed, as there will be no hits found which are compatible with their track state.

Figure 7.1 shows the track finding efficiency for muons in measurement data and compares this performance to the one achieved on events generated with Monte-Carlo simulation.

7.5. Final Track Fit

The output of the track building stage is a set of hits assigned to one track. To get the best-estimate of the track parameters, a dedicated fit procedure is performed which also uses the Kalman filter method to arrive at the final track parameters.

The procedure is similar to the track building: It combines the initial track state with measurements in the following layers starting from the innermost hit. The difference during track fitting is that only hits belonging to the track are used as an input, therefore removing the ambiguity resolution which was necessary in the track finding process.

Once the very last hit has been reached, the measurement information of all hits have been combined into the track state. But as the best-estimate of the track parameters is also desired at the interaction point, a so-called smoothing is performed. Therein, the outermost track state is used, but with greatly increased errors, to perform a outward-in Kalman filter combination. The final track parameters at each hit will then be the parameters resulting from Kalman-combining the result of the fitting and smoothing at these hit.

During the fit procedure the strength and direction of the magnetic field and the material effects along the particle trajectory need to be considered.

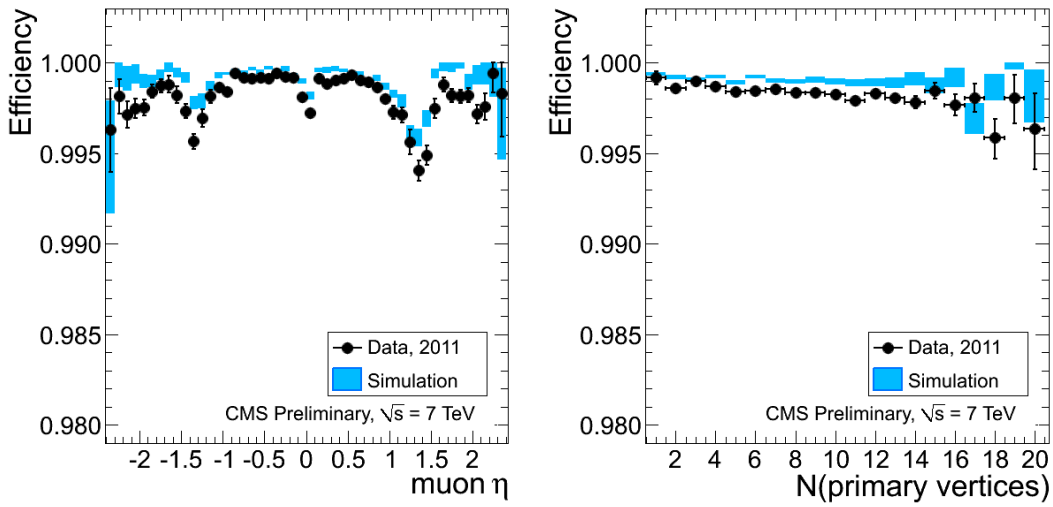


Figure 7.1.: Tracking efficiency of muons originating from $Z \to \mu\mu$ decays using the tag-and-probe method. The left plot displays the efficiency depending on the η position of the muon, the right one shows the efficiency as a function of the number of primary vertices reconstructed in the event [56].

7.6. Iterative Tracking Procedure

The CMS track reconstruction employs multiple steps of track finding and fitting. Each of these stages takes the unassigned hits as input, performs a pattern recognition using a Kalman filter to detect particle tracks and creates a list of found tracks. The hits assigned to these tracks are not passed on to the following stage, thus reducing the possible combinations and allow to perform a seeding for more displaced and low transverse momentum tracks.[57]

To reconstruct the recorded events of the 2012 data taking, seven iterative tracking steps were used[57]. The seeding in the first two steps requires tracks to originate close to the beam spot and to have three hits in the pixel layers. The later steps allow for more relaxed seeding criteria, as a portion of the hits have already been assigned to tracks. This enables the later stages to perform seeding with hit pairs and for tracks which have only hits in the strip layers.

7.7. Vertex Reconstruction

In the nominal LHC run conditions, multiple proton-proton particle interactions occur during the same bunch crossing, resulting in more than one primary vertex per measurement. As these vertices cannot be observed directly, they must

inferred from the reconstructed tracks which originate from them.

To achieve this within a high-luminosity environment, CMS employs a sophisticated deterministic annealing clustering method [58]. Therein, tracks are softly assigned to vertex candidates, meaning they can belong with the probability p_n^i to a range of vertex candidates. Furthermore, a so-called temperature T is used to controls the softness of this assignment [56]. The algorithm starts with a high value of T, which in turn allows a very soft assignment of tracks to vertices and, in principle all tracks can be assigned to one on vertex. The temperature T is step-wise decrease and new vertices are created, if necessary. Once T has reached its lowest value, the assignment of tracks to vertices is fixed.

The advantage of this algorithm is, that it is able to determine the number and location of primary vertices and compute the track to vertex assignment at the same time. Furthermore, it is able to achieve a good vertex separation, even in the high-luminosity environment present in the LHC.

Figure 7.2 shows two plots which illustrate the vertex position resolution of the CMS reconstruction in relation to the number of tracks which originated from the vertex. The two displayed datasets differ in the mean momentum of the contained particles.

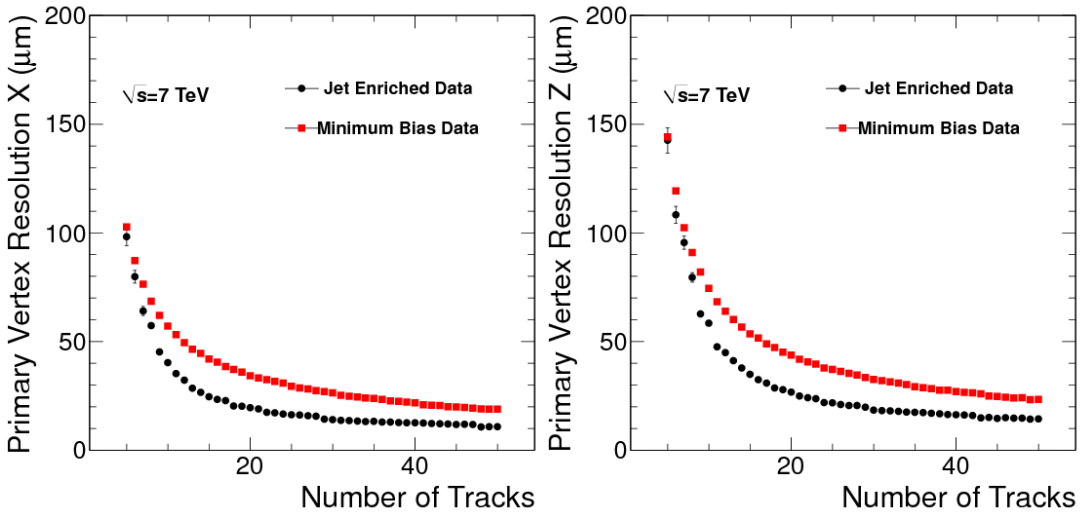


Figure 7.2.: The resolution of the primary vertex reconstruction in x and z direction. The two displayed datasets differ in the mean momentum of the contained particles [56].

Chapter 8

Computing Challenges in High Energy Physics

The results achieved in the high energy physics research field in the last decade were not only possible due to improvements in detector and accelerator technologies, but also to the evolution in computing and software technologies.

The increasing processing power of CPUs were an enabling technology which allowed the experiments to scale the measurement complexity and data rates to levels necessary to achieve ambitious physics goals. Many other technologies, like faster sensors, high-bandwidth networks, improved storage capabilities and miniaturization of hardware components also played important roles. For the following, the focus will be on the aspect of micro-processors and the computing capabilities they provide.

The need to build colliders and detectors at the edge of the technological limit is driven by the properties of the physics under study. Very rare processes, like the top production in the case of the Tevatron and the Higgs in case of the LHC, can only be properly studied if a sufficient number of events has been recorded during the lifetime of the collider. A higher beam energy is necessary to increase the production cross section for rare processes or even kinematically access possible processes involving heavy particles, like in SUSY models.

Figure 8.1 shows the evolution of the total amount of stored data for various completed, current and future collider experiments. The order of magnitude increase in storage requirements over the years is clearly visible. Although the amount of stored data cannot be directly linked to the necessary overall computing resources, like CPU hours, the requirements for these can be expected to be of the same order of magnitude.

An increased data rate and high beam energy also increase the computational challenges for all systems involved:

Online Trigger System

The online trigger system has to be able to cope with the high data rate delivered by the detector and accept or reject each acquired event within a certain timeframe.

Event Reconstruction

In proton-proton collisions, events with higher collision energy have a higher particle multiplicity. Performing pattern recognizion to identify individual tracks in this dense environment, where many hits are located in a small volume, increases the combinatoric complexity of the used algorithms and thus the runtime. Following steps, like the processing of energy deposits in the electromagnetic and hadronic calorimeters and jet clustering, require more processing time, as well. This increases the time needed for the full reconstruction of individual events.

In the case of the LHC, a higher instantaneous luminosity delivered by the collider comes with the downside of an increased pile-up contribution. The reconstruction of the additional particles from the non-primary collision increases the runtime of the reconstruction algorithms significantly and is further discussed in the following section.

Monte Carlo Event Simulation

The production of Monte Carlo datasets becomes more time consuming with the high-multiplicity events resulting from high-energetic collisions. For example, the full simulation of the decay $Z \rightarrow e^+e^-$ within the CMS detector takes around 51 seconds with a center-of-mass energy of 8 TeV and 117 seconds with 14 TeV [59].

Furthermore, a larger dataset recorded by the detector also requires a comparably big set of Monte Carlo simulated events. While a one to one ratio between the amount of recorded and simulated events is considered the lower limit, it is more desirable to have two to three times more Monte Carlo events than recorded events to be able to approximate the tails of distributions correctly. The computing model of the upcoming Belle II experiment even foresees a 1 to 6 ratio between amount of recorded and simulated events [60].

Monte Carlo events need more processing time than recorded events: After a generator, like Pythia [44] or Herwig++ [61], created the initial particles and their four momenta, they must be propagated through the detector with Geant4 and their interaction with active detector elements must be computed. Subsequently, the same event reconstruction as for data is performed.

In the year 2012, the ATLAS collaboration utilized around 50% of its grid resources for Monte Carlo production and around 20% for the event reconstruction of the Monte Carlo datasets [62].

Physics Analysis

The processing done to perform a physics analysis with the recorded data is very diverse and strongly depends on the goal of the concrete analysis. One thing in common across all analyses is the need to load either recorded or Monte Carlo events from storage systems. Therefore, the runtime of user analysis strongly depends on the speed of the storage system and is affected by the size of an individual event and the overall amount of the dataset.

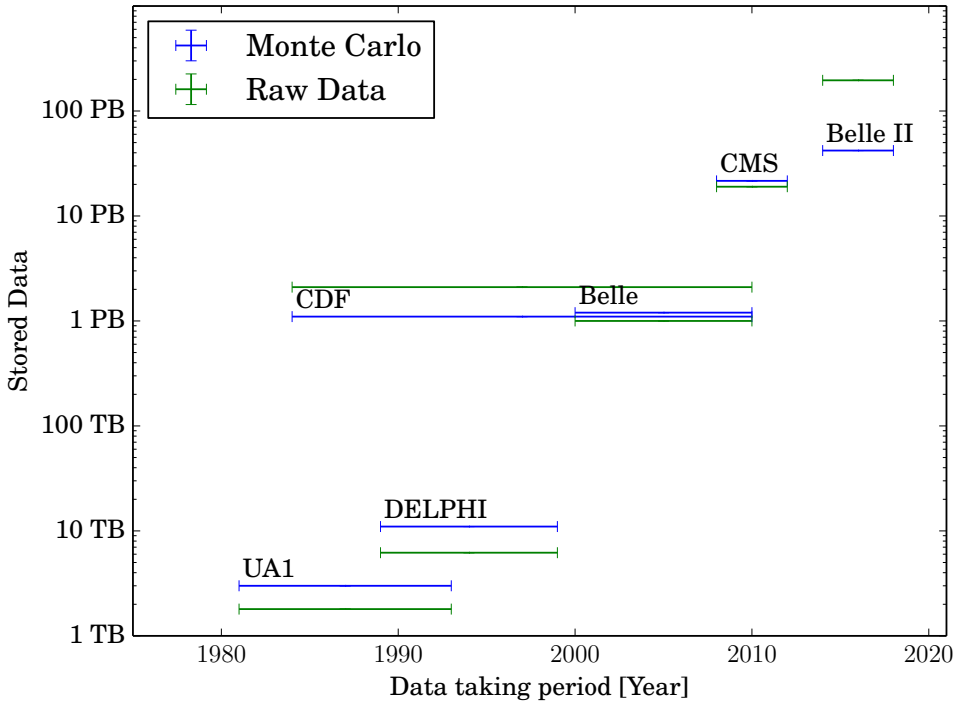


Figure 8.1.: The amount of stored data of past (UA1, DELPHI, CDF, Belle), current (CMS Run-I) and future (Belle II) particle collider experiments. Note that the data taking of CMS will continue in 2015 and the quantities for Belle II are projections based on the expected performance of the accelerator and detector. Created with data from [63, 64, 65, 66, 67, 68].

8.1. Influence of Pile-up on the Event Reconstruction Runtime

High-luminosity proton-proton experiments like the LHC collide many protons in one brunch crossing to achieve the desired instantaneous luminosity. The particles

originating from pile-up collisions overlaying the primary collision can not be separated at recording time, but an event reconstruction is needed to efficiently apply pile-up subtraction techniques.

With the high-luminosity setup required from the LHC during the 2012 data taking and in the future, the contribution of pileup events to the overall reconstruction time is significant.

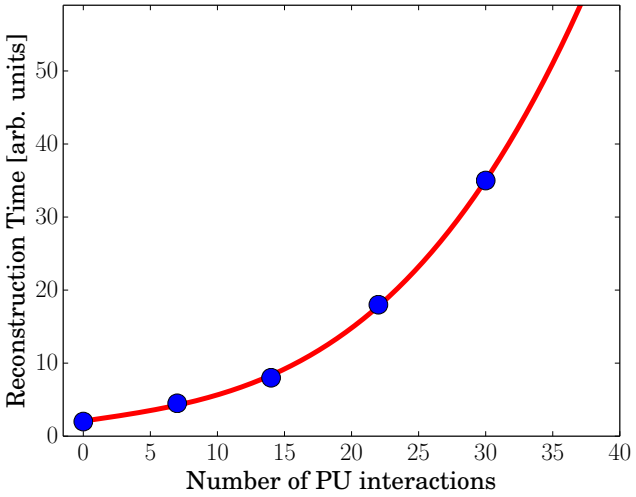


Figure 8.2.: Runtime of the CMS Reconstruction with QCD events for various pileup scenarios. This software version has been used for the 2011 data taking. To cope with the increased pile-up in the data taking of the year 2012, a set of optimizations have been applied [69], some of which will be discussed in this work. Own work with data from [69].

Figure 8.2 displays the overall CMS event reconstruction time for events with a varying number of pile-up interactions. The plot also shows a polynomial function of the 3rd order which has been fitted to the time measurements. The CMS software version benchmarked for this plot was used for the data taking in the year 2011 as an example. Reconstruction algorithms of other collider experiments show comparable behaviours [70]. The reconstruction runtime scales polynomially and shows a drastic relationship between necessary compute resources and event complexity for the case of CMS.

Fortunately, the parameters used by the reconstruction algorithms can be tuned and software optimizations can be applied to decrease the overall runtime. A wide range of these techniques have been successfully applied to the CMS software in preparation for the data taking in the year 2012 [69].

The currently foreseen LHC run conditions for the restart of the machine in 2015 deliver on average 43 pile-up collisions overlaying each primary collision.

This means that the average pile-up of 20 during the 2012 data taking is more than doubled. The design proposals for a possible high-luminosity LHC estimate the average pile-up collisions to be at 100 [71].

8.2. Challenges for Current and Next-generation HEP Experiments

As described in this chapter, the computing challenges will increase for today's and future collider experiments as the research goals become more sophisticated. To still be able to provide online triggering, event reconstruction and Monte Carlo simulation with the best possible physics performance, available computing resources must be utilized to the fullest and possible new platforms must be evaluated.

In this document, techniques to better utilize the available hardware will be introduced and performance improvements in the CMS reconstruction software and Geant4 simulation suite will be discussed in chapter 10.

Furthermore, a fast, GPU-based method to reconstruct particle tracks measured in the complex CMS detector will be described in chapter 11.

Chapter 9

Modern CPU and GPU architectures

Modern central processing units (CPU) are highly sophisticated integrated circuits which evolved significantly during the last 50 years of semi-conductor development. During this time, one of the major dimensions of improvement was the internal clock frequency of CPUs.

The Intel 4004, one of the first fully integrated CPUs, was released in 1971 and was operated with a clock frequency of $740\,kHz$. Comparing this to the $4.0\,GHz$ provided by a modern Intel Xeon EC3 CPU, introduced in the year 2013, an increase of almost factor 6000 in clock frequency was realized.

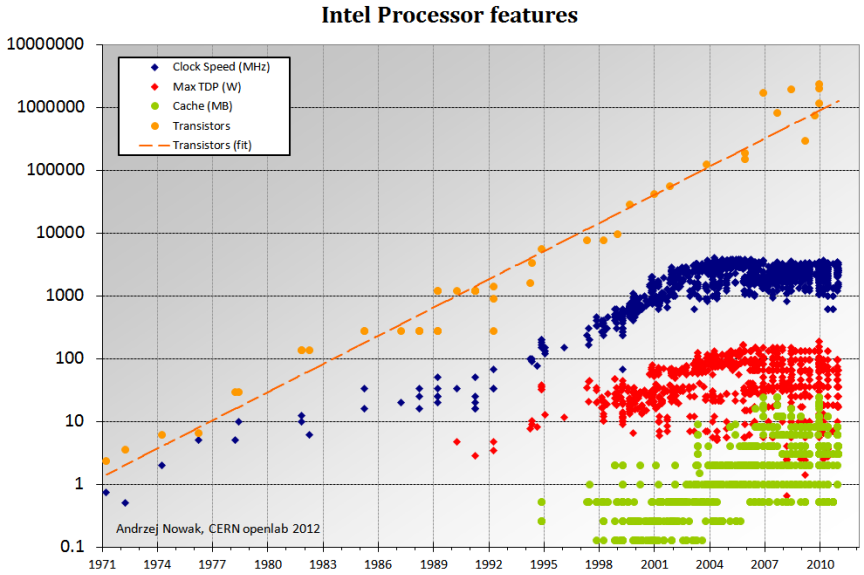


Figure 9.1.: CPU clock speed, transistor count and cache size over the last 40 years [72].

While this gain in clock rate improved the computational throughput, the number of transistors also increased significantly over the years. This development follows the observation coined as Moore's law, stating that the number of functional components (transistors) on one physical chip approximately doubles every two years [73].

While at first glance, both scaling regimes, frequency and transistor count, allow to increase the processing power, one comes with a significant drawback.

The power consumption of the entire CPU is highly dependent of the clock frequency. This relation can be approximated by the formula [74]

$$P = CV^2F \tag{9.1}$$

where P is the consumed power, C the capacity of the CPU circuits, V the voltage and F the clock frequency.

Increasing the frequency also increases the power consumption on the chip. Despite smart power management systems in modern CPUs, the excess heat generated by the chip must be dissipated. Once a certain clock frequency has been reached, it is not feasible, neither efficient, to generate more and more heat on the chip. For this reason, the CPU vendors ceased to increase the clock count around the year 2002. This allows to limit the thermal design power (TDP), which is the maximum heat generated by the chip.

Figure 9.1 plots the CPU clock speed, TDP, transistor count and cache size of Intel processors ordered by their release date. The plateau in clock frequency, which has been reached around 2002, is clearly visible.

In contrast to clock frequency, the transistor count can still be increased as predicted by Moore's law. Therefore, the primary focus of chip designers now rests on optimizing the usage of the available transistors to increase the computational power of the chip.

The increased number of transistors can be employed, among other uses, to increase the register size. While the Intel 4004 was a 4-bit CPU, most modern desktop and server CPUs have a 64-bit wide architecture. This enables the CPU to process more data during one clock cycle. Additional facilities like fast, on-chip memory caches and smarter instruction processing can also be implemented as more transistors become available.

This led to the introduction of super-scalar processors. CPUs supporting this feature take a serial stream of machine instructions as input, decode and analyze them for data-interdependencies. Instructions which are independent of each other are scheduled for parallel execution. To facilitate this, such CPUs have multiple execution units which can either perform arithmetic or memory operations. The benefit is, that multiple machine instructions can be processed during the cycle and the overall throughput of the CPU increases.

Furthermore, multiple compute cores can be implemented with the available

transistors on one physical package. This allows to distribute computations among the many cores, decrease the clock frequency and thus lowering the total energy consumption, and therefore the generated heat, of the whole system. Since increasing the CPU frequency is not feasible any more, employing multiple CPU cores became one of the primary ways to achieve the performance increase available in modern-day micro-processors.

9.1. Vector Units in Modern CPUs

Another possibility to increase the computational power of one processing core is to handle more data during one clock cycle. In this respect, vector units can apply one instruction on a whole range (vector) of input values. On a conceptual level, the term Single Instruction Multiple Data (SIMD) [75] is used to describe this type of processing.

Although modern SIMD instruction sets also provide operations on integer and bitwise inputs, the focus will be on operations on floating point numbers in the following as these are prominently used in HEP applications. The IEEE 754 standard defines the representation of floating-point numbers in binary form [76]. It also defines various levels of precision of these representations. In the following, the single-precision (using 32 bits) and the double-precision (using 64 bits) representations will be used.

While the SIMD concept has been applied to scientific computing since many years, it was introduced in the commodity CPU market by Intel with the MMX instruction set in the late 1990s. Since then, the compute capabilities of the SIMD instruction sets have greatly improved. Following MMX, the SSE2, SSE3 and SSE4 instruction sets were developed and introduced 128-bit wide registers, allowing 4 single-precision float values to be processed in one instruction.

The most recent SIMD instruction set is the AVX2 specification and features a register size of 256-bit, allowing to process 8 single-precision float values in parallel. This results in a theoretical speedup of a factor 8 for floating point operations. In reality, the actual speedup is lower as there is always some overhead involved when loading the values into the vector registers and some parts of the processing which can not be expressed with vector instructions.

Virtually all consumer x86 CPUs shipped since the year 2002 support at least the SSE2 instruction set. Therefore, SSE2 can be considered the lowest common denominator when deploying applications compiled with SIMD support.

Figure 9.2 illustrates the register size of the SSE2 and AVX instruction sets and how they can be populated with either single or double precision values.

The SSE and AVX specifications sets offer a wide variety of instructions, but a special focus has been put on processing floating point numbers of single and double precision. As a lot of the simulation and reconstruction code in HEP ap-

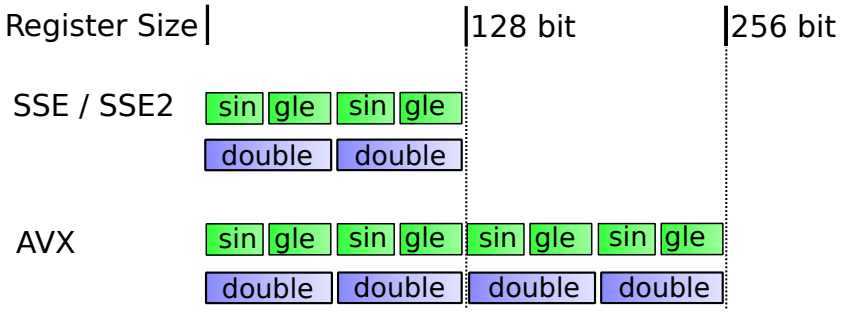


Figure 9.2.: Vector register size of the SSE/SSE2 and AVX instruction sets. The SSE/SSE2 registers can either contain four single precision or two double precision floating point values. The AVX registers can hold twice as many for each data type.

plications is using floating point data heavily, these instruction sets are especially beneficial to increase the runtime performance of these applications.

However, to take advantage of this additional computing power, the software needs to be adapted to use the SIMD instruction during its computing operations.

Chapter 10 will discuss how existing HEP applications can be adapted to benefit from the vector instructions and the runtime improvements which can be achieved.

9.2. Multi-Core Design

Another possibility to increase the throughput of one physical CPU is to integrate multiple independent processing cores into one physical micro-processor. While these cores share the same infrastructure, like connection to main memory, cooling and power supply, they can have their own computation units, registers and first level caches.

This allows the CPU to execute multiple instruction streams at the same time and can increase the computation capabilities of the the CPU significantly. In theory, the throughput can be increased be a factor N for a CPU with N cores. The speedup in real applications highly depends on implementation details and memory access patterns of the parallel application.

Early consumer CPUs started to use the Simultaneous Multi-Threading (SMT) technique to implement a fast, hardware-supported context-switching between two instruction streams on one physical core. Intel's implementation of the SMT concept, named hyper-threading, is widely available in Intel's product palette today and complements also multi-core CPUs. The performance improvements reported depend highly on the actual workload and range from almost no improvement to up to 20% in alogrithm runtime [77][78].

Modern Intel Ivy Bridge server CPUs offer up to 15 physical cores, which multiplies to 30 virtual cores when hyper-threading is enabled.

Nowadays, operating systems provide a sophisticated process scheduler to distribute the running process in the best possible way to the CPU cores. For this reason, a lot of existing applications can benefit from the multi-core capabilities without specifically supporting this feature the in their software design.

Multiple process instances of the same application can be started, wherein each instance uses only one CPU at a time and the operating system distributes the running processes across the available cores.

In the case of event simulation and reconstruction in HEP in general, and in CMS specifically, distributing the input task across multiple processes is feasible as the individual event are not interdependent. This technique is used in CMS computing centers to fully load also multi-core CPUs.

The price to pay for this approach is the increased memory consumption because each process has to have its own, dedicated memory region to store constant data, like the detector geometry or magnetic field maps. While the multi-process approach in HEP applications is feasible if the ratio between available system memory and CPU cores is around 2 GB RAM per CPU core, it becomes harder to maintain if more and more CPU cores become available, but the amount of memory does not increase with the same rate.

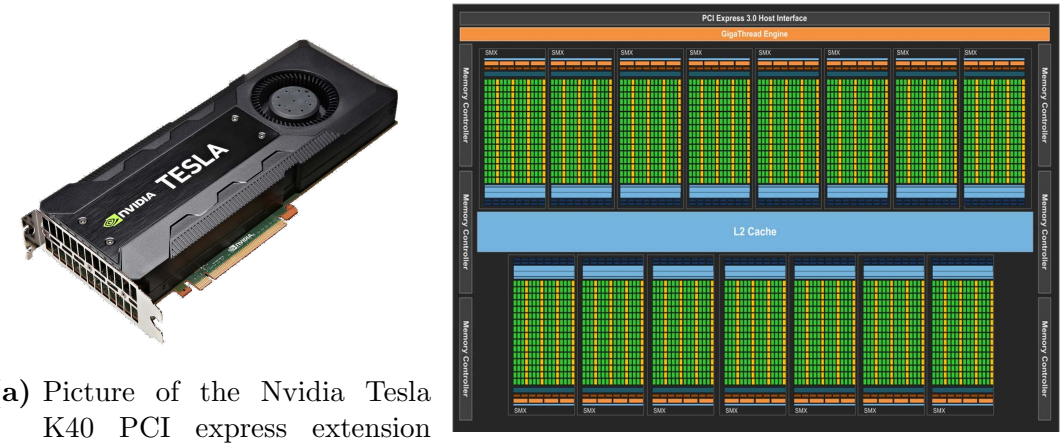
Chapter 11 documents how track reconstruction algorithms can be implemented that are able to take the parallel nature of modern CPUs into consideration and is able to use many CPU cores from within one process. This allows the sharing of constant resources, like detector geometry magnetic field map, and reduces the memory consumption when using many CPU cores at the same time.

9.3. Massively-Parallel Graphics Processing Units

From the field of computer game graphics originates a very different, in some ways much more radical, approach to reach the highest possible throughput with the available transistors. Instead of employing a high clock frequency to achieve a high throughput, Graphics Processing Units (GPU) rely on many small processors which work on the compute task in a parallel fashion. This allows for a high data-throughput with a low energy-consumption because a limited clock frequency is sufficient.

By now, the benefits of this processing architecture have been recognized also outside of the gaming community and GPUs are used in a wide area of applications, including signal processing, engineering simulation, weather prediction as well as theoretical and experimental physics.

One of the leading vendors of GPU devices, both targeting computer gaming and scientific applications, is Nvidia. This company offers the PCI extension card Tesla K40, designed for HPC computing. Compared to the $\sim 2\,GHz$ clock rate used in modern CPUs, this card employs a relatively low base clock rate of $745\,MHz$, but offers 2880 parallel processor cores [79].



(a) Picture of the Nvidia Tesla K40 PCI express extension card [80].

(b) GK110 full chip block diagram [79].

Figure 9.3.: Picture of the Nvidia Tesla K40 card and a diagram of the GK110 chip diagram.

A picture of the Tesla K40 extension card can be seen in figure 9.3a. The chip schematic of the Kepler architecture GK110, which is underlying the K40 product, can be seen in figure 9.3b. The main building blocks of this design are up to 15 streaming multi-processors (SMX). These are independent processor with 64 Kb of shared memory each and 192 individual CUDA processing cores. [79].

Each of the streaming multi-processors can execute up to 2048 parallel threads of execution, resulting in the capability of a GPU to process the workload in a

massively parallel fashion. The Tesla K40's on-chip memory is supplemented by 12 GB of on-board RAM which are connected via a memory bus with a bandwidth of 288 GB/s [80].

With this hardware layout, the Tesla K40 card is able to deliver a peak single precision performance of 4.29 teraflops and a peak single performance for double precision of 1.43 teraflops.

Compared to the 100 gigaflops provided by an Intel Core i7 Sandy-Bridge CPU, clocked at 3,4 GHz and having 4 cores, the GPU offers a significant gain in theoretical peak performance.

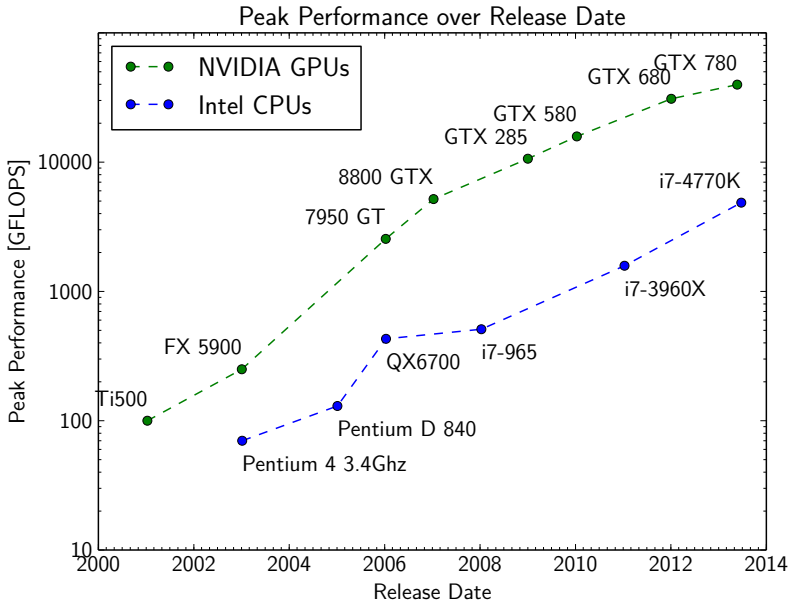


Figure 9.4.: GPU and CPU single precision peak performance comparison [81].

Figure 9.4 shows the plot of the peak performance of GPU and CPU products, ordered by their release date. Both, CPUs and GPUs profited from the increase of compute capabilities along the years. Only judging by peak performance, the GPUs are able to provide about an order of magnitude more compute power and this trend is stable across the last 10 years.

Chapter 11 describes a method to perform particle track reconstruction on GPUs, the necessary adaptions and the gains which are possible for HEP workloads by using GPUs.

Chapter 10

Vectorization Potential for HEP Applications

Floating point computations are an important ingredient to almost every HEP application. Significant time is spent with floating point operations and the evaluation of mathematical functions like the exponential and logarithmic functions. Therefore, taking advantage of the vector units in modern CPUs is a possible way to improve the runtime performance.

However, C++ code needs to be adapted to conform with the flat data structures required by vector operations. For big HEP application packages like CMSSW, which consists of around 5 million lines of code and many thousand locations where floating point computations are performed, such adaptations would require a major rework of the code base.

However, other ways exist to support vector units in applications. One option is to provide a fast and vectorized implementation of often used mathematical functions and use them instead of the default ones provided by the system libraries. This process and its benefits will be described in section 10.2.

Furthermore, especially compute-intensive hot-spots can be detected and sped-up using vector instructions. This process will be described in section 10.5.

10.1. Introduction to Vector Unit Programming

Although the SIMD instruction sets are a well-established feature of CPUs for some time, these extended functionalities could only be accessed by programmers willing to interleave their high-level C or C++ source code with quasi-assembler instructions called compiler intrinsics. The SSE2 and AVX technologies use different instructions. If all must be supported, the same algorithm has to be implemented several times.

Modern compilers, like the GNU Compiler Collection (GCC) [82], provide the possibility to transform regular C or C++ source code into vector operations. This process is called auto-vectorization and is transparent to the programmer.

The CMS Collaboration opted for the usage of open-source software technologies and therefore relies on the GCC compiler to build the complete CMSSW software stack. The presented results were achieved by using the GCC version 4.6.1, if not otherwise stated [78].

The benefits of using the auto-vectorization features of modern compilers instead of low-level intrinsics are manifold. The algorithms can still be expressed as high-level C/C++ code, which brings advantages for the development and maintainability. In addition, a programmer without SIMD-expertise is able to understand and extend the algorithms.

Furthermore, existing algorithms can be adapted to be auto-vectorized by the compiler. If intrinsic instructions are used, the algorithm must be reimplemented from scratch using the dedicated SIMD commands.

Arguably, the biggest advantage is portability. Once the algorithm can be auto-vectorized, the compiler is able to generate machine code for every available SIMD instruction set and thus support all hardware architectures. Also future SIMD specifications can be supported by simply upgrading to a new version of the compiler. Therefore, an existing auto-vectorizable C/C++ code fragment is guaranteed to benefit from advancements of the SIMD technology in upcoming microprocessors [78].

A polynomial of the order 3 can serve as a basic case to illustrate the auto-vectorization process and possible runtime gains. The code listing below shows the C++ code fragment which computes this polynomial for a list of input values. This code was compiled with GCC without vectorization (scalar), SSE and the AVX instruction set.

```
1  ...
2  std::vector<double> x (ArraySize);
3  std::vector<double> y (ArraySize);
4  ...
5  for ( size_t i = 0; i < ArraySize; ++i)
6  {
7    y[i] = a_3 * ( x[i] * x[i] * x[i] )
8           + a_2 * ( x[i] * x[i])
9           + a_1 *   x[i] + a_0;
10 }
11 ...
```

The runtimes for a varying size of the input element list was measured for the three compiled versions. The results obtained when executing the program on a

Intel Core i7-3930K CPU can be seen in figure 10.1. For this simple example, the auto-vectorization performs very well and the compiler is able to generate vectorized code targeting the selected instruction set. The SSE version is about 1.5 times faster than the non-vectorized, scalar version. Using the AVX instruction set, a speed improvement of a factor 3.3 can be achieved with respect to the scalar version.

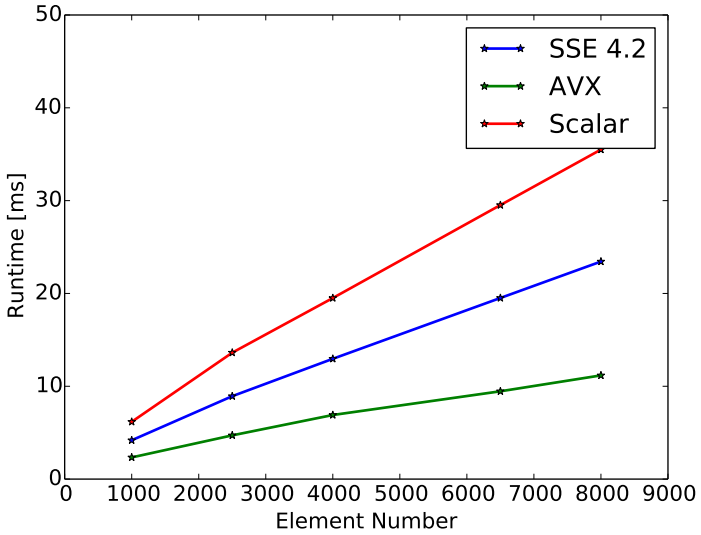


Figure 10.1.: Computing a basic polynomial of the 3rd order using either no vectorization (*scalar*) or various types of vector instructions(*SSE 4.2*, *AVX*) shows significant decreases in the overall runtime when using the vector intsructions.

Furthermore, existing algorithms can be adapted to be auto-vectorized by the compiler. If intrinsic instructions are used, the algorithm must be reimplemented from scratch using the dedicated SIMD commands.

10.2. The VDT Mathematical Library

Analysis shows that existing HEP applications packages spend a significant amount of time in evaluating mathematical functions like logarithm, exponential, sine, cosine and others. For the Geant4 material simulation of the CMS detector, between 12 and 30 percent of the overall runtime can be attributed to mathematical functions [78]. For the CMS event reconstruction, around 9 percent of the runtime is spent evaluating mathematical functions [78].

10.2.1. Existing Mathematical Libraries

The system math library shipped with most Linux distributions is called `libm` [83] and offers a guaranteed accuracy but is not optimized for the use with vector units. Optimized `libm` replacements are offered, among others, by Intel [84] and AMD [85]. They provide single and double precision floating-point mathematical operations which guarantee a certain accuracy in relation to the `libm` implementation. Both implementations are closed source and in the case of Intel require the purchase of a license.

10.2.2. Introducing the VDT Library

To allow the developers of CMS and the wider scientific community to benefit from a fast and vectorized mathematical library, the decision was taken to develop a royalty-free and open-source implementation within the CMS and CERN software groups.

At the basis of the **V**ectorise**D** ma**T**h (VDT) library [78] is the implementation introduced by the Cephes library[86]. The implementation uses the Padé approximants which have been adapted and re-engineered for `vdt` to be auto-vectorized by modern compilers. In Padé approximations, the function to compute is expressed by a rational function with an arbitrary order. The independent terms in the Padé approximation allow for a fast computation, even in scalar mode, as the terms can be distributed over the execution units of super-scalar CPUs.

The `vdt` library provides fast and auto-vectorizable implementations in single and double precision floating point for the mathematical functions **sin**, **asin**, **cos**, **acos**, **sincos**, **tan**, **atan**, **atan2**, **log**, **exp** and **1/sqrt**. These functions are provided in C++ header files, which allows the compiler to inline the function body. This allows the compiler to apply its optimization passes not only on the mathematical function, but also incorporate the surrounding code. This can potentially result in better optimized and larger blocks which can be auto-vectorized.

The `vdt` library is provided as open-source under the GNU Lesser General Public License (LGPL3) licence [87]. This allows users to include `vdt` in open-source as well as commercial products free of charge. The `vdt` source code and further documentation can be downloaded on the project website [88].

As the `vdt` library is provided as a source code release and does only rely on the C++ standard, it can be compiled with a variety of compilers and hardware platforms. This library design allows to achieve the maximum portability, both in terms on vector instruction set and machine architecture. The library has been successfully compiled with the `GCC` and `clang` compilers for the SSE and AVX instruction set on x86 CPUs and for the ARM platform. As the SIMD instructions will be generated by the compiler, highly-optimized machine code can be generated for each platform.

10.3. Speed and Accuracy Comparisons for VDT

The two primary criteria when designing `vdt` were the speed improvements with respect to the `glibC libm` and very good accuracy of the approximation.

The quality of the approximation can be best validated by comparing bits in the floating point representation itself. Floating point numbers in the IEEE representation [76] are expressed by a sign bit (s), a mantissa (m) and an exponent (e). An arbitrary floating point number (F) within the range of the exponent can now be written as:

$$F = s * m * 2^e \tag{10.1}$$

To compare the quality of the `vdt` approximation, the number of differing bits between `vdt` and `libm` in the mantissa, starting at the least significant one, is compared.

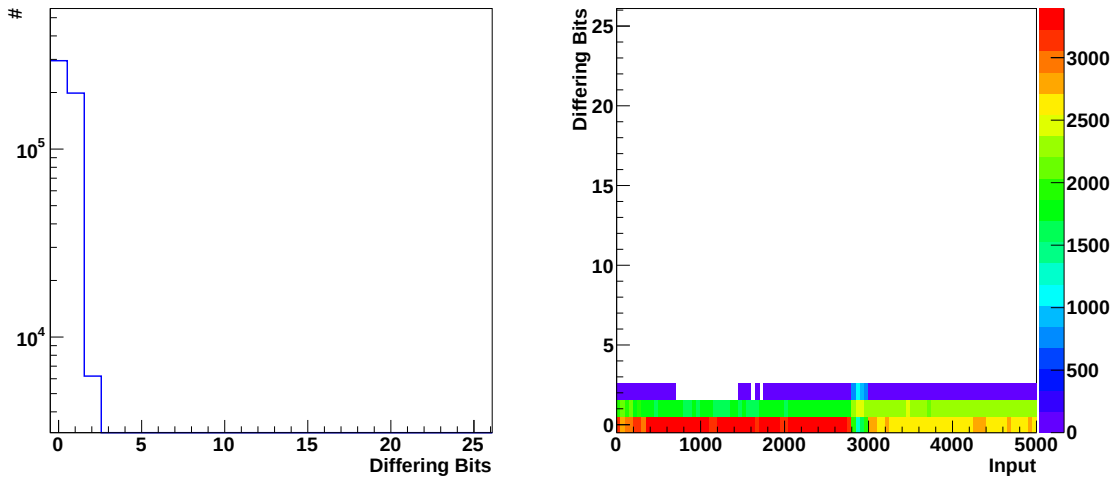
Table 10.1 compares the accuracy of `vdt` to `libm` as reference. The second column lists the maximum number of differing bits starting from the least significant bit in the whole validation run. This number is a measure for the most pessimistic scenario possible when using `vdt` as a replacement for `libm`. The column number 3 lists the number of differing bits, averaged over the whole test sample. This shows, that the maximum difference estimates are corner cases and that, on average, the amount of different bits is well below 1.0 for all functions implemented by `vdt`.

Figure 10.2a and 10.2b shows the evaluation of the accuracy of `vdt` with respect to the `libm` reference implementation for a selected input number range of $[0, 5000]$ for the double precision log function. As visible in figure 10.2a, the `vdt` approximation results in either the same output values or differs by only one bit for the predominant part of the input numbers evaluated.

Figure 10.2b visualizes the quality of the `vdt` approximation depending on the sector of the input numbers. While the guaranteed maximum divergence of two bits is uphold throughout the whole interval, the quality of the approximation might change. Around the input number 1000, the approximation does not diverge more than one bit. In contrast, around the number 3500, more 1-bit and 2-bit discrepancy can be observed.

More validation plots of the `vdt` accuracy can be found in appendix A.

Table 10.2 lists the execution speed of the implemented math function for `libm` and two different compile options for `vdt`. The scalar option is to compile `vdt` without any vector instructions. This is the lowest common denominator and will run on every x86 system. The second option listed in table 10.2 is still scalar, but uses the fuse multiply add (FMA) instruction introduced in the most recent Intel Haswell architecture. This machine code can perform a multiply and an add operation on a floating point number with one instruction.



(a) Histogram of the maximum number of differing bits of the `vdt` approximation wrt. to the `libm` reference implementation

(b) Frequency of the maximum number of differing bits of the `vdt` approximation wrt. to the `libm` reference implementation for the evaluated input range

Figure 10.2.: Both plots show the evaluation of the approximation accuracy of `vdt` with respect to the `libm` reference implementation for the double precision log function. 500.000 random input values, uniformly distributed in the the considered range $[0, 5000]$ were used for this study.

Table 10.1.: Accuracy and interval of definition of `vdt` compared to the `libm` reference implementation for double precision [89].

Function	Interval of Definition	Max. differing bits	Avg. differing bits
Exp	$[-708, 708]$	2	0.14
Log	$[0, 1e307]$	2	0.42
Sin	$[-2\pi, 2\pi]$	2	0.25
Cos	$[-2\pi, 2\pi]$	2	0.25
Tan	$[0, 2\pi]$	2	0.35
Asin	$[-1, 1]$	2	0.32
Acos	$[-1, 1]$	8	0.39
Atan	$[-1e307, 1e307]$	1	0.33
Atan2	$[-1e307, 1e307]$	2	0.27
Isqrt	$[0, 1e307]$	2	0.45

All runtime numbers have been measured on a Intel Core i7-4770K CPU running at $3.50\,GHz$. The operating system was Scientific Linux 6 with the GNU libc

version 2.12-1.107.el64.4 and the vdt functions were compiled with GCC version 4.8.

The significant speed improvements provided by vdt is visible in table 10.2. Due to implementation details, the vdt speedup is different for each mathematical function. The runtime for the exp function, which is one of the most widely used in HEP software, is improved by a factor of almost 13. The log function is sped up by a factor of almost 3. The FMA instruction brings slight runtime improvements to most of the vdtfunctions.

Table 10.2.: Runtime comparison of double precision math functions between vdt in scalar mode and glibC libm. The time is in nanoseconds per calculated value [89].

Function	libm [ns]	vdt scalar [ns]	vdt scalar FMA [ns]
Exp	102	8	5.8
Log	33.3	11.5	9.8
Sin	77.8	16.5	16.5
Cos	77.6	14.4	13.2
Tan	89.7	10.6	8.9
Asin	21.3	8.9	6.9
Acos	21.6	9.1	7.3
Atan	15.6	8.4	6.7
Atan2	36.4	19.9	18.9
Isqrt	5.7	4.3	2.8

Even though the scalar version already shows significant speed improvements, the may primary design goal of vdt is to take full advantage of the available vector instruction sets. Table 10.3 compares the runtime of the scalar version with two flavors of vector instruction. The same C++ source code is evaluated here, but merely the compiler was instructed to generate machine code for the indicated vector architecture.

The SSE vector instruction set can be considered the lowest common denominator, as it is supported by virtually all x86 machines used in the HEP community today. Here, the expected time 2 decrease in runtime, which can be expected with double precision numbers, is achieved by most of the vdt functions, and exceeded by some.

Using the more powerful AVX2 instruction set, all of thevdt functions are able to achieve a better runtime, but only some (**exp**, **sin**, **cos**) are actually twice as fast as with SSE. For some (**asin**, **acos**) a modest increase can be achieved with larger vector registers because the vector opportunities in the mathematical expressions can already be almost fully expressed with SSE.

Table 10.3.: Runtime comparison between `vdt` in scalar mode and the library compiled to take advantage of the SSE and AVX2 vector instruction sets [89].

Function	`vdt` scalar [ns]	`vdt` SSE [ns]	`vdt` AVX2 [ns]
Exp	8	3.5	1.7
Log	11.5	4.3	2.2
Sin	16.5	6.2	2.6
Cos	14.4	5.1	2.3
Tan	10.6	4.4	3.2
Asin	8.9	5.8	5
Acos	9.1	5.9	5.1
Atan	8.4	5.6	5.1
Atan2	19.9	12.7	8.4
Isqrt	4.3	1.8	0.4

10.4. Application of `vdt` to Geant4

To study the real-world gains possible with `vdt` on big software packages, two approaches are possible:

- Replace hot-spot mathematical functions with the `vdt` version

  Modifications to specific areas of the code are necessary to call the `vdt`functions.

- Replace all calls to mathematical functions by their respective `vdt`version

  The library pre-load technique can be used to fully replace all calls to selected function without recompiling the targeted application. The dynamic linking in Linux system already offers this functionality.

For the following study, the pre-load technique was employed to replace all calls of Geant4 to `libm` with their respective `vdt` counterparts. This allows for an non-intrusive validation and it is conservative, both in terms of runtime improvements and physics performance. Due to the nature of dynamic linking, which happens during runtime and not compile time, the compiler is not able to inline the `vdt` mathematical functions and apply further optimizations. Secondly, by replacing all mathematical functions, the `vdt` approximations will be used in all areas of Geant4 and potential changes to the physics performance can be easily spotted during the validation process.

For the following tests, top anti-top events with a center-of-mass energy of 8 GeV were simulated with Geant4 within the full CMS geometry.

The physics objects of the `libm` and `vdt` versions of Geant4 have been compared using the full suite of the CMS validation workflow, which contains more then 100.000 histograms. The output of the `vdt` version shows is compatibile to the `libm` reference implementation.

Two different type of workflows have been studied. The first workflow is to evaluate speedup achieved, when Geant4 runs for an extended amount of time, a typical scenario when performing Monte Carlo production on the LHC computing grid. For this, 50 events were simulated with Geant4 and an overall speedup of 6.6% can be achieved with `vdt`.

Measuring the runtime of just one event simulated by Geant4 shows how the `vdt`library can improve the startup time. When using `vdt`, the for event is processed 25.0% faster then when using `libm` [89]. The speed improvements available with `vdt` are more pronounced during the first event, as values need to be initialized, which make heavy use of mathematical functions.

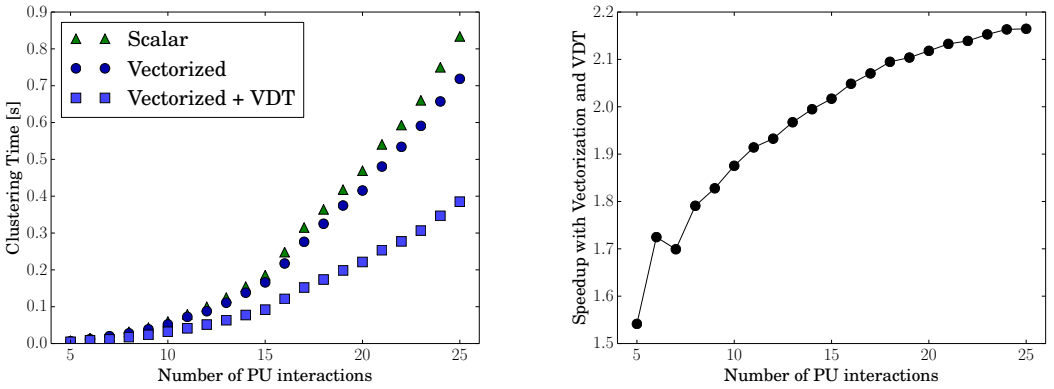
With the most recent Geant4 10.0 release, the `vdt` library was introduced to improve the runtime of the overall application [90]. This will enable the speed benefits of `vdt` to all users of Geant4, independent of the operating system or platform.

10.5. Improving the Runtime of CMS Vertex Clustering

As described in section 7.7, the CMS event reconstruction relies on a deterministic annealing algorithm [58] to compute the location of the primary vertices in one event. Especially with an increasing number of pileup-interactions, this reconstruction step in CMSSW amounts for a considerable part of the overall application runtime [89]. To prepare for the 2012 data taking conditions, the runtime performance of this algorithm needed to be improved. Due to the intensive use of floating point computations in this algorithm, it is an ideal case to apply the auto-vectorization technique.

After adapting two compute intensive loops of the deterministic annealing algorithm, the `GCC` compiler was able to auto-vectorize the performed mathematical operations. Furthermore, a large fraction of the algorithm runtime is spent in the evaluation of the exponential function. The previously introduced fast and auto-vectorizable exponential function of the `vdt` library was used to speedup this calculation. The decision was taken to use the SSE instruction set in the compilation process, as it is available on all processors installed in the data centers which provide computing resources to the CMS collaboration.

The physics output of the improved algorithm has been fully validated using the extensive set of quality monitoring tools of CMS and a perfect agreement with the previous, scalar implementation is observed. Neither the vectorization



(a) Runtime of the vertex clustering for various pile-up scenarios and implementations.

(b) Speedup of the vectorized vertex clustering using `vdt` with respect to the scalar version.

Figure 10.3.: Runtime and speedup of the scalar and the vectorized implementations and the vectorized version using the `vdt` library for different pile-up content in the event. The mean pile-up interactions for CMS was 20 in the year 2012.

process, nor the application of the approximate exponential function provided by the `vdt` library did influence the physics performance.

Figure 10.3a shows the mean runtime of the scalar and the vectorized implementations and the vectorized version using the `vdt` library for different pile-up content in the event. For this runtime study, 10000 Events recorded by the CMS detector during the year 2012 with a center-of-mass energy of $\sqrt{s} = 8\,\text{TeV}$ have be used to quantify the runtime of the implementations.

Figure 10.3b displays the speedup which can be achieved by the vectorized implementation with respect to the default one. The benefits of vectorization are especially pronounced for more pile-up contribution, meaning more particle tracks in the event. In these cases, the vertex clustering algorithm has to iterate over a bigger set of input tracks and perform the computations which have been sped-up by the use of vectorization and `vdt`. Starting with 16 pile-up interactions the speedup is more than a factor of 2. The average number of pile-up interactions per event in the year 2012 for CMS was 20 [22].

The SIMD-enabled version of the vertex clustering algorithm was used for the offline recontsruction of the LHC data in the year 2012. This allowed to achieve an excellent vertexing performance in high-luminosity conditions while staying within the runtime envelope assigned to the vertex clustering.

The LHC run conditions for the restart in the year 2015 will increase the average pile-up collisions per event to 43. With the goal to achieve an excellent vertexing performance in these conditions, the offline reconstruction will continue to use the

SIMD-enabled vertex clustering presented here.

Also the online reconstruction used in the high-level trigger faces a more difficult measurement environment with the increased pile-up contribution. Due to the vectorization optimizations presented here, the deterministic annealing vertex clustering became a possible candidate to be used in the more time-constrained HLT farm. Studies are underway to estimate to which extend the DA vertex clustering can improve the physics performance in the HLT compared to the faster, but less elaborate algorithm used before.

Chapter 11

Fast track reconstruction for massively parallel hardware

As outlined in chapter 9, GPUs can provide a significant advantage in terms of floating point peak-performance and energy consumption compared to classical CPUs. This is made possible by the radically different design decisions taken by the GPU manufacturers, above all to use many basic compute cores instead of only a few, but very powerful ones.

These differences in design are reflected in the programming techniques necessary to fully exploit the capabilities of GPUs. Complex algorithms and programs, which have been designed and optimized for classical CPUs, are likely to not perform well on GPUs. Some of the fundamental features GPU-adapted implementations must address are:

Massively parallel

The NVIDIA Tesla K40 card provides 2880 parallel processor cores [79], more than a factor 100 of what is provided by a even high-end CPUs. Algorithms must be designed and implemented to provide sufficient workload for all of these processors to be busy at the same time. Failing to do so will result in parts of the GPU being idle and wasting bought resources, and therefore money.

While classic CPUs can be fully loaded with around 10 parallel threads, GPUs require 1.000s, even better 10.000s, parallel threads of execution.

Memory layout and size limitations

The high-end NVIDIA Tesla K40 card has up to 12 Gb of on-board RAM, which is available to application usage. This memory is shared between all processing cores, the access has a high latency and access operations from multiple cores at the same time must be serialized to protect for memory corruption. To allow for faster memory operations, each multi-processor has its own shared memory, which

can be exclusively accessed by threads running on it. In case of the NVIDIA Tesla K40 card, the shared memory is 64 kb large.

To achieve peak performance, this memory region must be managed properly to have the data necessary for a computation available close and fast to the processing core using it.

Furthermore, memory buffers on the GPU must be pre-allocated and have a predetermined size. Some high-end cards support dynamic memory allocation during execution on the GPU. This functionality is only available on a small subset of hardware and is not portable to other vendors than NVIDIA.

Data transfer overhead

The on-board memory of the GPU is logically and physically separate from the host-PCs memory region. Therefore, necessary data needs to be copied from the host to the GPU prior to the computation and vice versa.

GPU implementations must make sure that the data-transfer time is well amortized by the actual work done on the GPU. This latency can also be hidden, by streaming data continously to the GPU, while the previous computation is still in progress.

Floating point precision

The GPU market emerged from the computer game segment, there was initially no need for high-precision floating point operations. Computer graphics works well with the limited representation of single precision floating point numbers, but many scientific applications require double precision floating point, as they want to reduce the error introduced by the floating point representation as far as possible.

Even modern GPUs, which are targeted at the scientific market, are only able to deliver around one third of their maximum performance in double precision computations. Therefore, an additional performance gain can be achieved when the employed algorithms are numerically stable and the single precision floating point representation is sufficient.

External libraries

External libraries, for example for linear algebra, which are used in a CPU implementation cannot be easily integrated into a new GPU implementation. Some mathematical libraries, like cuBLAS [91], exist and can be used in GPU-ports of existing software.

11.1. Track Reconstruction on GPU Devices

The described characteristics of GPU hardware make it either impossible or hard and with degraded performance to translate CPU-designed reconstruction algorithms and implementations to accelerator hardware.

Existing algorithms must therefore be re-evaluated and potentially changed in terms of data and control flow to be adapted for GPU hardware.

The CPU-implementation of the current CMS track reconstruction is involving many different types of computations and features a complex control flow. Furthermore, a variety of input data is necessary for even the initial seeding step of the track reconstruction: detector layout, magnetic field and detector material information. These factors make the current CPU-bound implementation not feasible to be ported directly to GPU architectures.

This chapter will present a GPU-targeted implementation of the initial track seeding and discuss how complete tracks can be reconstructed to harness the compute capabilities of modern GPUs.

11.2. GPU and Accelerator Programming

Specialized programming libraries exist to access GPU and other hardware accelerators. The two most popular options are the CUDA platform, provided by NVIDIA, and the OpenCL standard, which is implemented by a variety of vendors.

Both platforms provide a similar set of components:

- **Custom compiler**

  In both OpenCL and CUDA, the program which runs on the GPU can be written in dialect of the C language and is translated by a custom compiler to the machine instructions of the actual GPU.

- **Host library**

  This library is loaded inside of the host program running on the CPU and is used to control the memory allocation, data transfer and program execution on the GPU.

- **GPU device driver**

  This component keeps track of all GPUs connected to the host machine and interfaces the GPUs to the host library.

While both OpenCL and CUDA offer a similar set of features, the most striking difference is the reach of these platforms.

The CUDA platform is a product of NVIDIA, and therefore only supports NVIDIA GPUs. This allows NVIDIA to integrate advanced features, like dynamic

memory allocation, on the hardware as well as on the software side at the same time.

The OpenCL platform is an open standard formulated by many industry leaders in the hardware and software domain, among them being Apple, Intel, AMD and NVIDIA. OpenCL's goal is to support the accelerator hardware of different vendors without changing the implementation of the program. This aspect of OpenCL is also named *portable performance*.

OpenCL-based programs cannot only run on the GPUs of NVIDIA, Intel and AMD, specialized hardware accelerators like the Intel Xeon Phi, but can also be executed on x86 CPUs without any changes. This is not possible with the CUDA SDK provided by NVIDIA.

In the context of CMS and grid computing, portability to different types of hardware is an important design constraint. If a capable accelerator device is available on a host machine, OpenCL-enabled applications can make use of them. In the fallback case, the CPU can be used to execute the OpenCL programs.

Another important advantage in using OpenCL is the possibility to easily switch or even support different GPU hardware vendors at the same time. The GPU market is a very dynamic one, with vendors like Intel, AMD and NVIDIA. By not relying on one specific vendor, possible price or performance advantages across vendors can be better utilized.

These two arguments let to the decision to use OpenCL as the basis for the track reconstruction implementation. In the following, an introduction to OpenCL and its most important concepts will be given.

11.3. Introduction to OpenCL

Influenced by NVIDIA's CUDA from the GPU side and by OpenMP, which originates from the classical CPU side, the open OpenCL standard is characterized by a formulation which is abstract enough to support both CPU and GPU computing resources. This is an ambitious goal, since providing an abstract interface together with the possibility to achieve peak performance is a challenging task. OpenCL employs a strict isolation of the computation work into fundamental units, the kernels. These kernels can be developed in the OpenCL C programming language, a subset of the C99 language, with some additional OpenCL specific keywords. In general, these kernels are hardware independent and compiled by the OpenCL runtime when they are loaded. To be able to fully exploit the parallel execution of the kernel code, multiple so-called work items are started to process a set of input values. The number of concurrently running work items is determined by the OpenCL system. How a concrete algorithm can be partitioned into work items has to be decided by the programmer.

11.3.1. Intel OpenCL SDK

As part of the OpenCL consortium, Intel played an important role in creating the OpenCL standard. The Intel OpenCL Software Development Kit (SDK) compiles OpenCL kernels in order to run them on x86-64 CPUs, Intel GPUs and the Intel Xeon Phi accelerator PCI board. Code vectorization is exploited to distribute the calculations of a kernel to the vector units. The most recent release of the SDK, supports the SSE and AVX vector instruction sets.

11.3.2. NVIDIA OpenCL

One of the forerunners of the OpenCL standard is the CUDA system for GPUs by NVIDIA. Since 2009, the NVIDIA graphics driver also supports the OpenCL standard and can compile and run OpenCL kernels on the graphics card.

11.4. clever: Simplified OpenCL Programming

The CMS reconstruction software has been developed and is maintained by a diverse group of physicists and computer scientists. Therefore it is necessary that the source code is accessible to all scientists involved, even if their programming skills and experience might differ. Furthermore, one algorithm might have been implemented by one person but will be modified by changing maintainers over the lifetime of the CMS experiment. For this reason, the amount of expert code, which can only be extended by personnel with specific knowledge, should be limited as much as possible.

The API provided by the OpenCL framework is based on the C programming language and explicit memory management is required. Furthermore, OpenCL kernels are not syntax checked during the compile time of the host application. They are loaded from text files and compiled during the startup of the host application. This makes it harder to spot syntax errors as early as possible.

The open-source `openclam`[1] library offers a convenient way to encapsulate OpenCL kernels inside of C++ classes. Using this method, the OpenCL C-based programming statements can be conveniently inserted in between regular c++ program lines. This allows for a syntax check of the OpenCL kernels during compile time of the host application and no separate text files need to be maintained to hold the kernel's source code. The development of the `openclam` library has ceased since some years and lacks some desired features like more elegant memory management. To improve this situation, the `clever` library has been developed as part of this thesis and builds on the idea of `openclam` to integrate the OpenCL kernels

[1] `http://code.google.com/p/openclam` (19.12.2014)

within the C++ source code and offers a convenient C++ wrapper for OpenCL's API calls.

The following section will given an overview of the functionalities offered by the `clever` library. The equivalent C++ source code of the OpenCL API has been omitted for the sake of compactness but can be found in any OpenCL programming introduction.

11.4.1. OpenCL Compute Context Creation

The C++ class `clever::context` is the main entry point of the `clever` library. If no parameters are supplied to this class, the first OpenCL runtime which is available on the current platform is selected. This object can also be created with a set of configuration parameters to select a specific OpenCL runtime or define whether a GPU or CPU should be used. Once the object of the class `clever::context` is destroyed, also all OpenCL resources which have been allocated via this context object is automatically released. This provides a programmer-friendly way to manage the lifetime of OpenCL resources.

Two examples of context creation can be seen in figures 11.1 and 11.2. In the first example, a context object is created without further specifying the platform and device type. The second example shows how a compute context can be requested on the NVIDIA platform. If the NVIDIA platform is not available on the current host, a C++ exception will be thrown.

```
// create compute context on default platform
clever::context context;
```

Figure 11.1.: Instantiation of an OpenCL Compute Context on the default Platform.

```
// create compute context on the NVIDIA platform
clever::context_settings cs ( clever::opencl::PlatformNameNVIDIA( ) );
clever::context context( cs );
```

Figure 11.2.: Instantiation of an OpenCL Compute Context on the NVIDIA Platform.

11.4.2. Memory Management and Data Transfer

In most modern programming languages, the lifetime of objects and the memory allocated to hold their state is either partially or fully handled by the compiler or runtime environment. In the C++ language, the memory regions of objects

which are allocated on the stack of a specific method are automatically released once the stack of the method is left. C++ objects which have been allocated on the heap must be manually allocated and de-allocated by the programmer, but the language offers convenient constructs to shield the concrete size of memory blocks from the programmer. Other languages, like Java, provide a garbage collection, where all allocated memory regions are automatically released once they are not in use any more.

In contrast, all memory buffers holding the input and output of computations performed with OpenCL must be explicitly allocated and de-allocated by the programmer. This can become especially cumbersome, when OpenCL memory buffers are used within larger applications, like CMSSW, and must be shared among many different modules.

To ease the memory management with OpenCL, the `clever` library includes C++ classes which take care of allocating OpenCL buffers of the correct size internally. The lifetime of the OpenCL buffer is directly coupled to the lifetime of the C++ object, which allows the programmer to apply the same rules he already knows from C++ resource management to control the lifetime of OpenCL memory buffers. Once the C++ object referencing the OpenCL buffer is destroyed, also the underlying memory region of the buffer is released.

Code listing 11.3 shows an example using the `clever` library's matrix type to allocate a 5-dimensional matrix with double precision floating point numbers and transfer a set of values to the OpenCL device.

11.4.3. Programming OpenCL Kernels

The `clever` library greatly simplifies the development of OpenCL kernel code. In the standard OpenCL framework offered by the vendors, OpenCL compute kernels are only syntax checked at the runtime of the host application. Building on the ideas of the `openclam` framework, a set of C++ macros are offered to the programmer to define the number of parameters passed to the kernel and the program code executed with the kernel.

The OpenCL C-language code within the kernel's function is syntax checked during the compile time of the host application. Code snippet 11.4 shows an example which defines an OpenCL kernel which takes two parameters as input.

The macro `KERNEL2_CLASS` will create a C++ class which can be instantiated by the programmer at the time the kernel code must be executed. At this time, the kernel code is compiled with the OpenCL framework and registered with the name `add_val`.
More complex kernels, taking more input parameters have been developed using this technique, but are not shown here for the sake of compactness.

```
clever::context context;

// define 5x5 matrix
typedef clever::matrix<double, 5> TestMatrix;

// don't set initial value but download it later on
TestMatrix m1( context );

// initialize the matrix entries
m1.fromArray( { 1,0,0,0,0
0,1,0,0,0
0,0,1,0,0
0,0,0,1,0
0,0,0,0,1 } );

// download values to device
m1.from_array(m1_in );
```

Figure 11.3.: Allocation of a custom matrix data type on an OpenCL compute context.

```
KERNEL2_CLASS( add_val, cl_mem, double ,
__kernel void add_val( __global double * a, const double b )
{
a[ get_global_id( 0 ) ] += b;
});
```

Figure 11.4.: Definition of a simple kernel with two parameters. The kernel only adds a constant value `b` to the values contained in the input array `a`.

11.4.4. Executing OpenCL Kernels

Once a C++ object holding the OpenCL kernel has been created, the kernel code is ready to be executed on the compute device. To pass all required input parameters to the kernel, the `run()` method is called. Parameters can either be scalar values or pointer to previously created OpenCL memory buffers. The code snippet 11.5 displays how one memory buffer (`m1`) and the scalar number 55 are passed to the kernel. The third parameter is passing the OpenCL work dimension, which defines over how many entries in the `m1` memory buffer the kernel will iterate.

For the track seeding implementation on GPU-devices presented in the following section, the `clever` library has been used to simplify memory management and to define and execute OpenCL kernels.

```
kernel.add_val.run(m1.get_mem(), 55, m1.range() );
```

Figure 11.5.: Calling an already defined kernel. `m1` is the matrix in memory, which has been defined before, 55 is the constant which is added to every element of the matrix and `m1.range()` passes the size of the matrix to the kernel execution.

11.5. Track Reconstruction with the Cellular Automaton Method

The cellular automaton (CA) track reconstruction method [92] uses a different approach to full track finding than the classical Kalman-method described in chapter 7. In the classical method, initial trajectory seeds (hit pairs or triplets) are generated at the innermost detector layer and subsequently propagated to the outer layers, with compatible hits being added to the tracks.

In the CA-method, triplets are not only generated for the innermost detector layer, but for all layers of the detector. Subsequently, compatible triplets on adjacent layers are joined to form a track candidate.

The triplet finding step of the CA-method is a localized operation: only hit data from three neighbouring detector layers enter the computation. Furthermore, triplets across the whole detector can be generated independently from each other, which allows for a large amount of parallelism in the algorithm.

Additionally, the CA implementation presented here will use cut-based criteria and parametrize the material effects on the track. This avoids using the Kalman-filter technique in the seeding step, which is known to be numerically unstable when using only single precision floating point representation [93]. As the presented implementation can therefore use single precision, the larger performance of GPUs when processing single precision floating point numbers can be harnessed.

These properties make the CA-method especially interesting to study as a GPU implementation.

The CA-method can be separated into four stages:

- **Hit pair generation**

  Finding two compatible hits on layer n and layer n-1 which might belong to the same track.

- **Triplet prediction**

  Using the information from the hit pair, compute the location where the third hit on layer n+1 is expected.

- **Triplet filtering**

  Generate triplets using the hit pairs and the compatible hits found on layer n+1. Apply quality cuts to decrease the fake rate of the generated triplets.

- **Triplet joining**

  Combine compatible triplets on adjacent layers to form longer track segments.

11.6. Hit Pairs pre-selection

Due to their high number, not all possible combinations of hits on two adjacent layers can be passed on to the downstream triplet generation. While two hits allow for only a limited determination of a tacks parameter, their location and the assumption that all tracks originate within a window around the collision point allow to reduce the possible combinations significantly.

Assuming a hit $h_2 = (\phi_2, z_2)$ in the *outer* layer, the search range for z in the inner layer can be limited to [94]

$$\left[r_1 \frac{r_2}{z_2 + z_{0,\max}} - z_{0,\max}, r_1 \frac{r_2}{z_2 - z_{0,\max}} + z_{0,\max} \right] \tag{11.1}$$

where r_1 and r_2 are the radii of the detector layers containing h_1 and h_2, respectively. $z_{0,\max}$ is the size of the acceptance window in z direction.

Also the change in the ϕ position between the two hits can be evaluated for their compatibility with a track originating at the center of the detector. The tracks of charged particles are curved in the magnetic field of CMS and they travel along a helix with radius r, which is related to the transverse momentum p_T of the track. A minimum radius $r_{\min}$ can be defined, for the minimum curvature of tracks which are accepted for the hit pair building.

Furthermore, the maximum transverse impact parameter $d_{0,\max}$ is introduced to allow for tracks which do not originate exactly at the origin in the transverse plane.

Using the minimum radius $r_{\min}$, the accepted $\Delta\phi$ can then be limited to [94]

$$|\Delta\phi| \leq \left| \arccos\left(\frac{r_2}{2r_{\min}}\right) - \arccos\left(\frac{r_1}{2r_{\min}}\right) \right| + \arctan\left(\frac{d_{0,\max}(r_2 - r_1)}{r_1 r_2}\right) \tag{11.2}$$

11.7. Triplet Prediction

To limit the number of third hits combined with the generated hit pair, a prediction of the possible positions of the third hit is performed. In a similar fashion

to the hit pair pre-selection, equations 11.1 and 11.2 are used to compute the accepted region of the third hit. The origin of the coordinate system is moved to the position of the first hit of the hit pair, and the second hit of the hit pair is used to extrapolate to the possible search window for the third hit. In this computation $d_{0,\max}$ needs not be considered as a possible transverse displacement of the track has already been handled by defining the first hit as the origin of the new coordinate system.

All hits in the third layer, which are in the predicted search window, are selected as triplet candidates.

11.8. Triplet Filtering Criteria

Additional filter criteria on the triplet candidates are applied to reject fake combination of hits.

Cut on $d\theta$

The differences in θ between the lower half (hit one and two) and second half (hit two and three) is expected to be zero. The magnetic field in CMS is only affecting the particle trajectory in the transverse plane. The particle propagation in the y-z plane can be assumed to be a straight line. Multiple scattering effects and measurement uncertainties still affect the recorded position of the hit. Therefore, a narrow window of acceptance must be available.

$$\left|\frac{\theta'}{\theta} - 1\right| \leq d\theta \tag{11.3}$$

wherein θ is of the lower half and θ' of the upper half of the triplet and $d\theta$ is the accepted difference between the two. Figure 11.6 illustrates this cut.

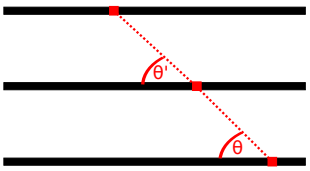


Figure 11.6.: Definition of θ and θ' of a triplet in the z-y plane.

Cut on $d\phi$

Similar to $d\theta$, an additional cut on dϕ can be applied. Contrary to θ, the angle ϕ might change due to the curvature of the track in the magnetic field. This feature must be considered when determining the cut value $d\phi$.

$$|\phi' - \phi| \leq d\phi \tag{11.4}$$

Figure 11.7 displays a slightly curved trajectory in the x-y plane and the associated ϕ and ϕ' values.

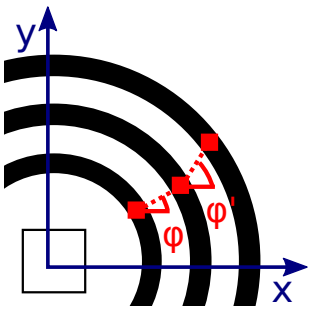


Figure 11.7.: Definition of ϕ and ϕ' of one triplet in the x-y plane.

Cut on the transverse impact parameter

The previous two filter criteria ensure the triplets to be internally consistent by preventing kinks in the θ and ϕ angles. Another criteria to improve the quality of final triplets is to require tracks to originate in the vicinity of the interaction point. To determine this transverse impact parameter - the closest approach of a track to the detector center - a fit with the three hits included in the triplet is performed.

In the transverse plane, the trajectory of the charged particles can be approximated as a circle which is fully defined by the three triplet hits. The Riemann method [95] maps points on circles on to planes and performs a fit in this transformed space. Its performance and runtime have been compared to other fitting techniques for the circle fit use case and found to be up to 40 times faster [95].

Additionally, the program code to implement the Riemann fit is compact and contains no recursion or complex control flow. These features make this method especially suited to be implemented on GPUs.

Once the circle passing through all three hit positions has been computed, the point of closest approach d_0 to the origin can be easily determined. This allows to apply a selection with $d_{0,\max}$ being the maximum accepted distance from the detector center.

$$d_0 \leq d_{0,\max} \tag{11.5}$$

Figure 11.8 displays three hits assigned to a triplet in the transverse plane. Also visible is the fitted circle and its transverse impact parameter.

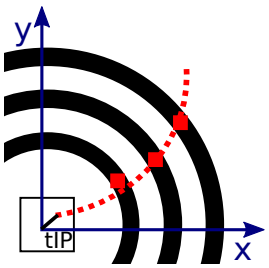


Figure 11.8.: The transverse impact parameter of a triplet after all three hits have been fitted with the Riemann method.

11.9. Determination of Triplet Filter Cuts

To determine the cut values, a dataset with simulated top pair decays was used. During the of decay the top quarks, many secondary particles are generated and are concentrated in specific regions of the detector. This high track density is a difficult environment for reconstruction algorithms and allows meaningful estimates of an algorithm's fake rate.

Cut values have been derived for triplets located on various combinations of tracker layers. These cuts have been balanced to achieve a reconstruction efficiency of 80%.

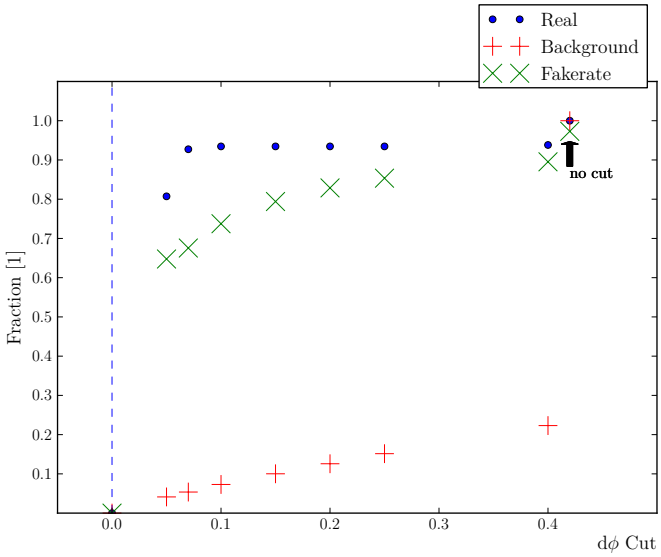


Figure 11.9.: The fraction of real, fake and background triplets which pass the $d\phi$ cut plotted over a range of the cut quantity. For the plot point "no cut", none of the cuts are applied and all possible triplet combinations are in the final selection.

Figure 11.9 shows the triplet finding efficiency for various values of the cut quantity $d\phi$ for triplets on the first three detector layers in the barrel region. The background contribution is also plotted which is the percentage of wrong combinations passing the cut criteria divided by the overall amount of wrong combinations. The fake rate is the percentage of wrong combinations divided by the overall number of combinations passing the cuts. These definitions are also listed in equations 11.6.

$$Eff = \frac{N_{passed}}{N_{gen}} \qquad Fake = \frac{N_{fakes\ passed}}{N_{passed}} \qquad Bkg = \frac{N_{fakes\ found}}{N_{fakes}} \tag{11.6}$$

For the study presented in figure 11.9 only the $d\phi$ cut was applied to the input set of possible triplet combinations. In the actual implementation described below, the three cuts are applied sequentially.

The cut value evaluations of $d\theta$ and $d_{0,\max}$ for the first three detector layers in the barrel region can be found in figures B.1 and B.2 in appendix B.

Figure 11.10 shows the efficiency, fake rate and background contribution when all three triplet filter cuts are applied sequentially on a set of input triplet candidates from the first three layers in the barrel tracking detector.

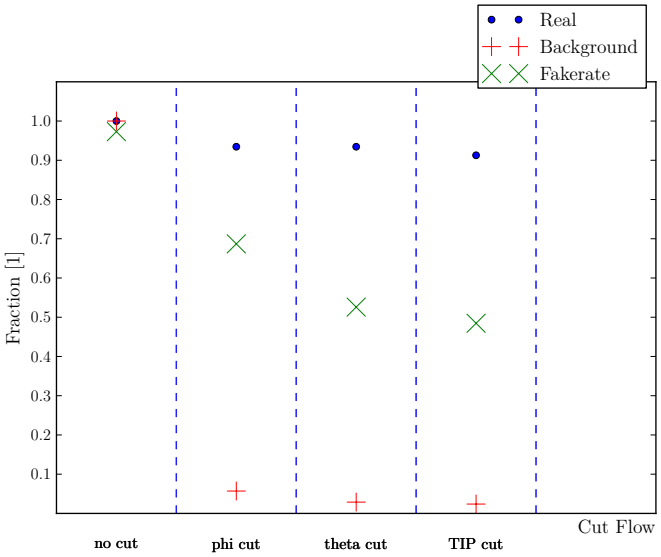


Figure 11.10.: The fraction of real, fake and background triplets if all three triplet filter criteria are applied sequentially.

After all three cuts have been applied, the triplet finding effiency is at 92% with a fake rate of 49%.

In the same manner, optimal cut values have been determined for the other layer combinations which are used for triplet seeding. Table 11.1 lists the final

Triplet Layers	$d\phi$	$d\theta$	$d_{0,\max}$
1-2-3	0.075	0.01	0.5
2-3-4	0.2	0.2	0.5
3-4-5	0.4	0.4	1.0
4-5-8	0.4	0.5	3.5

Table 11.1.: Cut quantities for the $d\phi$, $d\theta$ and $d_{0,\max}$ triplet filter for all studied layer combinations.

values of the three quality cuts for the studied layer combinations. The more the studied layer combinations are located in outer regions of the detector, the more the cuts needed to be relaxed to achieve the targeted triplet efficiency of 80%.

This has three primary reasons:

- The measurement uncertainty of the strip silicon detectors, starting at layer four, is coarser than the pixel silicon detectors of the first three layers
- The traveled distance between the detector layers is bigger, thus increasing the effect of track bending
- The distance to extrapolate the trajectory back to the detector center is larger, thus the computed TIP becomes less precise.

11.10. Investigation on Triplet Joining criteria

Once triplets on all layers have been generated, they can be combined to form longer segments of tracks. In the following, possible combination criteria for triplets from layers 1-2-3 and layers 2-3-4 will be studied.

To this extent, the three hits associated to a triplet haven been fitted with the Kalman-method of the CMS reconstruction. Due to the smoothing procedure described in chapter 7.5, the best-estimate of the track state is available at each of the three hits. This allows to check the triplets for compatible direction and momentum at each hit they share.

In the following, the three combination criteria, which have been studied, will be discussed:

Shared hits between triplets

When searching for triplets in neighbouring detector layers, triplets which contain hits from the same track must share two hits on the two inner layers.

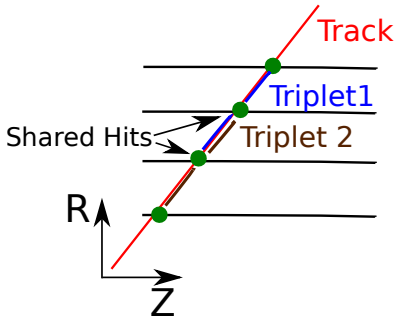


Figure 11.11.: Illustration of two triples on neighbouring detector layers sharing two hits. If both triplets have been generated with the hits from the same track, as shown here, they must have the two innermost hits in common.

Figure 11.11 illustrates two triplets, which belong to the same track, sharing their inner hits. To determine the hits shared between triplets, no explicit computation is necessary. Only the identification numbers of hits included in the triplets must be compared, which is a very fast operation on CPU and GPU. Only triplet combinations with two inner hits shared are considered for further processing. This already reduces the amount of input data passed to the more compute intensive steps which follow.

Charge and momentum compatibility

Due to the Kalman fit of the triplet hits, the charge and momentum of the track is available at the position of each hit. For both shared hits, the charge divided by the magnitude of the momentum must be similar. Formula 11.7 is used to cut on this quantity by ensuring the absolute difference between the values of both triplets is smaller than dp. The charges q and q' and the momenta $\vec{p}$ and $\vec{p}'$ are of the first and second triplet, respectively.

$$\left| \frac{q}{|\vec{p}|} - \frac{q'}{|\vec{p}'|} \right| \leq dp \tag{11.7}$$

Direction compatibility

For both shared hits, the direction of the track is expected to be similar, thus having no kink. Formula 11.7 is used to express this requirement by computing the magnitude of the difference between the normal vectors $\vec{n}$ and $\vec{n}'$ of the track direction for each triplet. Only triplet combinations for which this quantity is smaller then dx pass this cut and are considered a valid combination of two triplets.

$$|\vec{n} - \vec{n}'| \leq dx \tag{11.8}$$

11.11. Applying the Triplet Joining criteria

The optimal cut values of the triplet combination criteria described in the previous section have been studied with the requirement to achieve a 90% combination efficiency. The same top pair decay dataset as in the triplet filter studies was also used as the test sample. As input to the triplet combination algorithm, only triplets passing the filter criteria defined in section 11.9 are considered.

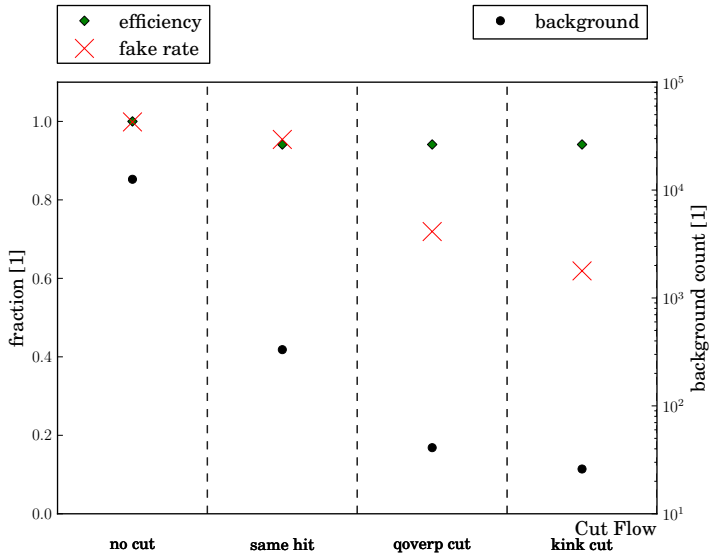


Figure 11.12.: Triplet combination efficiency and fake rate for joining triplets from the first three layers 1-2-3 with triplets from the layers 2-3-4. The background count, being the absolute number of fake combinations still in the final selection, is also plotted.

Figure 11.12 shows the triplet combination efficiency and fake rate when joining triplets from the first three layers 1-2-3 with triplets from the layers 2-3-4. The background count, being the absolute number of fake combinations still in the final selection is also plotted. Requiring the same shared hits in the triplets, already reduces the background by a factor of two. Applying the two following, more compute intensive cuts, decreases the fake rate further while achieving a combination efficiency of 90%.

$$Eff = \frac{N_{\text{correct combinations}}}{N_{\text{tracks}}} \qquad Fake = \frac{N_{\text{wrong combinations}}}{N_{\text{found combinations}}} \tag{11.9}$$

11.12. Physics results of the OpenCL Implementation

As part of this thesis, the fast triplet finding algorithm described in the previous sections has been implemented using the OpenCL platform and the `clever` library.

For the following study, two computing platforms will be used to run the OpenCL implementation of the algorithm:

- **Classical CPU** - Intel Core i7-3930K

  This high-end consumer CPU has 6 parallel processing cores which run at 3.2 GHz. In the year 2014, this CPU costs around € 500 and provides a peak floating point performance 154 GFlops.

  The operating system Scientific Linux version 6.4 and the Intel OpenCL SDK 2012 was used to run programs on this hardware.

- **Consumer GPU** - NVIDIA GeForce GTX 660

  This GPU is part of a NVIDIA consumer graphics card, costs € 200 in the year 2014 and offers a peak floating point performance of 1880 GFlops.

  The NVIDIA driver version 319.23 was used to provide the OpenCL platform to the programs using this hardware.

The same CPU was also used to execute the default CMS triplet finding algorithm, which will be used as a runtime reference. This algorithm is implemented in C++ and runs inside of the CMSSW reconstruction framework. Both CMSSW and the host-side of the OpenCL implementation have been compiled with `GCC` version 4.7.2.

Figure 11.13 shows the triplet finding efficiency and fake rate of the algorithm running on the NVIDA GPU. The clone rate is the fraction of triplets which have been found twice for the same layer combination. This can happen as the individual detector elements holding the active silicon surface overlap slightly within one layer. Therefor, tracks passing in this overlap region, two hits will be produced in one detector layer. This ambiguity cannot be resolved at the time of triplet finding but has to be addressed at a later processing stage.

The plot contains entries for each of the studied layer combinations. For combinations incorporating hits from strip layers, a higher fake rate can be observed. This is due to the previously discussed effects of larger measurement error and spacing between these layers. The targeted reconstruction efficiency 80% is achieved for all studied layer combinations.

To investigate the triplet finding performance for a varying occupancy of the tracker, a Monte Carlo generated set of muon tracks has been used. Increasing numbers of tracks have been artificially inserted into one event and handed to the OpenCL triplet finding.

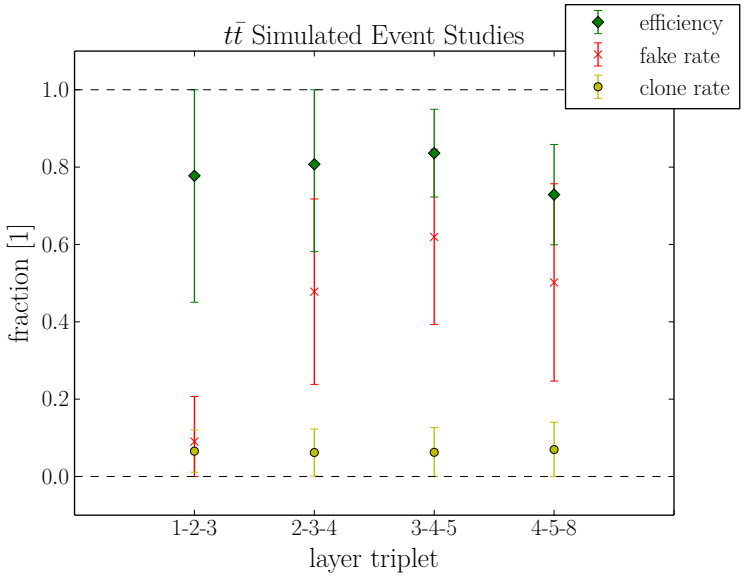


Figure 11.13.: Triplet finding efficiency, fake rate and clone rate of the OpenCL implementation running on the NVIDA GPU.

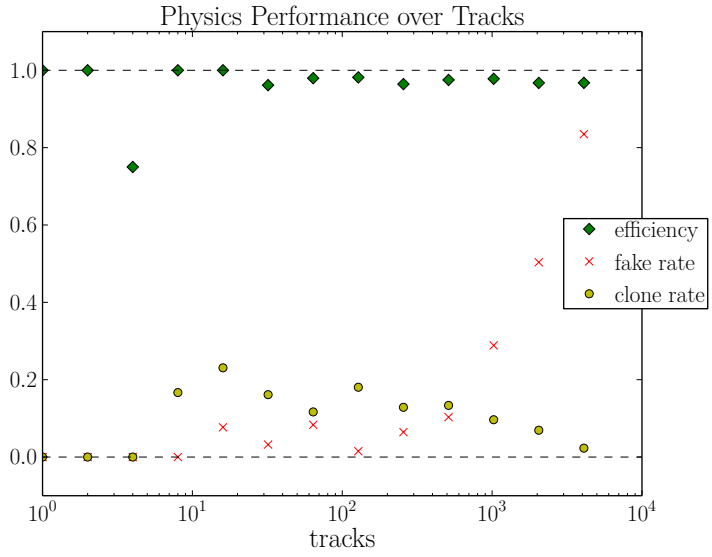


Figure 11.14.: Efficiency, fake rate and clone rate for the triplet finding in the first three layers of the detector for an event with one track up to an event with 4000 tracks.

Figure 11.14 displays the efficiency, fake rate and clone rate for the triplet finding in the first three layers of the detector for an event with one track up to an event with 4000 tracks. The efficiency is stable over the whole range of studied tracker occupancy, except for the dip at three tracks. In this case, one of the three generated tracks is not found and thus affecting the efficiency heavily. As visible from the other efficiency entries, this can be considered to be an outlier.

Also, the fake rate is stable and below 30% for up to one thousand tracks per event. A sharp increase in fake rate can be observed for even higher occupancies. One way to counter the increase in fake rate in these high-occupancy regimes is to tighten the cuts on triplet candidates and use multiple triplet finding rounds to recover the efficiency losses, very similar to the iterative tracking employed by the default CMS reconstruction.

The triplet finding algorithm, the cut configuration and the OpenCL implementation running on a GPU have proven to be able to achieve a satisfying reconstruction performance. This demonstrates the feasibility of the proposed approach and shows for the first time, that triplet finding can be performed on CMS input hits on a GPU device.

11.13. Runtime Results of the OpenCL Implementation

In this section, the runtime behaviour of the OpenCL implementation and the default CMS triplet finding will be compared. For the OpenCL-based triplet finding, two hardware backends will also be studied: CPU and GPU. After demonstrating that a good physics performance can be achieved with GPU-based code, the next important step is to study whether significant gains in runtime can be achieved.

To arrive at a detailed understanding of the runtime performance of various implementations, artificially generated events with a varying number of muon tracks are used.

Figure 11.15 shows the runtime per event for the OpenCL implementation and the runtime of the CMSSW triplet finding as reference. For OpenCL, two distinct numbers are plotted: The *wall time* reports the time period from starting the computation on the host device until the result is available on the host for further processing. Therefore, the *wall time* includes the data transfer to the compute device, the execution of the kernel, and the time to transfer the result data back to the host device's memory space. In contrast, the *kernel time* only measures the time spent in the actual OpenCL kernels performing the triplet finding.

The CMSSW triplet finding runs solely on the CPU, therefore having no need for data transfers, only the overall runtime of the computation is plotted. The CMSSW runtime scales exponentially with a growing number of input tracks, while no pedestal in runtime can be observed for only a few tracks.

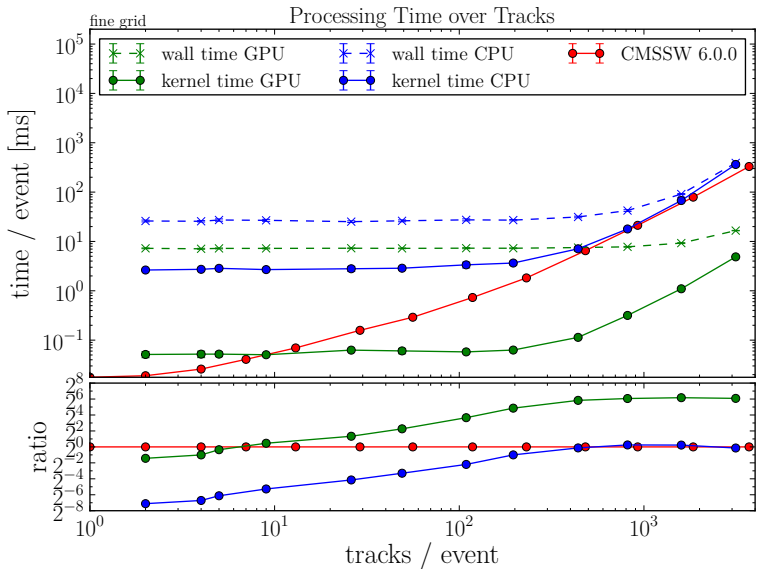


Figure 11.15.: Runtime per event for the OpenCL implementation and the runtime of the CMS triplet finding as reference. The *wall time* is reporting the complete processing time, including the OpenCL kernel startup and data transfer, while the *kernel time* measures only the runtime of the actual compute kernel.

Both OpenCL platforms, GPU and CPU, behave differently here. For both, a clear pedestal in runtime is observed. This constant overhead is expected, as it is not economical to send very small events, therefore few computations, to the GPU and back.

The more important finding is when, or if at all, this constant overhead can be amortized by a increasing amount of computations on the GPU. Considering only the actual runtime of the OpenCL kernel on the GPU, this break-even point is already reached with 10 track per event. This is for a best-case scenario, where the results do not need to be copied back to the host memory and further processing can be performed on the GPU.

In fact, for the OpenCL implementation presented here, the data needs be moved back to the host side. Including all data transfer and GPU execution overhead, the GPU implementation outperforms the CPU for around 400 tracks per event.

For the OpenCL-code running on the CPU, a similar constant overhead can be observed. It is not able to achieve, or surpass, the CMSSW implementation when comparing the *wall time*. In fact, only for events with more than 2000 tracks is the OpenCL CPU version able to achieve a similar performance than CMSSW.

However, for the overall runtime of the GPU implementation a steady gain compared to the CPU implementations is visible, once the break even point has

been reached. For events with many tracks, a speedup over the CMSSW implementation of up to a factor of 64 can be observed.

The GPU implementation presented here is able to perform a ten times faster triplet finding for events with 1000 tracks, which is a realistic number of tracks per event for the upcoming run periods of the LHC.

One way to still amortize for the GPU-induced overhead in less-populated events is to bundle many events and send them in one batch to the GPU, process them and return the result.

11.14. Conclusion and Further Steps

The specific challenges in porting HEP reconstruction algorithms to GPU hardware have been outlined and the challenges faced by the programmer were iterated. Key features of the `clever` library, created as part of this thesis work, were explained and it was shown how `clever` can be used by non-GPU programming experts to express data structures, data transfers and algorithms.

Triplet finding and joining algorithms were presented, which have been tailored to work well on GPU hardware. Cut values to be used by the these algorithms have been determined by a representative top pair decay dataset. Triplets, which combine hits of the barrel tracker layers 1 to 5 and 8, can be reconstructed with an efficiency of 80% and an acceptable fake rate is achieved. In the second stage, triplets from the inner barrel part of the tracker can be combined with an efficiency of 90%.

The triplet finding algorithm was implemented using OpenCL and the `clever` library. The physics performance expected from the cut studies is achieved and the reconstruction effiency is stable, even for events with more than 1000 tracks.

For events containing 400 and more tracks, the GPU implementation shows clear runtime improvements with respect to the current, CPU only implementation. For the first time in the CMS experiment, part of the event reconstruction has been ported to a GPU target and the physics performance was measured with simulated events.

Using the CPU as a platform for OpenCL does not give any benefit. For small events with limited complexity, this option is even significantly slower than the C++ implementation of the CMSSW triplet finding algorithm. Therefore, at the present time, running OpenCL on CPUs should only be considered as a fallback option if no GPU is available on the host machine.

The feasibility of a GPU-based reconstruction has been demonstrated. Simplifications of the data structures, a reduction of the geometry detail and tailored algorithms were necessary to adapt to the GPU. With careful consideration, reconstruction efficiencies comparable with the current, CPU-based reconstruction can be achieved.

To enable the full track reconstruction on a GPU target, the triplet joining algorithm needs to be implemented. The largest work-item here is to implement the Kalman-filter. The ATLAS experiment has already shown the feasibility of such an implementation on GPUs [96].

Furthermore the integration of a GPU-based track reconstruction into the overall workflow of the CMSSW framework needs to be studied.

Chapter 12

Improving Track Fitting with a detailed Material Model

As described in chapter 7.5, a detailed knowledge of the material effects encountered by a particle traversing the detector is necessary to perform a fit of the particle trajectory.

The CMS track reconstruction employs a material parametrization to estimate the material effect on a particle trajectory. Therein, for each silicon detector surface in the tracker, material constants are stored which represent the material density around the location of each detector element. In this method, the three dimensional passive and active material volumes of the tracker material are condensed onto the two dimensional detector surfaces.

Two parametrized scalar values are available for each detector surface. The inverse radiation length $\frac{1}{X_0}$ and the material dependant part of the Bethe formula X_i. These constants have been determined for all detector surfaces in a special procedure.

Therein, a Monte Carlo generated sample of neutrinos originating from the detector center is used. The neutrinos are uniformly distributed in η and ϕ and cross the detector material with practically no interactions. Still, the knowledge about the Geant4 material volumes and detector surfaces they crossed can be used to assign material to specific detector surfaces and compute the two material constants.

Once this procedure has been performed for a given detector geometry, the derived material constants can be used to estimate the energy loss and multiple scattering effects on passing tracks in all parts of the tracker.

To compute the energy loss and multiple scattering effects on a track going from detector surface N to surface $N + 1$, the material constant is looked up for the detector surface N. Going from surface $N + 1$ to surface $N + 2$, the material constant from $N + 1$ is used and so on.

12.1. Limitations of the Current Material Parametrization

Although the parametrization of the tracker material allows for a fast and reliable estimation of the material effect, there are some simplifications and short-comings associated with this method.

As the material constants are linked to the detector surfaces, the actual path of a specific particle from surface N to $N+1$ is not taken into consideration. While this is a valid approximation if the material between layers is equally distributed, it can be an approximation which is too grain in areas with diverse material types and densities.

Figure 12.1 illustrates this limitation with two tracks originating and arriving at the same surface but traversing very different material densities. In the parametric approach, the material effects would be estimated by using an average of both material densities.

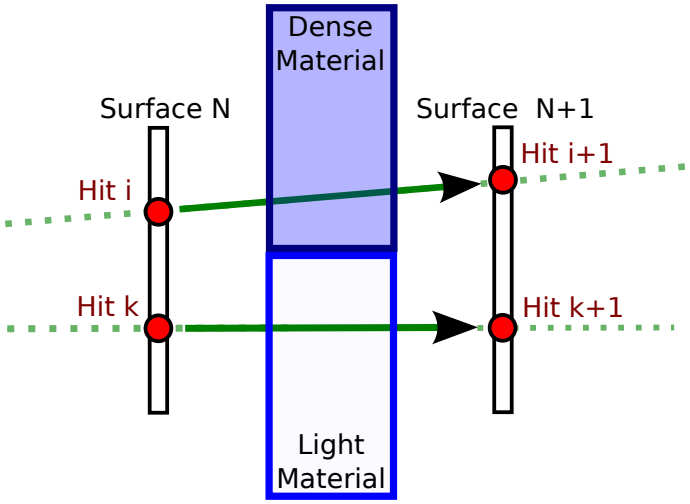


Figure 12.1.: Two particle tracks with hits on the same detector surfaces will use the same material constants in the parametric approximation. In this depiction, they cross very different material densities. The parametric energy loss approximation will use the mean material density of the transition from surface N to $N+1$ in this case.

Furthermore, the parametrization assumes that particle tracks originate from the center of the detector. This is true for most tracks, but not for tracks originating from particle decays in secondary vertices. These tracks can originate multiple centimeters away from the detector center. Many of these signals are important in the search for possible long-lived SUSY particle candidates or the reconstruction of particle masses, as shown below for the decay of a short-lived kaon.

An additional effect, which is not considered in the parametrized approximation, is the track curvature due to the magnetic field covering the whole detector volume. As visible in figure 12.2, positive or negative charged tracks will cross different tracker material, depending on their curvature, even though they start at exactly the same point on the detector surface. As a second order effect, the energy loss experienced by a particle along its trajectory will also alter the curvature between two detector surfaces.

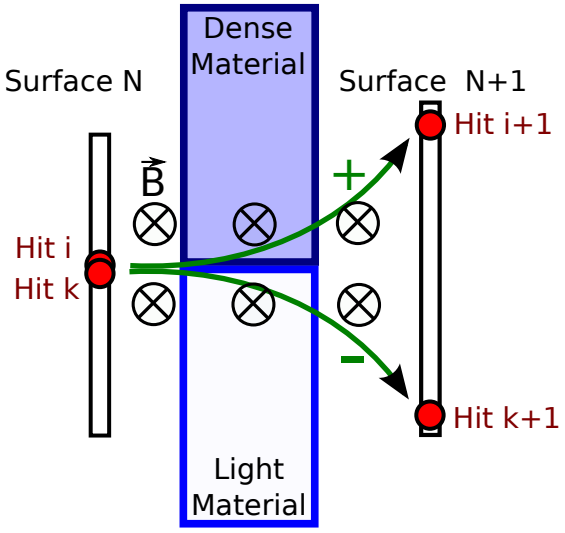


Figure 12.2.: Although originating at the same point, the trajectory of positive and negative charged tracks can differ significantly due to the magnetic field B. This implies that they might cross different material densities when propagating from detector surface N to $N+1$.

A study of CMS has been performed in the year 2010 to determine the momentum scale and resolution of the CMS tracker and reconstruction software [97]. Among other reference signals, the decay $K_s^0 \to \pi^+\pi^-$ was used to reconstruct the mass of short-lived kaons and compare it to the well-known reference measurements. In figure 12.3, the reconstructed particle mass in measured collisions is always lower than predicted by Monte Carlo and the difference between the reconstructed and reference K_s^0 mass increases with larger pseudo-rapidity. The study notes that the used material effects modelling within the CMS tracker might be a possible explanation for the biases observed in this mass reconstruction.

To investigate this possibility, a fine-grained material effect estimation will be implemented and evaluated in this and the following chapter.

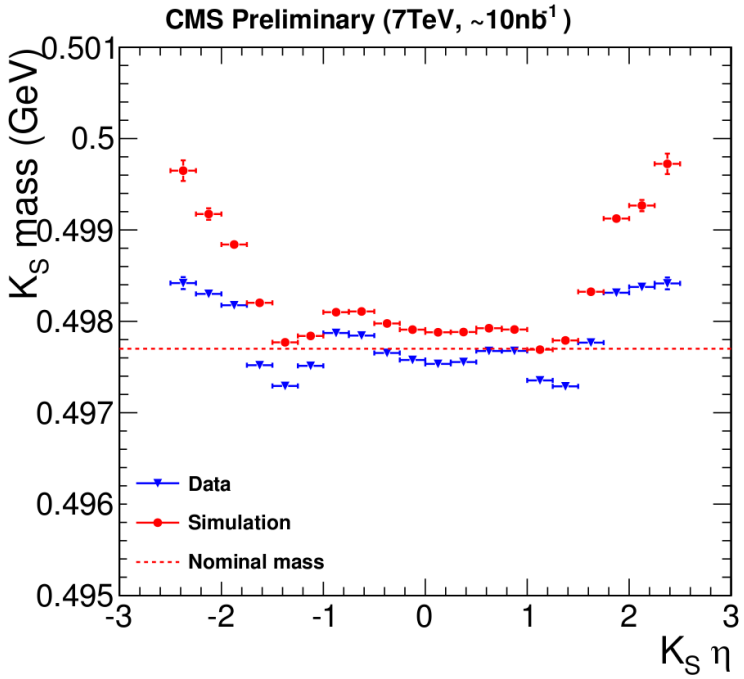


Figure 12.3.: Dependence of the K_s^0 mass on η of the reconstructed kaon. The data from measured events was recorded with the CMS detector in the year 2012 at a center-of-mass energy of 7 TeV . A relation between the detector region where the K_s^0 was reconstructed and the mass is visible, both in data and simulation. This effect is especially pronounced in the endcap region [97].

12.2. Geant4 Error Propagation Package

The Geant4 error propagation package [98], short Geant4e, is part of the Geant4 material simulation software suite and is specifically tailored to the requirements of track reconstruction.

Geant4e takes the track state and associated error of a particle in the detector as an input and propagates this state and errors to any target surface. During this procedure, interaction with the tracker material and the magnetic field are taken into account.

12.3. Energy Loss Validation for the CMS Tracker

To understand the quality of the energy loss approximation, both for the current material parametrization and the Geant4e method, the predicted energy loss can be compared to the simulation truth. To this end, Monte Carlo-generated particle tracks of muons with a momentum of 1 GeV originating from the detector center are used to compare the mean energy loss of both methods to the simulation truth.

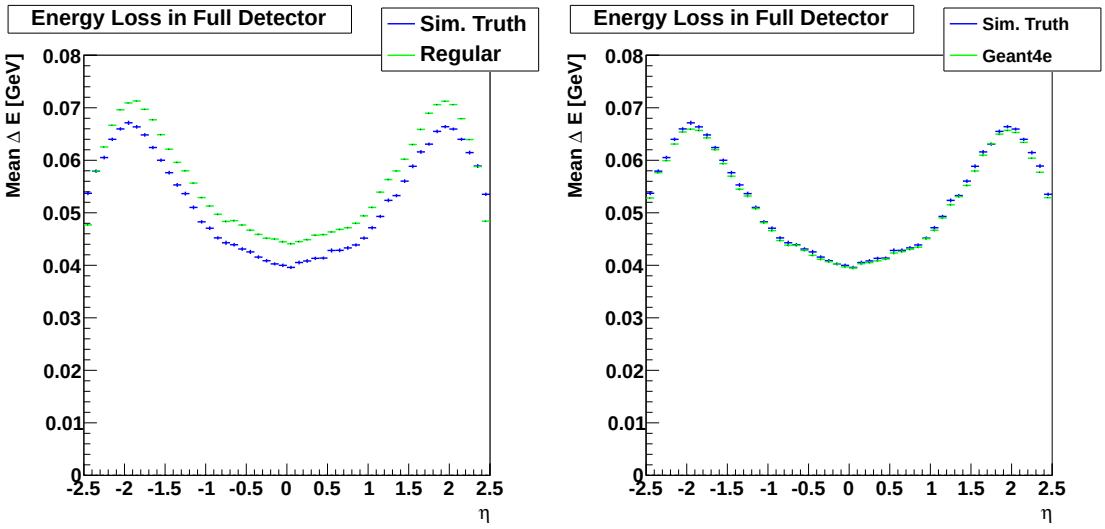
This comparison can be performed for single propagations when crossing dense regions of the tracker. Furthermore, the energy loss of multiple propagations along one track can be summed up and the mean energy loss for many tracks in this region can be computed.

Figures 12.4a and figure 12.4b show the mean energy loss computed by the regular parametrization method and Geant4e over the whole η range of the tracker. The reference mean energy loss derived from the simulation truth is also displayed in both plots. The mean energy loss estimated by the parametrization method shows a systematic shift to a higher energy loss for all areas of eta compared to the reference mean energy loss. When using Geant4e to approximate the mean energy loss, the result is closer to the simulation truth. The maximum difference is of the order of 2% in the region $\eta = -2$.

Figure 12.5a and figure 12.5b show the approximation for one selected particle transition, passing from the TID to the TEC sub-detector. This region was picked because the amount of material to traverse by the particles is especially high in this region, thus affecting the particle trajectory. Due to the geometrical properties of the CMS tracker, which can be seen in figure 5.4, the transition of particles from TID to TEC can only occur at a certain band in η, therefore these plots are showing entries only for the outer η regions.

As visible in figure 12.5a the parametric approach underestimates the mean energy loss in certain regions of η by more than a factor of 2. The parametric method is only able to approximate the mean energy loss in the region around $|\eta| = 1.7$ sufficiently well.

In contrast, figure 12.5b shows that the Geant4e method is able to provide a



(a) Mean energy loss approximated by the parametrization method and the simulation truth.

(b) Mean energy loss approximated by the Geant4e method and the simulation truth.

Figure 12.4.: Comparison plots of the parametrization and Geant4e energy loss computations for the whole detector.

good approximation of the mean energy loss over the full η range possible for this transition.

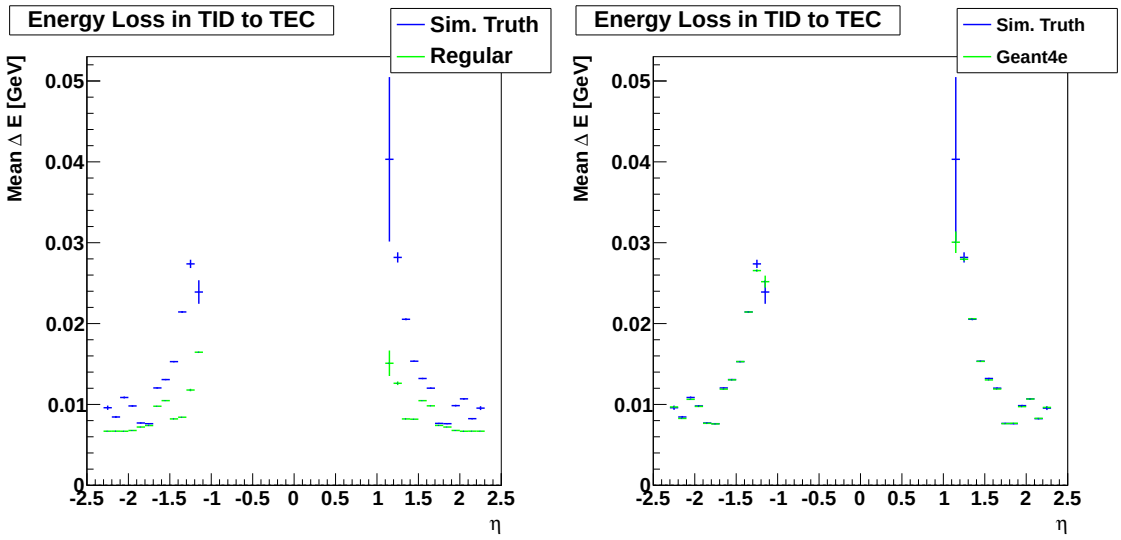
The approximation of the energy loss of individual particles can be validated further by studying the actual energy loss distribution of one individual bin in η. Figure 12.6a and figure 12.6b display the energy loss distribution for the bin $-1.8 \leq \eta < -1.7$ in the transition from the TID to the TEC for the parametric and the Geant4e method, respectively.

A rational function approximating the landau distribution has been fitted to the histogram of the simulation energy loss using the ROOT package [43] and is overlayed on the histogram. While both the parametric and Geant4e method show some discrepancy to the ideal Landau distribution of the energy loss, the Geant4e method performs better.

The backside of the parametric approach becomes especially visible here, as the material values are discretized on a per-detector level, the computed energy losses are concentrated on a few values and the proper shape of the Landau distribution is not approximated properly.

The mean energy loss is not properly modeled by the parametric approach for this complex transition region of the detector. Although the Geant4e method does not model the Landau tail of the energy loss distribution properly, the mean energy loss is in good agreement to the values derived from simulation.

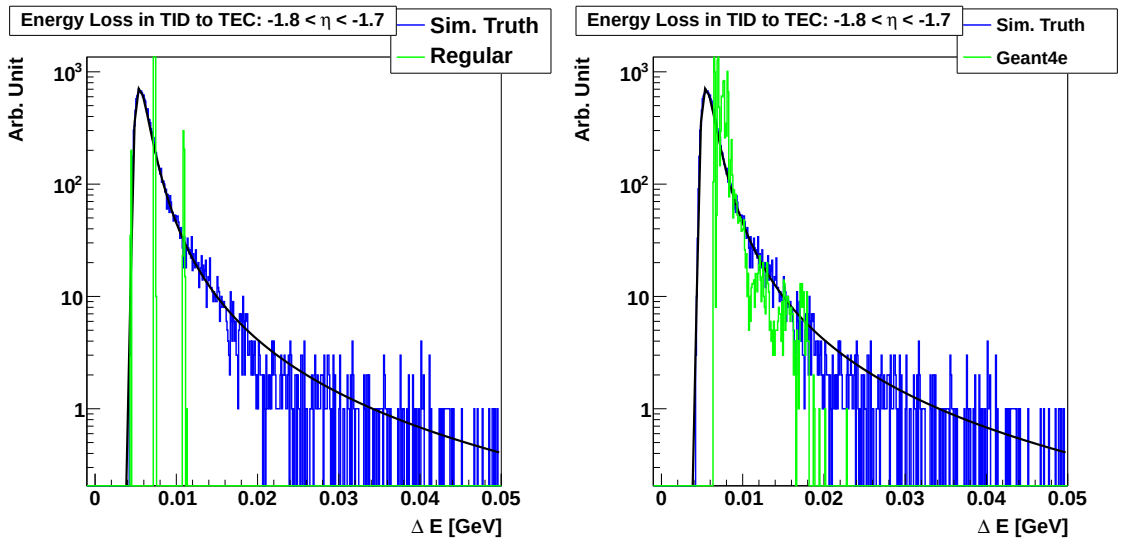
Comparisons for all other regions and critical transitions in the CMS tracker



(a) Mean energy loss approximated by the regular parametric method and the simulation truth.

(b) Mean energy loss approximated by the Geant4e method and the simulation truth.

Figure 12.5.: Comparison plots of the parametrization and Geant4e energy loss computations for the transition from the TID to the TEC sub-detector.



(a) Distribution of the energy loss estimated by the parametric method.

(b) Distribution of the energy loss estimated by the Geant4e method.

Figure 12.6.: Distribution of the energy loss estimated by the parametric and Geant4e methods of the transition from the TID to the TEC tracker elements. The content of these histograms are constrained to the bin $-1.8 \leq \eta < -1.7$ and the energy loss distribution have been fitted with a function approximating a Landau distribution.

have been performed and can be found in appendix C. The parametric approach is able to approximate the mean energy loss over the whole detector fairly well, as visible in figure 12.4a. In the complex transition region however, the parametric method shows significant deficiencies. The Geant4e package is able to achieve a good approximation of the mean energy loss in all parts of the CMS tracker.

12.4. Incorporation of Geant4e in the CMS Reconstruction

The CMS reconstruction framework uses abstractions for all functional components involved in the fitting process. Abstract C++ classes define the interface of a component and can be inherited to provide a concrete implementation of one part of the fitting functionalities. Figure 12.7 shows an overview of all components involved in the trajectory fit.

- **KFFittingSmoother**

  Combines the forward and backward fit of the trajectory to the final result
  - **KFTrajectoryFitter**

    Performs the forward fit of the track
    - **Propagator**

      Propagates track states from hit i to hit $i+1$
    - **KFUpdator**

      Updates the track state with the current hit i using the Kalman filter method
  - **KFTrajectorySmoother**

    Performs the backward fit of the track
    - **Propagator**

      Propagates track states from hit $i-1$ to hit i
    - **KFUpdator**

      Updates the track state with the current hit i using the Kalman filter method

Figure 12.7.: Software components involved in the track fit of the CMS reconstruction.

The Propagator interface was used to integrate the Geant4e functionality into

the trajectory fitting workflow of CMS. Propagator components take a trajectory state and a target surface as input values and are responsible for propagating the state and associated errors to the target surface. The trajectory state and the associated error matrix is returned to the calling program code. If the target surface cannot be reached, for example due to curling trajectories in the magnetic field, an invalid trajectory is returned.

Propagators support two modes of operation: forward and backward propagation. During forward propagation, the particle is propagated along its trajectory, starting from hit n and arriving at the next hit $n + 1$. In contrast, during backward propagation mode, the trajectory of the particle is followed in the opposite direction. Here, the propagation starts at hit n and arrives at the previous hit $n - 1$. Even if the backward mode does not reflect the travel direction of the particle, it is needed for the smoothing step of the Kalman filter.

After the backward propagation, the particle has gained energy along its trajectory. Furthermore, the particle charge has to be inverted so the influence of the magnetic field on the trajectory is consistent between forward and backward propagation. Both propagation modes are used during the trajectory fit.

Multiple Propagator implementations exist in the CMSSW code base. Some are intended for fast execution and assume the particle travels on a perfect helix, while others provide a more fine-grained approximation of the magnetic field and material effects.

For the default track fitting of CMS, a Runge-Kutta based method is used to compute the propagation of the particle in the magnetic field and the aforementioned parametric method is employed to compute the material effects, namely multiple scattering and energy loss, on the particle trajectory.

A Geant4ePropagator class has existed in the CMS code base for many years, but has never been successfully used for the final trajectory fit. It converts the track state and associated errors from the representation used by CMS to the one used by Geant4. Furthermore, it defines the so-called target surface to Geant4e. This a virtual plane in 3d space within the detector volume to which the state of the particle must be propagated. For a track propagation from detector surface N to $N + 1$, the target surface will be aligned with the active silicon plane of the detector $N + 1$.

During the regular CMS reconstruction setup, only a slimmed down geometry of the tracking system is available to the modules. This geometry contains only the active Silicon surfaces and their location and rotation in space. The Geant4ePropagator needs a more detailed information about the geometry, especially also about the passive volumes. Therefore, the Geant4 geometry is loaded before the reconstruction job is started using a special *GeometryProducer*. Furthermore, the magnetic field representation of Geant4 is loaded by the GeometryProducer.

In addition to the functionalities already provided by the Geant4ePropagator, a

range of additional features were implemented as part of this thesis to successfully perform a full trajectory fit:

- **Error deflation in backward mode**

  While the errors on the track parameters are inflated during the forward propagation, they have to be deflated during the backward propagation. Geant4e must be instructed explicitly to perform the error deflation. This can be done via the method call:

```
1  G4ErrorPropagatorData::SetStage( G4ErrorStage stage )
```

  This method can either be called with `G4ErrorStage_Inflation` or `G4ErrorStage_Deflation` to set the handling of the error during the propagation.

- **Unit conversion**

  The internal units of CMS are `cm` for length and GeV for energies. In contrast, Geant4 uses `mm` and `MeV` for length and energies, respectively. Therefore, all conversions of track parameters and error matrices between CMS and Geant4 need to take the different units of the two software packages into consideration.

  Missing unit conversions have been added to the Geant4ePropagator to achieve a consistent data exchange between the two software packages.

- **Path length convention**

  The path length of the propagation can be queried from Geant4e once the propagation is complete. Herein, Geant4e will always return a positive path length, also for backward propagations. The CMS convention however is, that a backward propagation must result in a negative path length value. To comply with this convention, the Geant4ePropagator was changed to provide a positive or negative signed path length to the CMS reconstruction, depending on the propagation direction.

Furthermore, a programming error in the Geant4 package was discovered during the integration with CMSSW. This software bug, called a memory leak, caused Geant4e to allocate unnecessary amounts of memory for each track propagation and prevented the processing of more than 1000 tracks due to the system running out of free memory. A bug fix was provided to the Geant4 development team [99] which corrects the copious memory usage. As a positive side effect of this correction, the overall runtime of the propagation step decreased by a factor of 4 [99].

The provided bug fix was integrated into the official Geant4 release and all experiments using the Geant4e propagation can now benefit from the improved memory and runtime behaviour of the application.

12.5. Fitting of Muon Trajectories

As a first application of track fits with Geant4e, Monte Carlo generated muons have been used. These particles originate at the center of the detector and their direction is equally distributed in ϕ and the η range $\eta = [-2.5, 2.5]$, exactly the coverage of the CMS tracker. The transverse momentum p_{T} of the muon particles is equally distributed between $[0.9, 1.1]\,\mathrm{GeV}$ as particles in this low transverse momentum range are most heavily affected by energy loss in the detector. Half of the muons have a positive charge, while the other half has a negative charge.

The tracks of the particles have been fitted with the parametric material estimation and the Geant4e method.

The first quality criteria which will be studied are the pull distributions of the fitted parameters. Overall, five parameters are fitted with the Kalman method as introduced in section 7.3. The pull of the fit parameter T can be computed using the formula:

$$p(T) = \frac{T_{\mathrm{reco}} - T_{\mathrm{sim}}}{\sigma_{\mathrm{T}}}$$

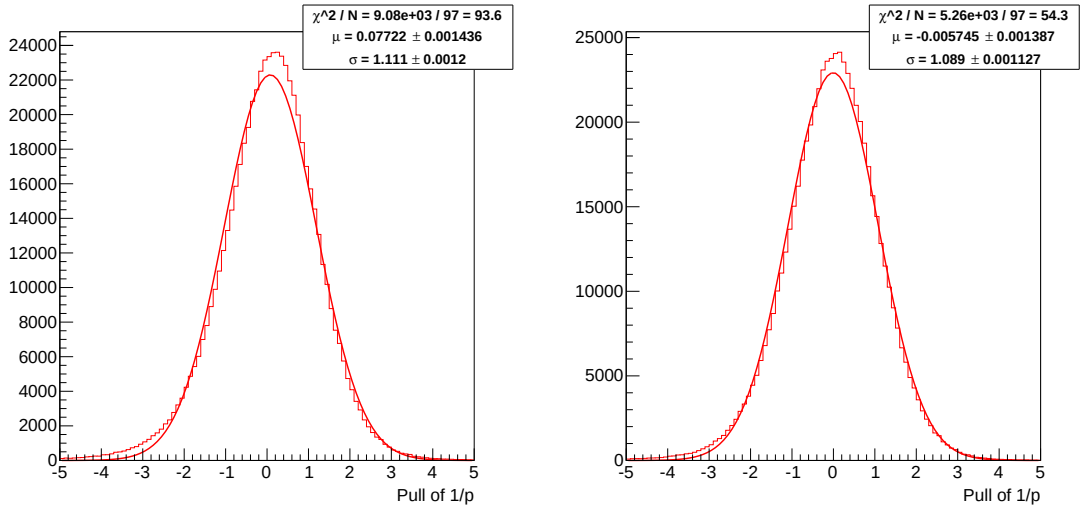
Herein, T_{reco} is the parameters value provided by the reconstruction, T_{sim} is the true value known from the simulation and σ_{T} is the error on the reconstructed value. The pull distribution resulting from computing the above formula for many fitted parameters allows to evaluate a possible bias of the reconstruction and also whether the uncertainties on the reconstructed parameters are under- or overestimated.

Figure 12.8a shows the pull distribution of the regular, parametric fit method for the fit parameter $1/p$ while figure 12.8b shows the same distribution for the Geant4e-based fit method.

A gaussian function has been fitted to both distributions and the resulting mean and sigma can be used to evaluate the quality of the track fitting procedure.

Furthermore, figures 12.9a and 12.9b show the pull distributions for the parametric and Geant4e-based fits for the θ fit parameter. A similar trend as in the $1/p$ pull distributions is visible here: the Geant4e-based method is able to achieve slight improvements both for the mean values of the pull distribution as well as for the width.

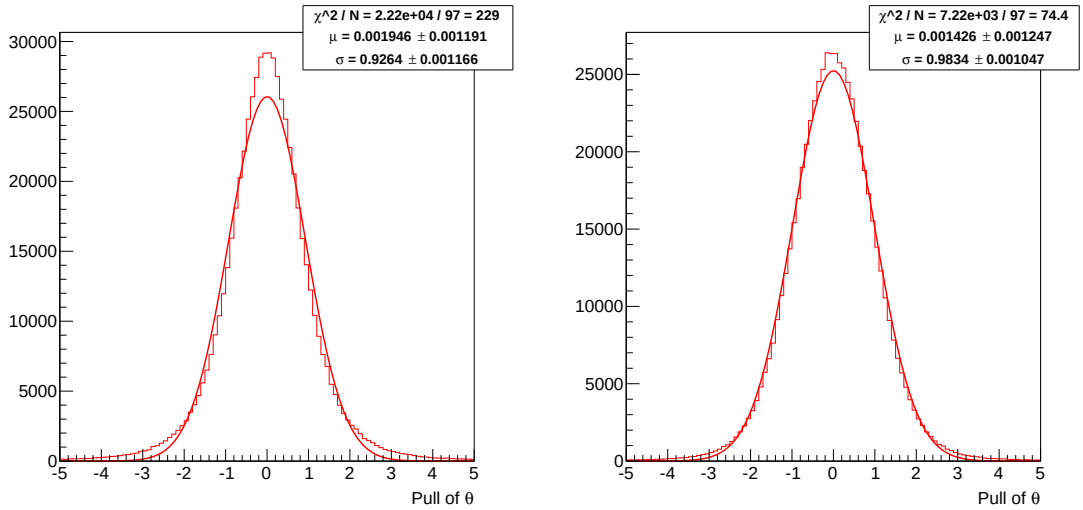
As described in chapter 5, the CMS tracker has very different material densities and types at various locations along the particle trajectory. Therefore, it is also of interest to study the fitting performance along various parts of the detector.



(a) 1/p pull distribution of the parametric method.

(b) 1/p pull distribution of the Geant4e method.

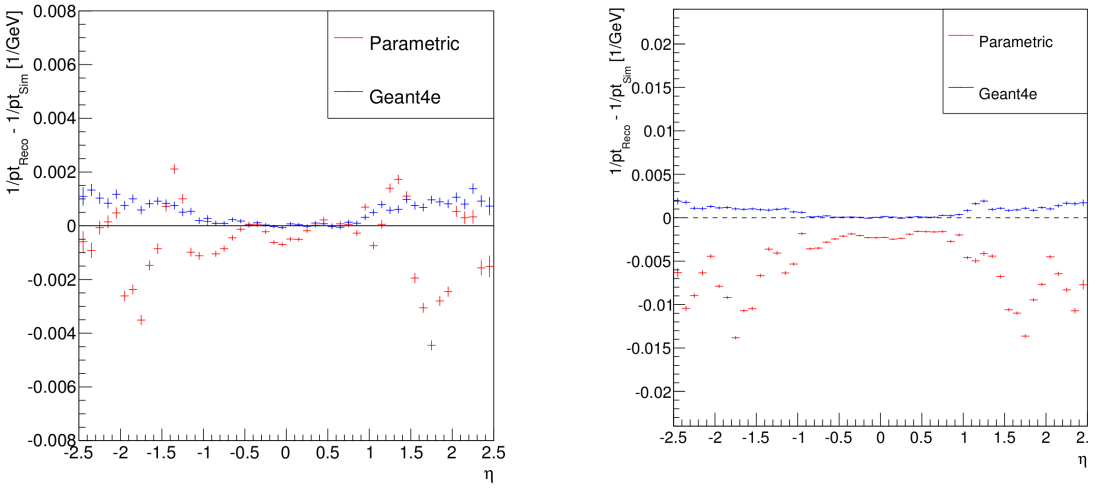
Figure 12.8.: Pull distributions of the fit parameter 1/p for the parametric and Geant4e methods.



(a) Pull of θ distribution of the parametric method.

(b) Pull of θ distribution of the Geant4e method.

Figure 12.9.: Pull distributions of the fit parameter θ for the parametric and Geant4e methods.



(a) Residual of the $1/p_{\mathrm{T}}$ value when fitting all track hits with the parametric and Geant4e methods.

(b) Residual of the $1/p_{\mathrm{T}}$ value when fitting only the strip detector hits with the parametric and Geant4e methods.

Figure 12.10.: Residual of the $1/p_{\mathrm{T}}$ value for various bins in η when using either all available hits of the track (left), or only the hits in the strip tracker (right). Note the different scale of the y-axis between the plots.

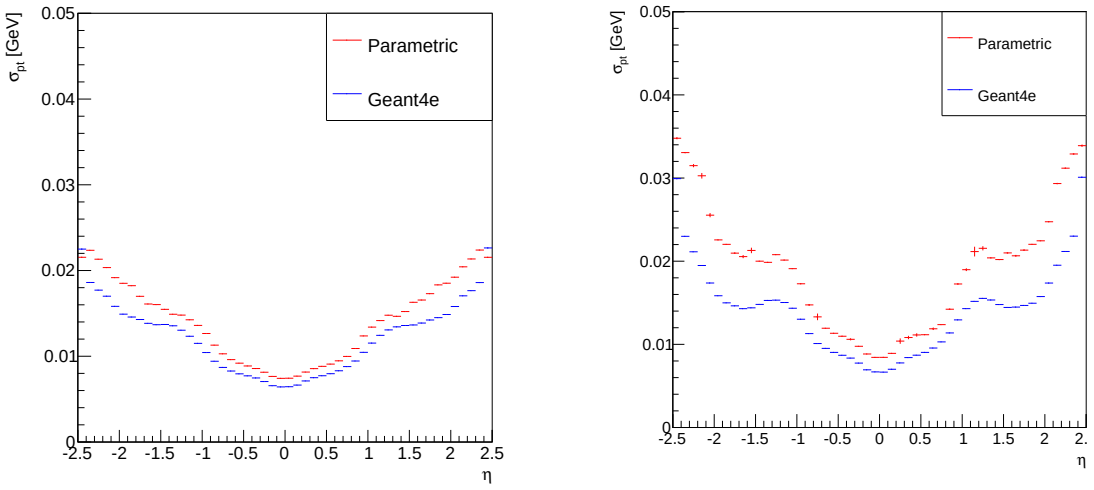
The mean residual of the fitted inverse transverse momentum parameter:

$$res(p_{\mathrm{T}}) = \frac{1}{p_{\mathrm{T}reco}} - \frac{1}{p_{\mathrm{T}sim}}$$

plotted over η is displayed in figure 12.10a. The improvements in the barrel region are visible when using the Geant4e-method compared to the parametric method. In the transition and endcap region of the tracker, starting at $|\eta| \approx 0.7$ the Geant4e-method shows clear improvements. The residuals in the outer bins of η are closer to zero and overall show a much flatter behaviour.

Figure 12.11b shows the distribution of the residuals over η in case only hits from silicon strip detectors are used. Therefore, the fit will be performed without the two to four pixel hits in the center of the tracker. Fits with missing pixel hits are necessary to reconstruct particles from secondary vertices which are outside of the last pixel layer or cosmic muon tracks which enter the detector from the outside and do not necessarily cross the pixel tracker volume.

In this scenario, the performance of the Geant4e-method is significantly better. The bias in the barrel region of the parametric method cannot be observed with Geant4e. While the residual in the endcap worsens considerably for the parametric method, the Geant4e-methode is able to maintain the smaller residual already seen in the fit with all tracker hits.



(a) Sigma of the transverse momentum measurement when fitting all track hits with the parametric and Geant4e methods.

(b) Sigma of the transverse momentum measurement when fitting only the strip detector hits of the track with the parametric and Geant4e methods.

Figure 12.11.: Sigma of the transverse momentum measurement for various bins in η when using either all tracker hits (left), or only the hits in the strip tracker (right).

Figures 12.11a and 12.11b show the measurement error on p_{T} as returned by the track fit for all hits and the fit with strip hits-only, respectively. The plots show, that the error on the reconstructed p_{T} is slightly smaller, especially in the endcaps, when using the Geant4e-method compared to the parametric one. As expected, the error of the strip hits-only fit is larger, as fewer measurement points were included in the track fit.

Chapter 13

Reconstruction of the short-lived Kaon Mass

The copious production of kaons in the pp-collisions delivered by the LHC machine allows to use the decay of these particles as a reference for the evaluation of reconstruction algorithms. The well-known reference mass of the kaon can be compared with the mass resulting from the event reconstruction in simulated and measured events. Especially interesting for track reconstruction validation is the decay of the short-lived kaon K_s^0. Table 13.1 lists the hadronic decay modes of the K_s^0 into either two pions of neutral charge, two pions of opposite charge or two pions of opposite charge and one neutral pion.

Table 13.1.: Branching fractions of the hadronic decays of the short-lived kaon K_s^0 [4].

Decay Mode	Branching Fraction Γ_i/Γ
$K_s^0 \to \pi^0\pi^0$	30.69
$K_s^0 \to \pi^+\pi^-$	69.20
$K_s^0 \to \pi^+\pi^-\pi^0$	$3.5 \cdot 10^{-7}$

Apart from the highly suppressed decay $K_s^0 \to \pi^+\pi^-\pi^0$, the decay $K_s^0 \to \pi^+\pi^-$ is the only one which can be reconstructed solely based on tracker information.

The mass of the K_s^0 particle is $m_{K_s^0} = 497.614 \pm 0.024\,\text{MeV}$ according to the fit of the Review of Particle Physics publication, which combines the measured results from multiple experiments [4]. This value will be used in the following as the K_s^0 mass reference.

13.1. K_s^0 Detection and Reconstruction

The decay of a K_s^0 particle leaves a distinct signature in the tracker: two tracks with opposite charge emerge from a displaced vertex. Due to the K_s^0 being neutral, no single track leads to this secondary vertex.

A range of cuts have been applied to the reconstructed particle tracks to identify possible K_s^0 decays [100]:

- The normalized χ^2 value of the track fit must be 5 or better
- The number of valid hits on one track must be 6 or higher

All tracks passing these initial quality criteria are evaluated in pairs of opposite charge combinations. Between each of the possible combinations, the point of closest approach between the tracks is computed. Two cuts are applied to reduce the candidate combinations:

- The mass of the di-pion system must be smaller than 0.6 GeV
- The closest distance between the tracks must be smaller than 1 cm

All track combinations which pass these criteria are used as input to a vertex fit. Once this fit has been performed, the position and associated errors for a possible secondary vertex have been computed and additional quality cuts can be applied:

- The normalized χ^2 of the vertex fit must be 7 or smaller
- The innermost hits on both tracks originating from the secondary vertex must not be closer to the beam spot than 4 sigma of the radial vertex position. This helps to ensure that the two tracks actually originate from the secondary vertex and do not stem from another primary vertex at the beam spot
- The reconstructed mass at the secondary vertex must lie between ± 70 MeV of the K_s^0 reference mass as listed by the PDG [4]

13.2. Dataset

For the following study, simulated collision events and data recorded by the CMS detector will be used. The simulated collisions have been generated at a center-of-mass energy of $\sqrt{s} = 8$ TeV using the Pythia6 [44] event generator with the Z2star tune [101]. The decay and the interaction of the generated particles with the detector were simulated using the Geant4 package.

The data presented here was recorded by the CMS detector at a center-of-mass energy of $\sqrt{s} = 8\,\text{TeV}$ at the beginning of the year 2012.

Both, simulated and recorded events, have been reconstructed using the CMS reconstruction and K_s^0 decay candidates have been identified using the technique described above.

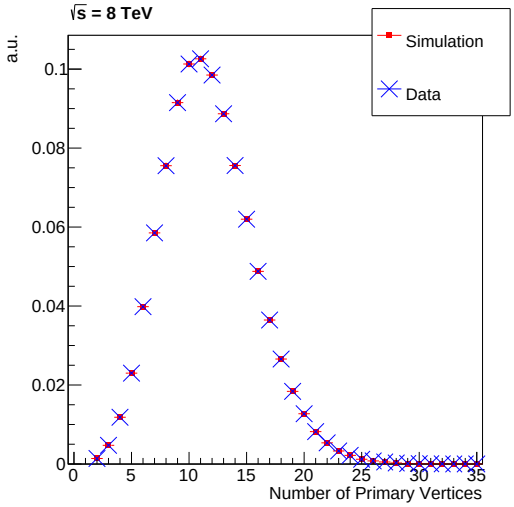
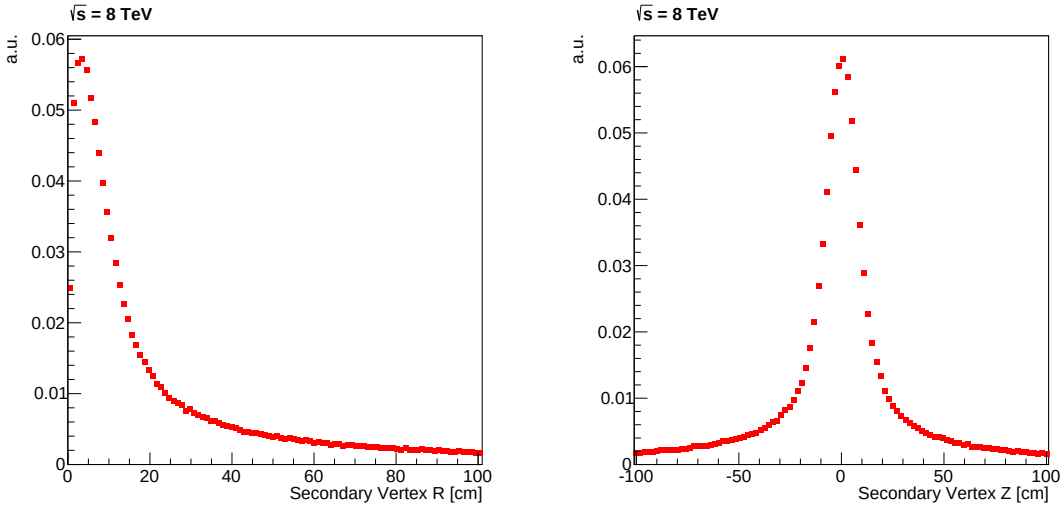


Figure 13.1.: Number of primary vertices (NPV) in recorded and simulated events. The simulated events have been reweighted to fit the NPV distribution observed in recorded events.

Figure 13.1 shows the distribution of the number of primary vertices (NPV), both in the recorded and simulated events with the average number being 11 in the recorded events. The simulated events have been reweighted so their NPV-distribution corresponds to the one observed in data.

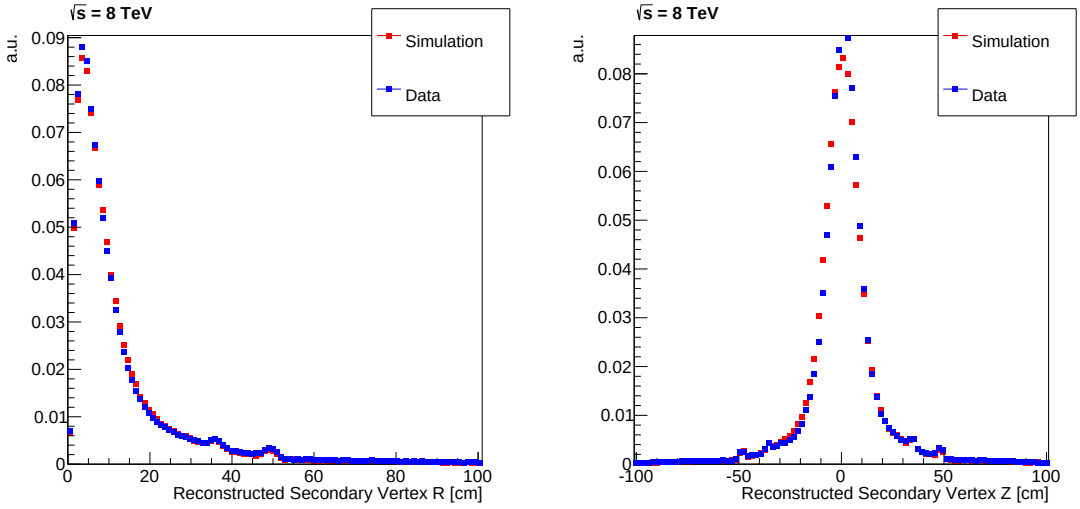
The position of the decay vertex of the K_s^0 in radial and Z-direction is shown in figures 13.2a and 13.2b. This information has been retrieved from simulation truth and therefore shows the decay position independent of the employed reconstruction technique. The most kaons decay close to the interaction point, but a significant amount of decays take place outside of the pixel tracker at $R > 12\,\text{cm}$ and $|Z| > 50\,\text{cm}$.

Only a part of the pion tracks resulting from the K_s^0 decays leave sufficient hits in the tracking system to allow for a detection and reconstruction of the decay. As described in chapter 7, tracks can only be found if sufficient hits exist in the inner tracking system to allow for an initial seeding of the track. Therefore, at least two properly measured hits on each of the pion tracks are required to find these tracks and in turn reconstruct the decay vertex of the K_s^0. As the seeding is performed in the most precise, innermost layers of the tracker, highly displaced vertices are less likely to be detected.



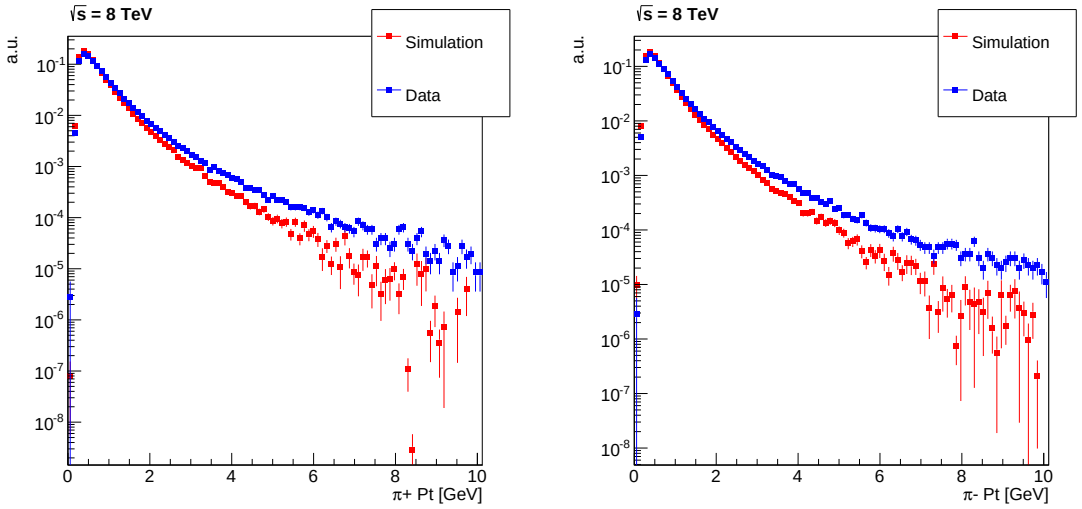
(a) Radial position of the K_s^0 decay vertex. **(b)** Z position of the K_s^0 decay vertex.

Figure 13.2.: Decay position of the K_s^0 in relation to the interaction point. This location has been retrieved from simulation truth.



(a) Radial position of the K_s^0 decay vertex. **(b)** Z position of the K_s^0 decay vertex.

Figure 13.3.: Reconstructed location of the K_s^0 decay vertex position in simulated and recorded events.



(a) Positive charged pions originating from the K_s^0 decay.

(b) Negative charged pions originating from the K_s^0 decay.

Figure 13.4.: Transverse momentum spectra of the positive and negative charged daughter pions used to reconstruct the short-lived kaon.

Figures 13.3a and 13.3b show the distribution of the K_s^0 decay location provided by the event reconstruction using the technique described in section 13.1 for both simulated and recorded events. The location of the reconstructed vertices is consistent between both datasets. The decrease of found decays with increasing distance from the interaction point is visible in both plots with virtually no reconstructed decays outside of $R > 55\,\text{cm}$ and $|Z| > 60\,\text{cm}$. Increases in the efficiency are visible in the plot 13.3b the positions $|Z| = 34.5\,\text{cm}$ and $|Z| = 46.5\,\text{cm}$. These are effects introduced by an interplay between the tracker layout and the reconstruction algorithm. Four layers of detection elements of the forward pixel detector are located at $|Z| = 34.5\,\text{cm}$ and $|Z| = 46.5\,\text{cm}$ on both sides of the interaction point. The additional measurement points provided by the layers at these locations enable the vertex finding to achieve a better, though very localized, increase of the efficiency.

Figures 13.4a and 13.4b show the distributions of the transverse momentum (p_T) of the positive and negative charged pions which are used to reconstruct the K_s^0 mass. Both plots show, that the p_T spectrum in data is harder, meaning more high energetic pion tracks have been reconstructed in recorded events as in the simulation.

As expected, this observation is also visible in figure 13.5, which shows the transverse momentum of the K_s^0 reconstructed from the two pion tracks.

These discrepancies in the momentum distribution can either be a result of imperfections in the modelling of the Pythia6 Monte Carlo generator or an artifact

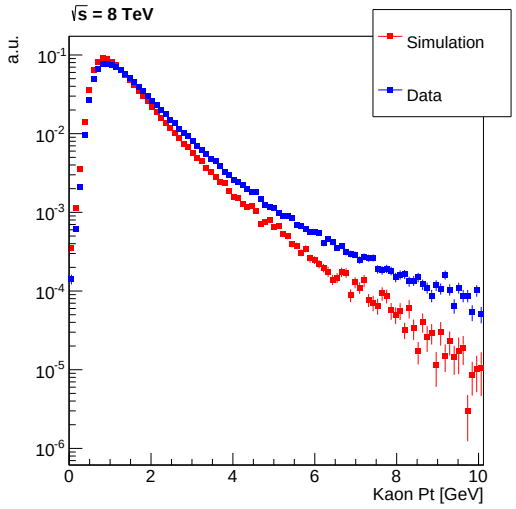


Figure 13.5.: Transverse momentum distribution of the K_s^0 reconstructed from two pion daughter particles.

of the slight differences in the reconstruction efficiencies between simulated and measured events. For the following study, no correction to the simulated spectrum will be applied, as simulated and measured events will always be treated separate and the absolute transverse momentum of both will not enter into the same computations.

13.3. Mass Reconstruction

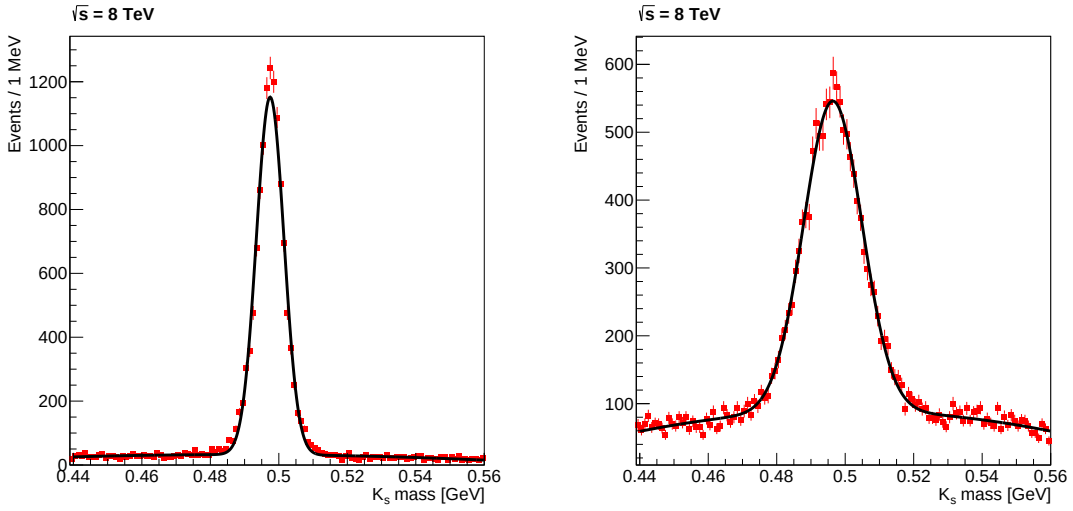
The reconstructed mass of each individual kaon decay is collected in a histogram and can be used to derive the invariant mass of the particle. The K_s^0 mass spectrum was measured differently either in bins of pseudo-rapidity η, the transverse momentum of the K_s^0 or the average transverse momentum of the pions.

$$D_{peak}(x) = g * exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2) \tag{13.1}$$

$$D_{bkg}(x) = a + bx + cx^2 \tag{13.2}$$

$$D(x) = D_{peak} + D_{bkg} \tag{13.3}$$

Each of these distributions is fitted with equation 13.3, which combines models of the decay peak and background distribution of the reconstructed particle decay. The mass peak of the K_s^0 decay is modeled as a Gaussian function, listed in equation 13.1, and the mean μ of the fitted Gaussian function is used as an estimator for the kaon invariant mass in this specific bin. The σ parameter of



(a) Central barrel bin $0.0 < \eta < 0.25$.

(b) Endcap bin $2.0 < \eta < 2.25$.

Figure 13.6.: Distribution of the K_s^0 mass for a central barrel and endcap bin. Both mass distributions have been fitted with function 13.3, a gaussian function to match the signal peak and a quadratic polynomial to model the background.

the Gaussian function can be used to estimate the reconstruction resolution of the invariant mass. Equation 13.2 is a quadratic polynomial and is used to model background contribution to the histogram.

The same modelling method has been used in the CMS publication [97] where the previous studies on the K_s^0 mass reconstruction in various η bins have been performed and resulted in the plot shown in figure 12.3. Using the same method in this study allows to perform a direct comparison of the results.

Figure 13.6a shows the distribution of the reconstructed K_s^0 mass from measured data for an η bin in the central barrel part of the detector. The histogram entries are overlayed with the result of fitting function 13.3 to the distribution. Figure 13.6b shows the same type of plot for a η bin in the endcap area of the detector. It can be observed from these plots, that the background contribution in the endcap region is more pronounced and also the width of the decay peak is wider in the endcap area.

As both plots show, the fitted function 13.3 is able to model the observed distributions, both in the central barrel and endcap region, properly. The distributions and mass fits for all η bins are listed in appendix D.

Figure 13.7 presents all mass fits for simulated events of the individual η bins over the whole acceptance range of the CMS Tracker. The red entries, labeled "Regular", are the result of the regular reconstruction using the parametric material estimation. The blue entries, labeled "Geant4e" show the reconstructed mass

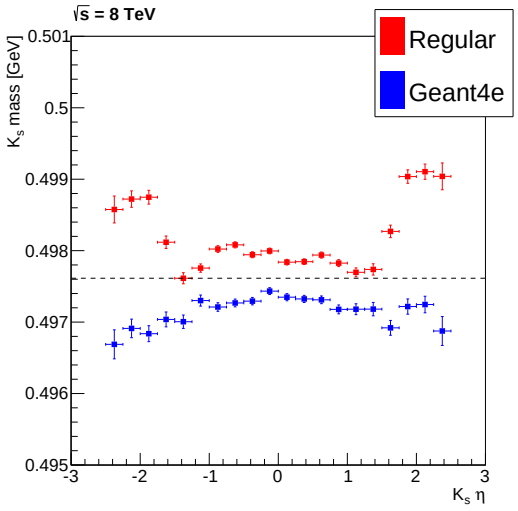


Figure 13.7.: Reconstructed K_s^0 mass in η-bins for the regular reconstruction and the uncorrected Geant4e reconstruction. The black dotted line marks the reference mass of the K_s^0 of the Particle Data Group [4].

when using Geant4e during the track fit of the K_s^0 reconstruction. The black dotted line marks the reference mass of the K_s^0.

The Geant4e-based fit is able to reconstruct the K_s^0 mass with a smaller fluctuation between the η bins than the regular method. Especially in the endcaps, the changes in mass between the different bins in η are smaller. While the regular method is slightly overestimating the K_s^0 mass, the Geant4e-based fit systematically underestimates the mass.

As figure 13.7 is purely based on simulated events, the same Geant4 material has been used to simulate and reconstruct the events. One naively expects the Geant4e-method to achieve a flat distribution over η and to match the K_s^0 mass correctly.

However, the material maps in the parametric method have been calibrated on known resonance peaks, like the K_s^0 decay, in measured events to achieve the best possible performance in the reconstruction of recorded data. This allows for a free parameter, the overall scale of the track momentum in the reconstruction, to be adjusted to achieve the correct scale on measured data.

$$p_{\text{corr}} = c * p_{\text{uncorr}} \tag{13.4}$$

To also account for this momentum scale parameter, a similar technique has been applied to the Geant4e-method to achieve the correct momentum scale. The momentum of the pion tracks have be multiplied by the factors 0.01, 0.05 and 0.005 and the average K_s^0 mass for all bins in η has been computed. With the constraint that the average K_s^0 mass must be the reference value of $m_{K_s^0} =$

$497.614 \pm 0.024\,\text{MeV}$. An interpolation between the tested correction factors was performed and the optimal correction factor to achieve the K_s^0 reference mass on average was determined to:

$$c = 1.001294$$

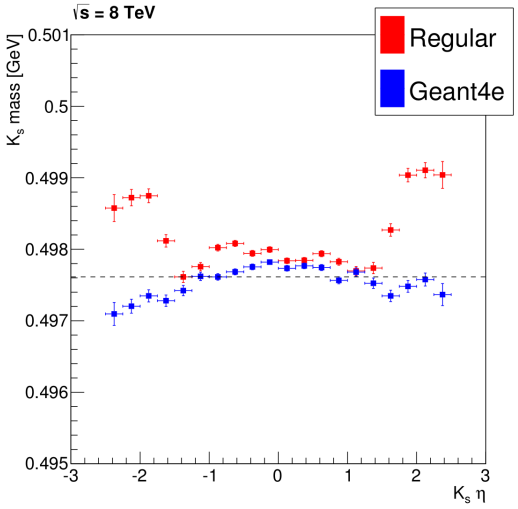
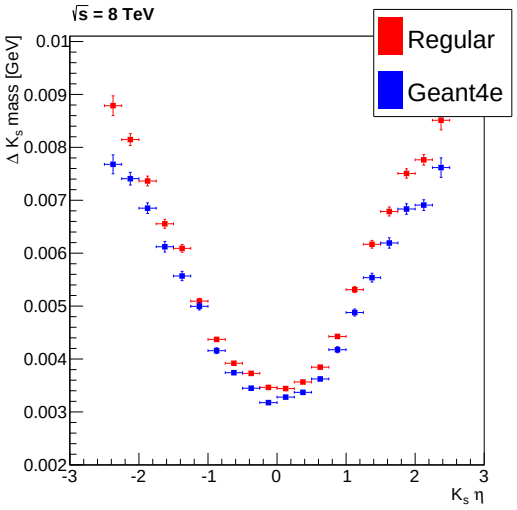


Figure 13.8.: Reconstructed K_s^0 mass in η-bins for the regular reconstruction and the corrected Geant4e reconstruction, with a momentum correction of $c = 1.001294$ applied.

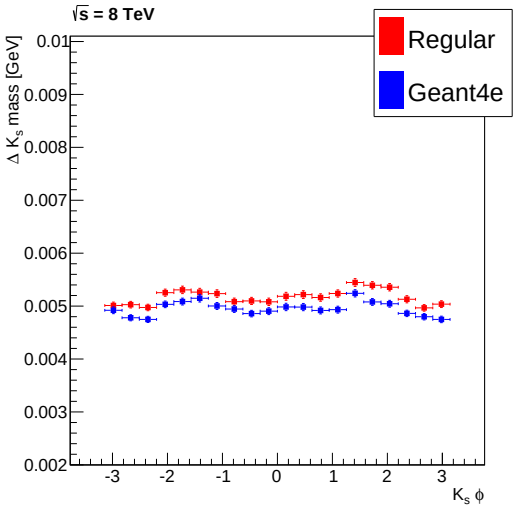
Figure 13.8 shows the reconstructed K_s^0 mass with the momentum scale correction applied to the Geant4e-fitted tracks. As expected, the correction shifts the reconstructed mass on top of the reference mass. A non-flat structure in η is still visible for the Geant4e-based reconstruction. This is especially visible in the endcap, where also the statistical error bars are larger, due to the wider reconstruction resolution. However, for all η bins, except for the one at $\eta = -1.4$, the Geant4e method is able to achieve a better compatibility with the reference K_s^0 mass than the the regular fit method does. This improvement is especially visible in the endcap regions for $|\eta| > 1.4$.

Figures 13.9a and 13.9b show the reconstruction resolution of the K_s^0 mass peak of the regular method and the scale-corrected Geant4e-method. Figure 13.9b shows the resolution of these two methods over ϕ of the reconstructed K_s^0. While a slight decrease of the resolution in the regions around $|\phi| = -2$ is visible for both methods, the Geant4e-method is able to achieve a better resolution, about 4% on average.

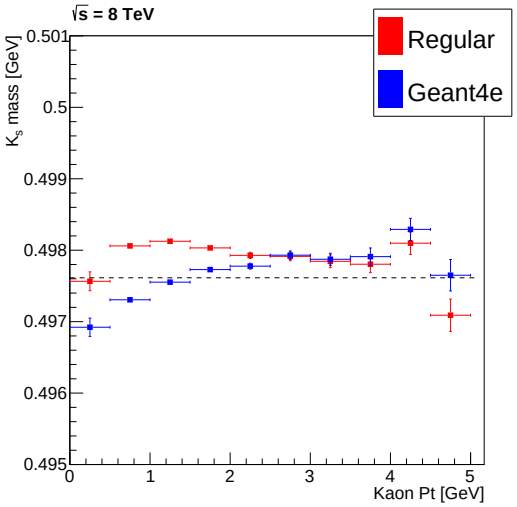
Figure 13.9a, which shows the resolution plotted over η, a small improvement when using Geant4e is visible over the whole η range. Especially in the three outermost bins of the endcap the improvement is most pronounced.



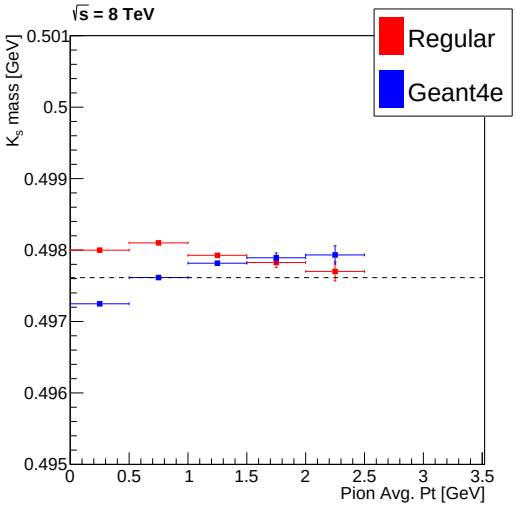
(a) K_s^0 mass resolution retrieved from the mass peak fit plotted in η bins.



(b) K_s^0 mass resolution retrieved from the mass peak fit plotted in ϕ bins.



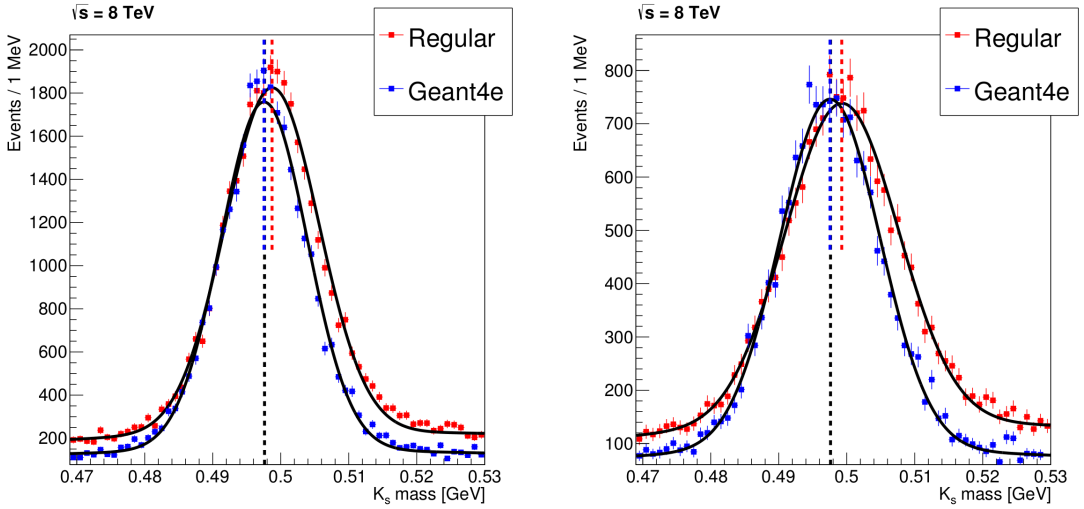
(c) Reconstructed K_s^0 mass depending on the transverse momentum of the kaon.



(d) Reconstructed K_s^0 mass depending on the average transverse momentum of both daughter pions.

Figures 13.9c and 13.9d display the reconstructed K_s^0 mass in relation to the transverse momentum of the kaon and average pion p_{T}, respectively. A dependence of the reconstructed K_s^0 mass on the transverse momentum is visible for both reconstruction methods, although the shape of the dependence is different in the region $p_{\mathrm{T}} < 2\mathrm{GeV}$.

13.4. Reconstruction Performance in Specfic Detector Regions



(e) Reconstructed K_s^0 mass distribution if at least one of the two daughter pions is located in the region $1.5 < \eta < 1.75$.

(f) Reconstructed K_s^0 mass distribution if at least one of the two daughter pions is located in the region $2.25 < \eta < 2.5$.

Figure 13.9.: Reconstructed K_s^0 mass distribution of the regular and Geant4e fitting methods for two detector regions with challenging material distributions. The black dotted line indicates the reference mass of the K_s^0 particle. Equation 13.3 has been fitted to both distributions and the resulting mass is drawn as red and blue lines for the regular and the Geant4e reconstruction, respectively.

As described in chapter 5, the transition and endcap regions of the CMS tracker have a larger material budget as the central barrel region. Therefore, improvements to the reconstruction material model can be expected to be most pronounced in these regions. In fact, the investigation on muon tracks and the reconstructed K_s^0 mass presented in this and the previous chapter have shown the biggest improvements in the transition and endcap regions.

To study the magnitude of possible improvements in detail, K_s^0 decays have been selected where at least one of the daughter pion tracks is located in material-dense η regions of the tracker. Figure 13.9e presents distributions of the reconstructed K_s^0 mass for the regular and Geant4e methods in the transition region $1.5 < \eta < 1.75$ where the radiation length x/X_0 is the largest. Both distributions have been fitted with equation 13.3 and the resulting K_s^0 mass is drawn as red and blue lines for the regular and the Geant4e reconstruction, respectively. The reference K_s^0 mass is plotted as a black line. Figure 13.9f shows the same type of plot, but requiring at least one of the daughter pion tracks in the endcap region of $2.25 < \eta < 2.5$.

Table 13.2.: Reconstructed K_s^0 masses using the regular and Geant4e methods in the transition and endcap region. The relative and absolute difference of the resluting K_s^0 masses compared to the reference mass is also listed.

	$1.5 < \eta < 1.75$	$1.5 < \eta < 1.75$
Regular Method		
K_s^0 mass [MeV]	498.78 ± 0.07	499.3 ± 0.1
Abs. diff to ref. [MeV]	1.7	1.69
Rel. diff to ref. [%]	0.25	0.34
Geant4e Method		
K_s^0 mass [MeV]	497.54 ± 0.05	497.5 ± 0.1
Abs. diff to ref. [MeV]	0.047	0.11
Rel. diff to ref. [%]	0.015	0.015

Table 13.2 lists the reconstructed K_s^0 masses for the two methods and the absolute and relative difference to the reference K_s^0 mass $m_{K_s^0} = 497.614 \pm 0.024\,\text{MeV}$[4].

In both studied regions, transition and endcap of the tracker, a significant improvement of the K_s^0 mass reconstruction when using the Geant4e method is visible. The regular material method results in a systematic shift of the mass distribution to higher masses, which cannot be observed with the Geant4e method. The K_s^0 mass reconstructed with the Geant4e method is compatible within the errors with the reference value while the result of the regular reconstruction is not.

13.5. Mass Reconstruction of Measured Events

The next logical step is to evaluate the performance of the Geant4e-based track reconstruction on events recorded with the detector. Figure 13.10 shows the K_s^0 mass reconstruction using the Geant4e-method both on simulated and measured events.

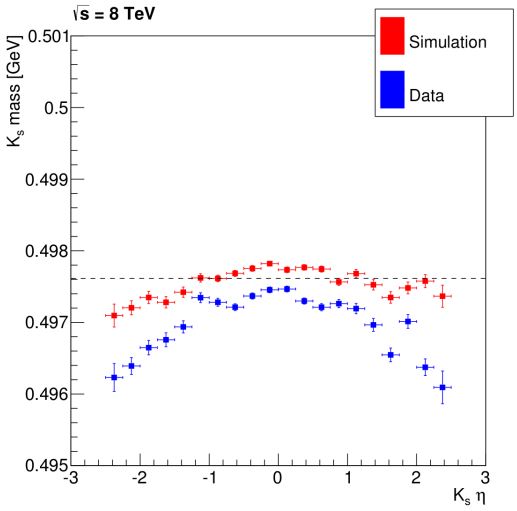


Figure 13.10.: Reconstructed K_s^0 mass using the Geant4e material model for simulated and real events.

The momentum scale correction derived in the previous section has been applied to both.

Employing the Geant4e-method results in a reconstructed K_s^0 mass which is systematically below the reference value. Additionally, the reconstructed mass in measured events diverts stronger from the reference mass as in simulation.

13.6. Conclusion and Future Steps

In this chapter, the mass reconstruction of the short-lived kaon K_s^0 was employed to validate the performance of the Geant4e-based track reconstruction.

For simulated events, a global momentum scale correction has been derived. After applying this correction, the new fitting method shows improvements in the reconstruction of the K_s^0 mass in the barrel region, but enhancements are especially visible in the endcap region. Also a slight resolution improvement of the reconstructed mass peak can be observed.

This improved performance cannot be achieved when using the Geant4e-method on events measured with the CMS detector in the year 2012. The reconstructed mass is systematically lower than the K_s^0 reference mass. One possible explanation for this result is the discrepancy between the Geant4 geometry setup and the actual material present in the detector. This implies, that the current Geant4 detector model is not sufficiently precise to achieve an improvement in the reconstruction performance in measured events.

The Geant4e-method developed as part of this thesis allows to improve this

Geant4 geometry to better reflect the actual material distribution within the detector. For the first time in CMS, particle tracks were fitted using the full Geant4 material estimation and detailed comparisons between simulated and reconstructed tracks were performed. Eventually, insights gained with these kind of studies can be fed back to the Geant4 detector model to improve the overall quality of simulated events in the CMS detector.

As shown in this chapter, the reconstruction performance of particle masses can be improved with the Geant4e method if a sufficiently precise Geant4 geometry model of the detector is available. Also, decay products in the endcap region can be more precisely reconstructed.

This allows to better reconstruct complex particle decays, where at least one decay product trajectory is located in an endcap region.

Chapter 14

Conclusion and Outlook

The first part of this thesis focused on improving the processing speed of HEP applications by making better use of the vector units in modern CPUs. Existing implementations can be adapted and auto-vectorized using modern C++ compilers. Furthermore, the `vdt` library, which has been developed and evaluated as part of this work, speeds up the computation of mathematical functions in simulation and reconstruction software.

Both techniques have been applied to the CMS reconstruction to improve the runtime performance of the vertex finding, resulting in a speedup of more than a factor 2. The improved version of this algorithm has been successfully used to reconstruct the events recorded during the 2012 data taking, which led to the announcement of the Higgs boson discovery in July 2014.

The application of auto-vectorization significantly widens the physics reach of current and future experiments by the ability to increase the number of processed measurements and thus study more rare processes. Future software designs for event simulation and reconstruction applications need to consider the vectorized processing of data in order to fully utilize the hardware capabilities of today and tomorrow.

The `vdt` library is now available open-source and royalty-free to allow other physics experiments to benefit from the collection of fast and vectorized mathematical functions. The Geant4 material simulation package uses the `vdt` library in its most recent release version 10 to improve the runtime of the application. Furthermore, `vdt` has been integrated into the ROOT data analysis framework.

To cope with the increased amount of recorded data expected during the upcoming years of the LHC, the parallel computing power of GPUs are a promising hardware option and complements the use of vector units in classical CPUs. In order to port the CMS track reconstruction to GPU accelerators, the algorithms need to be adapted to the differing memory layout and massive parallelism provided by this hardware. As part of this thesis, algorithms have been developed

which can perform well with single-precision floating point data and provide a high level of parallelism within the processing of one event.

Triplet finding and joining algorithms were presented, which have been tailored to work well on GPU hardware. Cut values to be used by these algorithms have been determined by a representative top pair decay dataset. Triplets, which combine hits of the barrel tracker layers 1 to 5 and 8, can be reconstructed with an efficiency of 80% and an acceptable fake rate. In the second stage, triplets from the inner barrel part of the tracker can be combined with an efficiency of 90%.

The triplet finding algorithm was implemented using OpenCL and the `clever` library. The physics performance expected from the cut studies is achieved when executing the algorithms on a NVIDA consumer GPU. For events containing 400 and more tracks, the GPU-based implementation outperforms the classical CPU implementation and can achieve a more than ten times faster reconstruction time for events with more than 1000 tracks.

The OpenCL implementation of tracking algorithms presented in this work shows, that significant runtime improvements of reconstruction software can be achieved when using hardware accelerators liks GPU. This finding is an important landmark for the CMS collaboration, but also for the wider HEP community, as it presents one appealing way to meet the processing demands expected in the future of the LHC and other HEP experiments.

Following on the GPU implementation presented in this thesis, further work is ongoing in the CMS collaboration to port more parts of the event reconstruction to GPUs and integrate these processing steps into the CMSSW framework.

To improve the reconstruction quality of particle tracks, the impact of a fine-grained material model has been studied in this thesis. The detailed Geant4 model of the CMS detector was used during the track fit procedure of muon tracks and improvements compared to the default fitting method, both in the pull distributions of the fit and in the residuals, were achieved. The fit quality improved especially in the transition and endcap region of the tracker were the most material is located.

The decay of neutral kaons was used to study the potential benefits of the Geant4-based fitting method when applied to the mass reconstruction of particles. This study showed, that the reconstruction performance of particle masses in simulated events can be improved with the Geant4e method. The improvements were notably pronounced in the endcap region of the CMS tracker. Especially the reconstruction of complex decays like that of a B meson, which can result in many tracks, may benefit from this improvement as it is likely that at least one of these tracks will be in the endcap region.

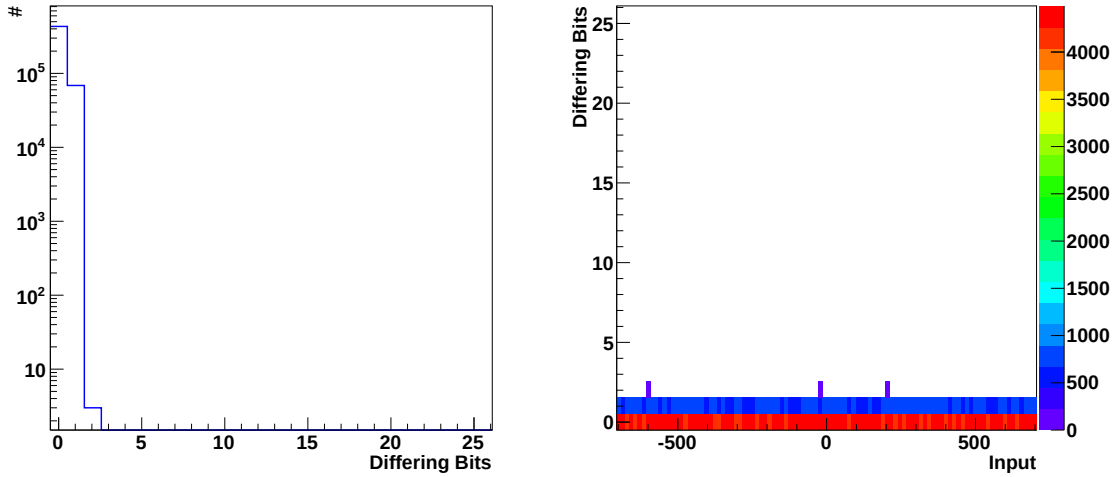
The novel method is also able to reconstruct the kaon decays in measured events, but the performance is not improved compared to the default fit method.

For the first time, the CMS track reconstruction is able to fit individual tracks using the full Geant4 material model. This allows to compare measured to simu-

lated events and use reference signals, like the kaon decay, to improve the Geant4 material model to better approximate the material distribution in the physical detector. Therefore, this thesis made an important contribution to improve the quality and precision of the measurements performed with the CMS experiment.

Appendix A

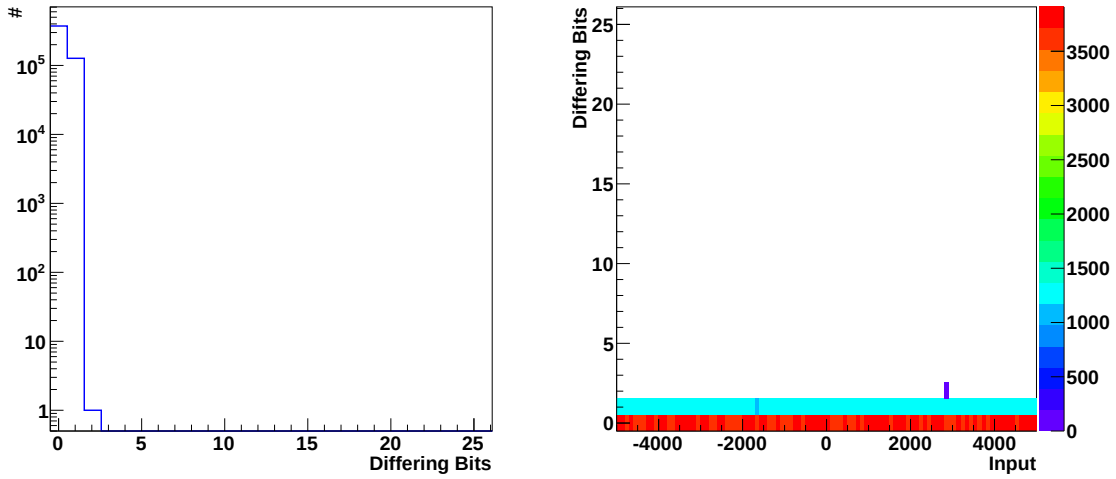
`vdt` Validation



(a) Histogram of the maximum number of differing bits of the `vdt` approximation wrt. to the `libm` reference implementation

(b) Frequency of the maximum number of differing bits of the `vdt` approximation wrt. to the `libm` reference implementation for the evaluated input range

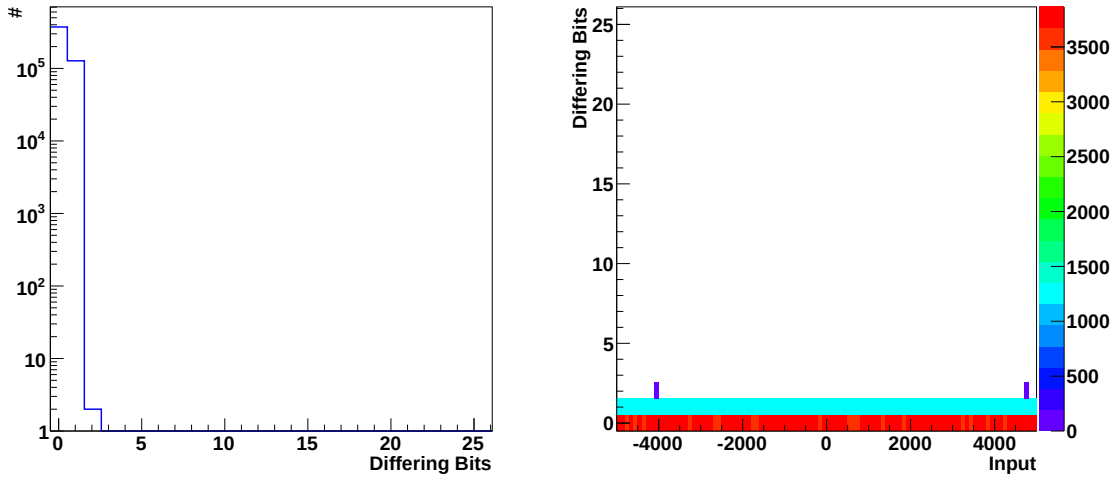
Figure A.1.: Both plots show the evaluation of the approximation accuracy of `vdt` with respect to the `libm` reference implementation for the double precision exp function. 500.000 random input values, uniformly distributed in the the considered range $[-705, 705]$ were used for this study.



(a) Histogram of the maximum number of differing bits of the `vdt` approximation wrt. to the `libm` reference implementation

(b) Frequency of the maximum number of differing bits of the `vdt` approximation wrt. to the `libm` reference implementation for the evaluated input range

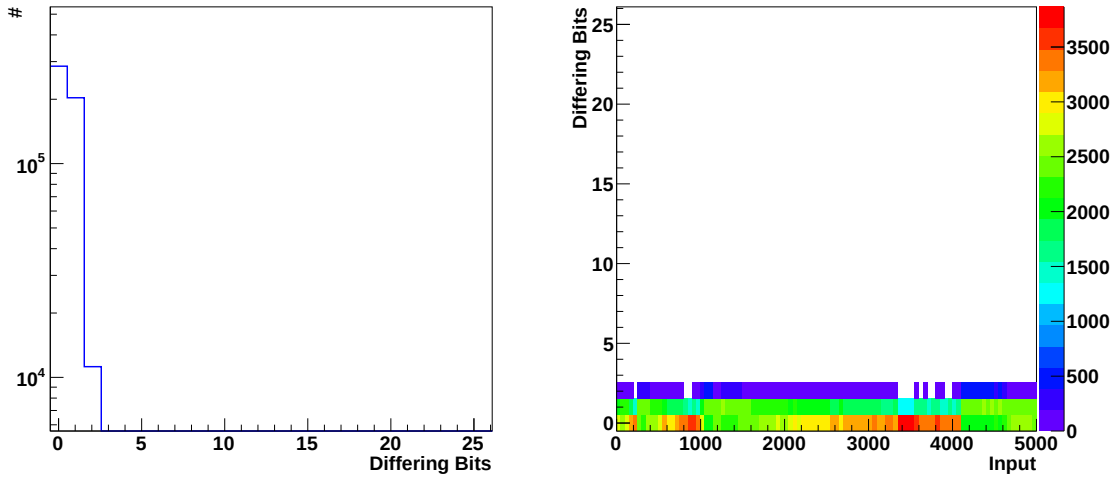
Figure A.2.: Both plots show the evaluation of the approximation accuracy of `vdt` with respect to the `libm` reference implementation for the double precision sin function. 500.000 random input values, uniformly distributed in the the considered range $[-5000, 5000]$ were used for this study.



(a) Histogram of the maximum number of differing bits of the `vdt` approximation wrt. to the `libm` reference implementation

(b) Frequency of the maximum number of differing bits of the `vdt` approximation wrt. to the `libm` reference implementation for the evaluated input range

Figure A.3.: Both plots show the evaluation of the approximation accuracy of `vdt` with respect to the `libm` reference implementation for the double precision cos function. 500.000 random input values, uniformly distributed in the the considered range $[-5000, 5000]$ were used for this study.



(a) Histogram of the maximum number of differing bits of the `vdt` approximation wrt. to the `libm` reference implementation

(b) Frequency of the maximum number of differing bits of the `vdt` approximation wrt. to the `libm` reference implementation for the evaluated input range

Figure A.4.: Both plots show the evaluation of the approximation accuracy of `vdt` with respect to the `libm` reference implementation for the double precision inverse square root function. 500.000 random input values, uniformly distributed in the the considered range $[0, 5000]$ were used for this study.

Appendix B

Track Reconstruction on GPU

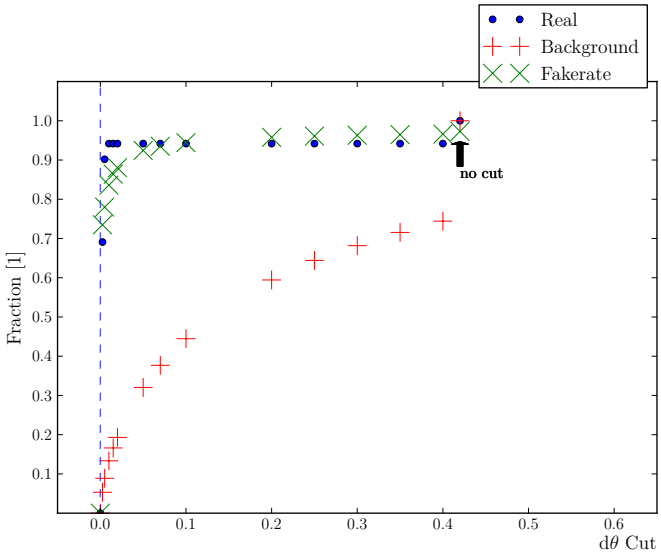


Figure B.1.: The fraction of real, fake and background triplets which pass the $d\theta$ cut plotted over a range of the cut quantity. For the plot point "no cut", none of the cuts are applied and all possible triplet combinations are in the fina

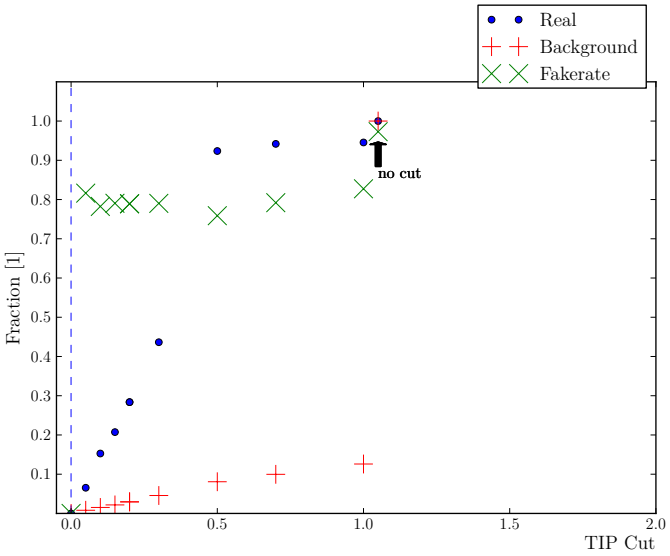
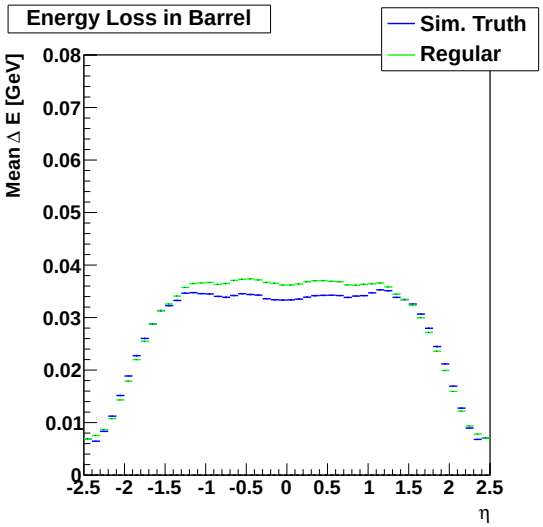


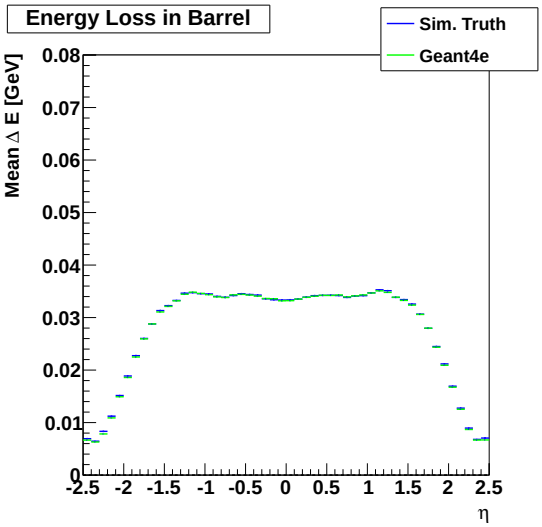
Figure B.2.: The fraction of real, fake and background triplets which pass the transverse impact parameter cut plotted over a range of the cut quantity. For the plot point "no cut", none of the cuts are applied and all possible triplet combinations are in the fina

Appendix C

Energy Loss Validation

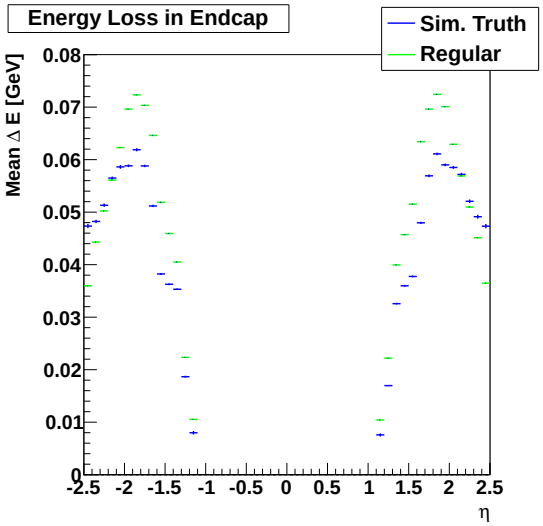


(a) Mean energy loss approximated by the parametrization method and the simulation truth.

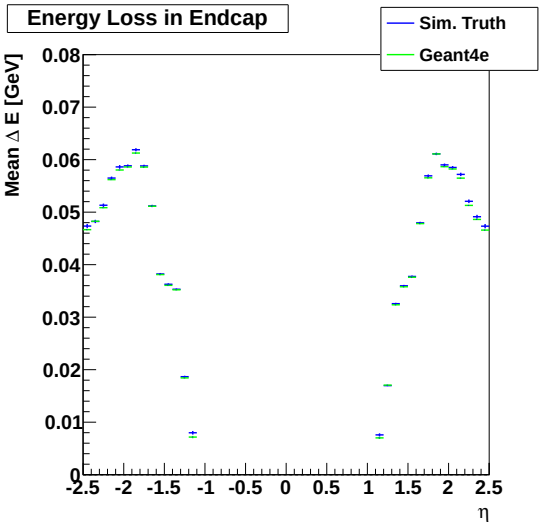


(b) Mean energy loss approximated by the Geant4e method and the simulation truth.

Figure C.1.: Comparison plots of the parametrization and Geant4e energy loss computations for the barrel part of the detector.

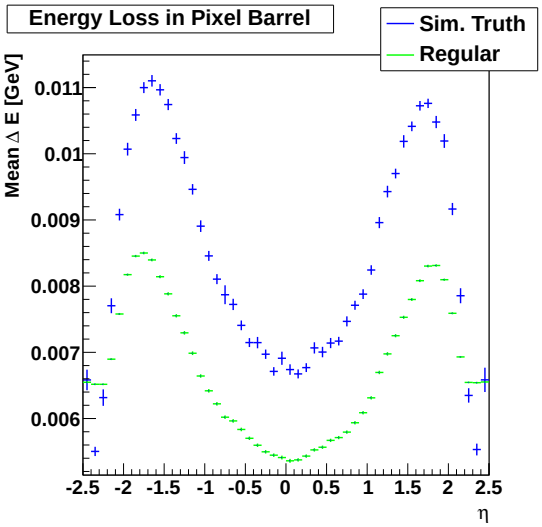


(a) Mean energy loss approximated by the parametrization method and the simulation truth.

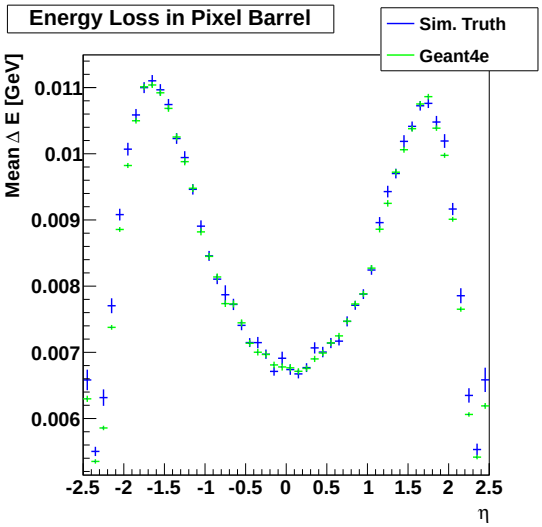


(b) Mean energy loss approximated by the Geant4e method and the simulation truth.

Figure C.2.: Comparison plots of the parametrization and Geant4e energy loss computations for endcap part of the detector.

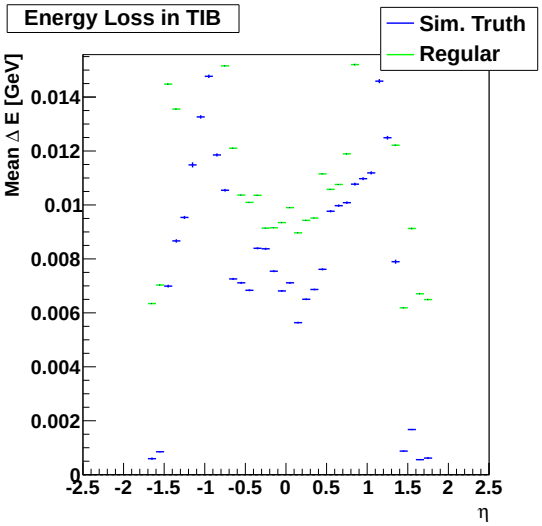


(a) Mean energy loss approximated by the parametrization method and the simulation truth.

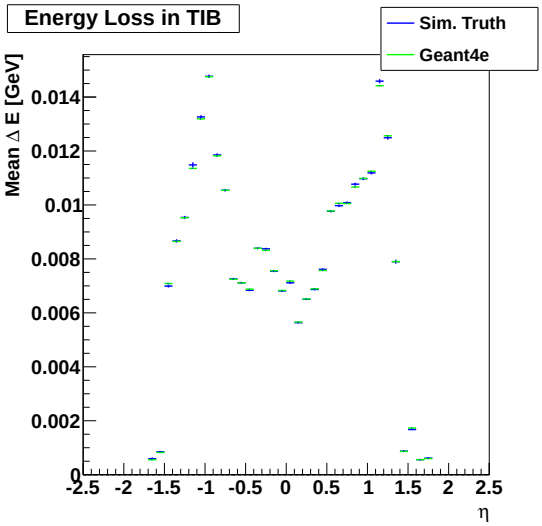


(b) Mean energy loss approximated by the Geant4e method and the simulation truth.

Figure C.3.: Comparison plots of the parametrization and Geant4e energy loss computations for the pixel barrel tracker.

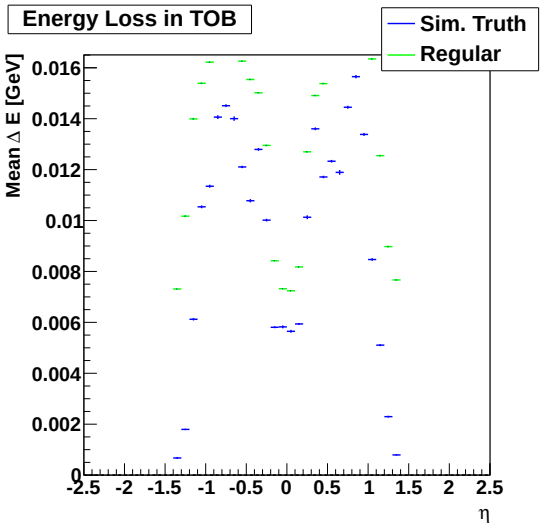


(a) Mean energy loss approximated by the parametrization method and the simulation truth.

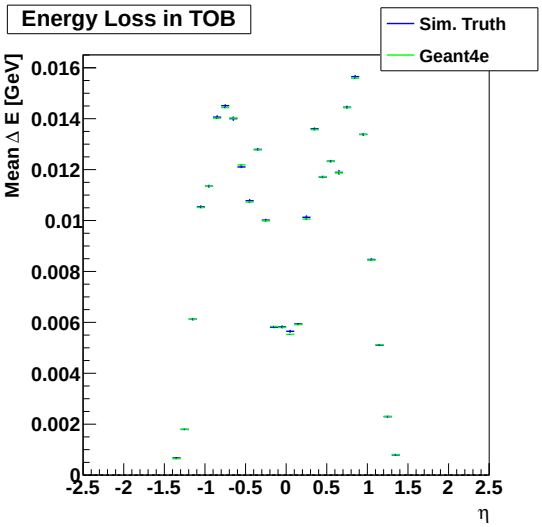


(b) Mean energy loss approximated by the Geant4e method and the simulation truth.

Figure C.4.: Comparison plots of the parametrization and Geant4e energy loss computations for the TIB tracker.

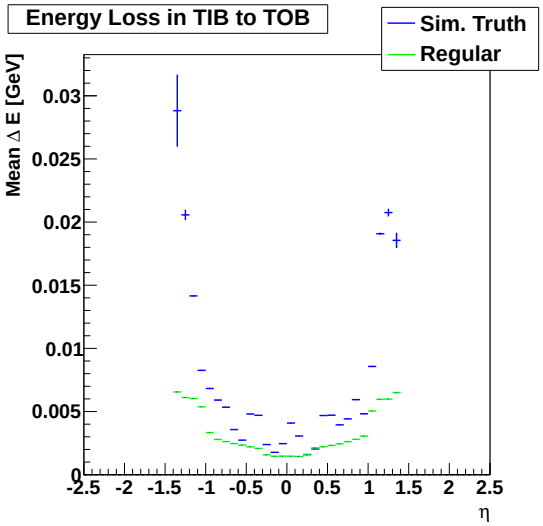


(a) Mean energy loss approximated by the parametrization method and the simulation truth.

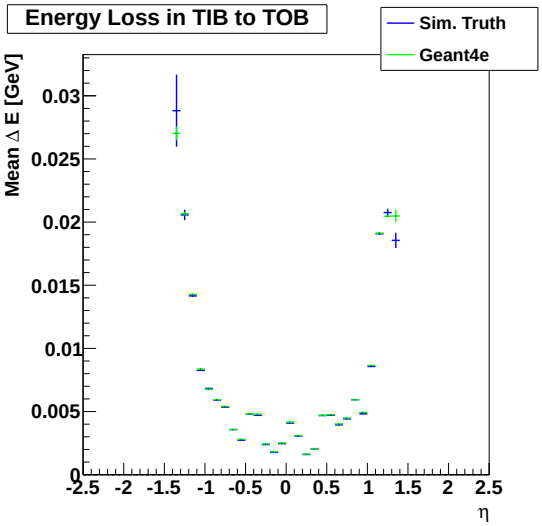


(b) Mean energy loss approximated by the Geant4e method and the simulation truth.

Figure C.5.: Comparison plots of the parametrization and Geant4e energy loss computations for the TOB tracker.



(a) Mean energy loss approximated by the parametrization method and the simulation truth.



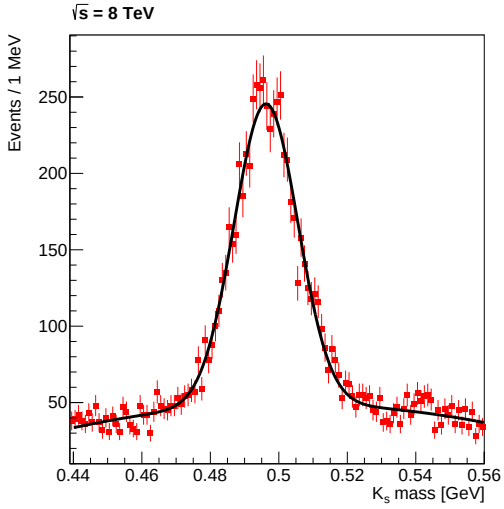
(b) Mean energy loss approximated by the Geant4e method and the simulation truth.

Figure C.6.: Comparison plots of the parametrization and Geant4e energy loss computations for the TIB to TOB transition.

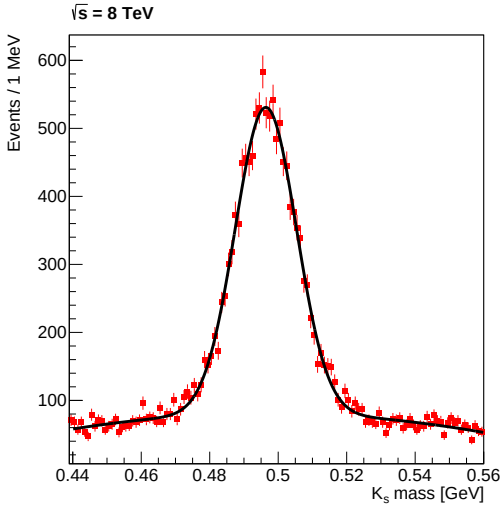
Appendix D

Kaon Mass Reconstruction

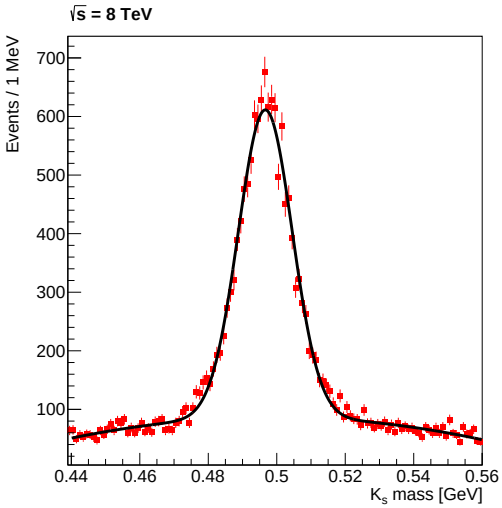
Distributions of the K_s^0 mass for a central barrel and endcap regions. All mass distributions in the figures below have been fitted with function 13.3, a gaussian function to match the signal peak and a quadratic polynomial to model the background.



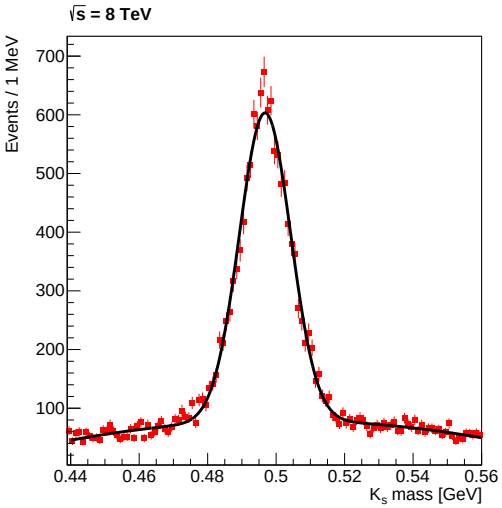
(a) Endcapl bin $-2.5 \leq \eta < -2.25$



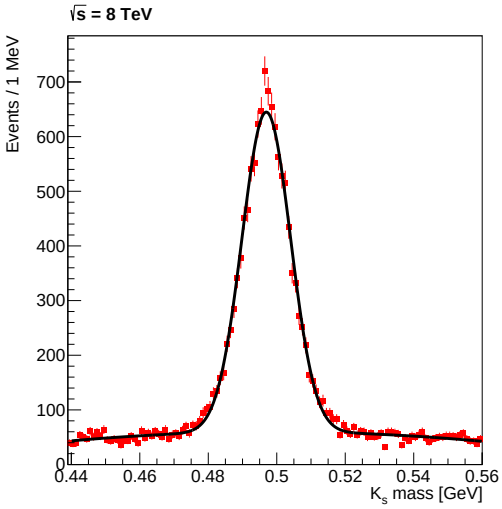
(b) Endcap bin $-2.25 \leq \eta < -2.0$



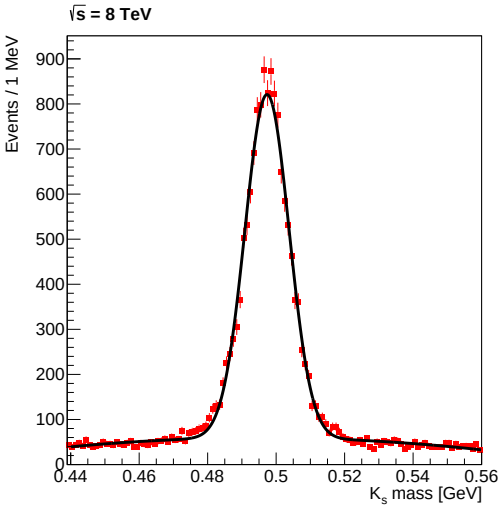
(c) Endcap bin $-2.0 \leq \eta < -1.75$



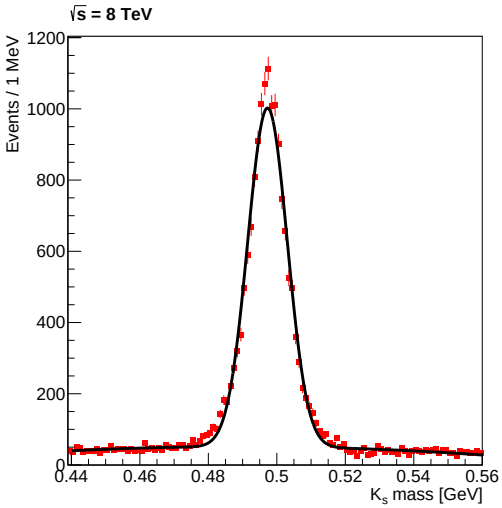
(d) Endcap bin $-1.75 \leq \eta < -1.5$



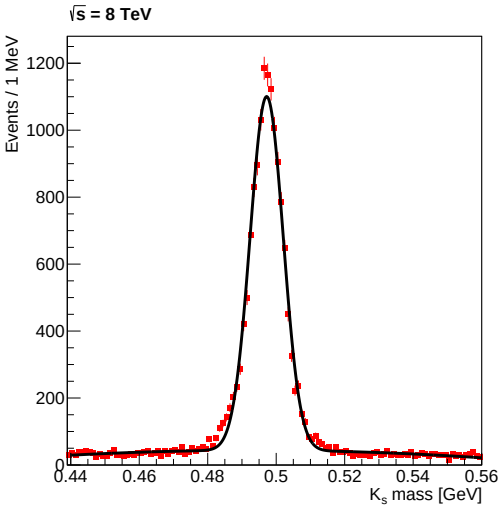
(e) Endcap bin $-1.5 \leq \eta < -1.25$



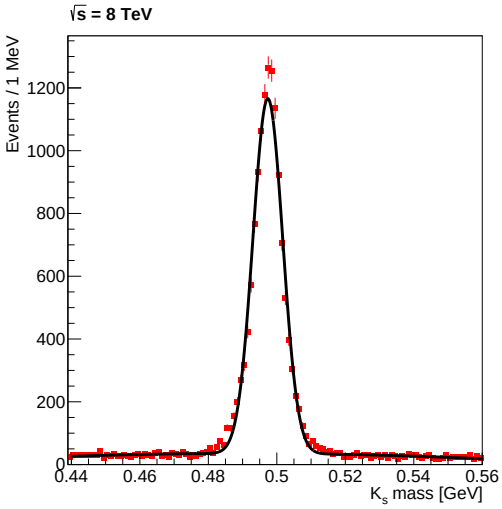
(f) Central barrel bin $-1.25 \leq \eta < -1.0$



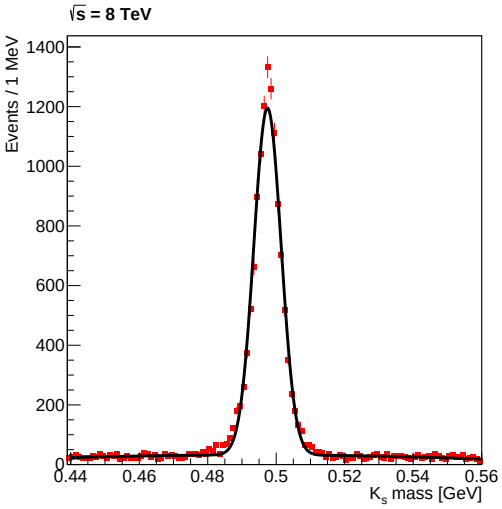
(g) Central barrel bin $-1.0 \leq \eta < -0.75$



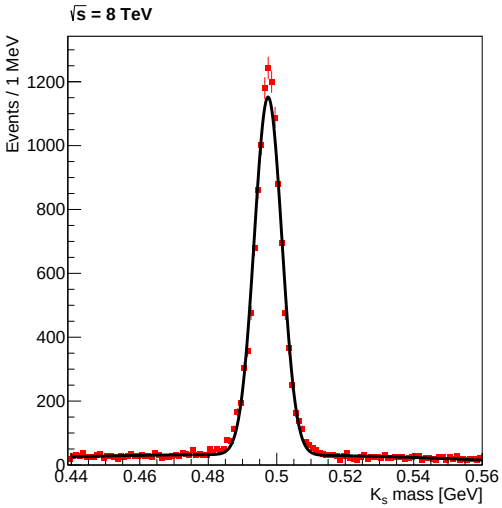
(h) Central barrel bin $-0.75 \leq \eta < -0.5$



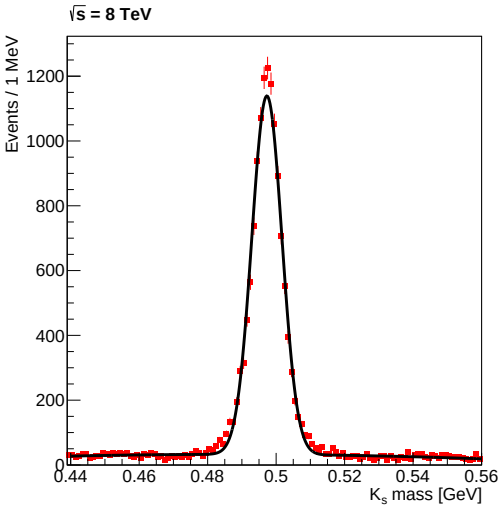
(i) Central barrel bin $-0.5 \leq \eta < -0.25$



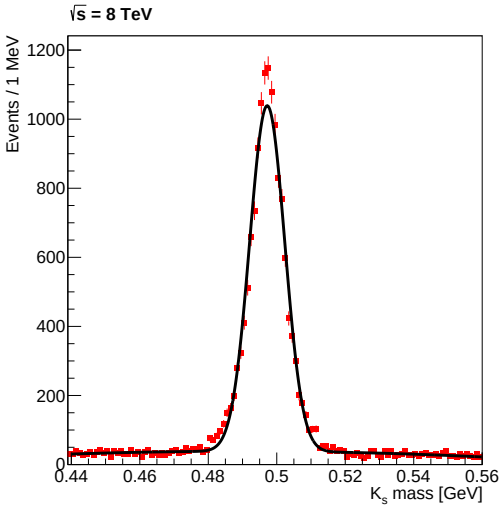
(j) Central barrel bin $-0.25 \leq \eta < 0.0$



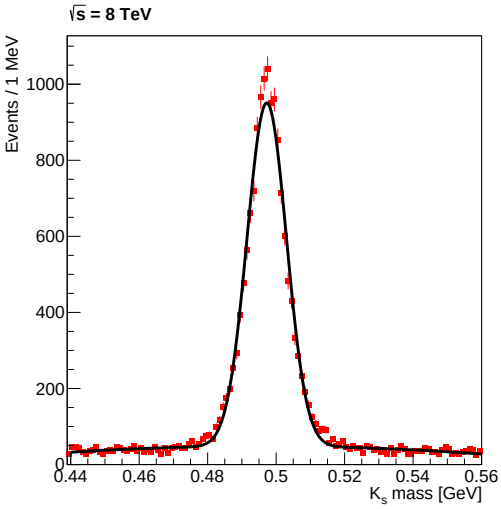
(k) Central barrel bin $0.0 \leq \eta < 0.25$



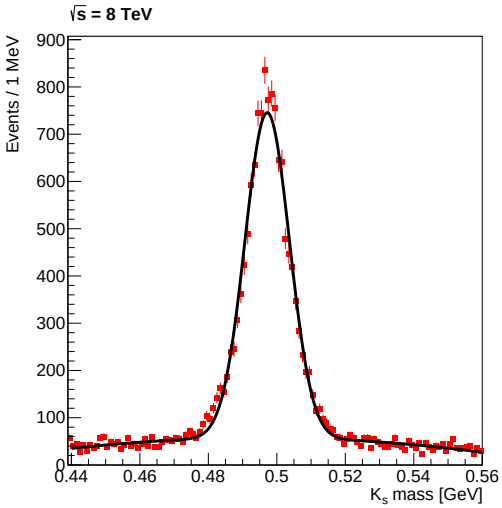
(l) Central barrel bin $0.25 \leq \eta < 0.5$



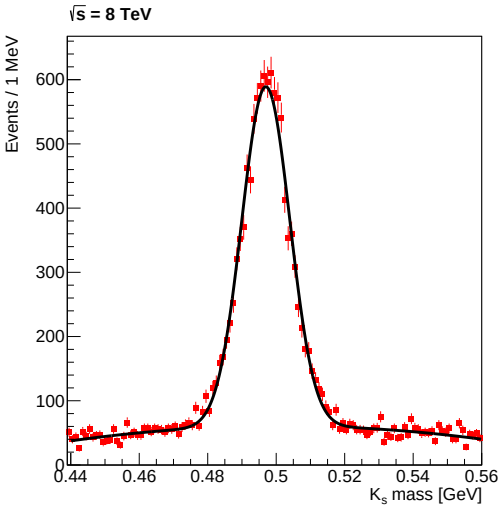
(m) Central barrel bin $0.5 \leq \eta < 0.75$



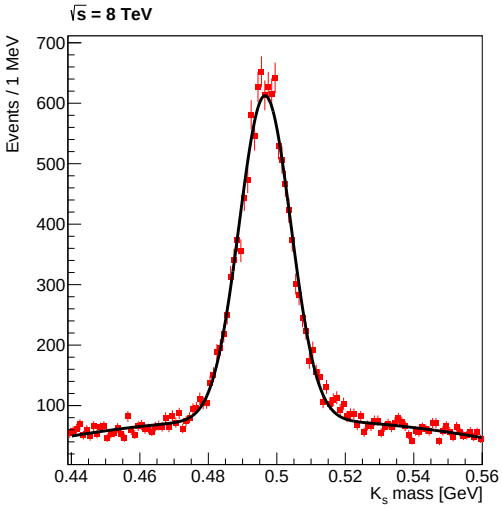
(n) Central barrel bin $0.75 \leq \eta < 1.0$



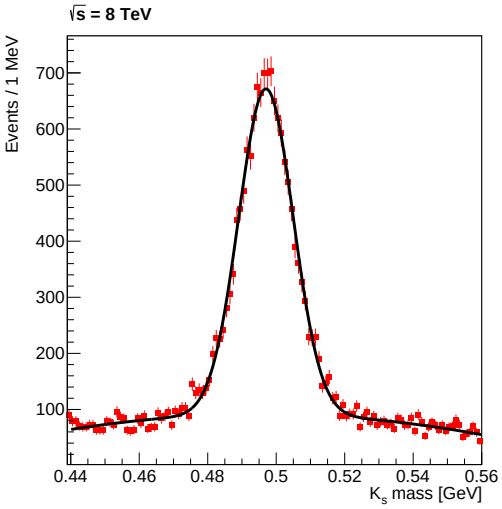
(o) Central barrel bin $1.0 \leq \eta < 1.25$



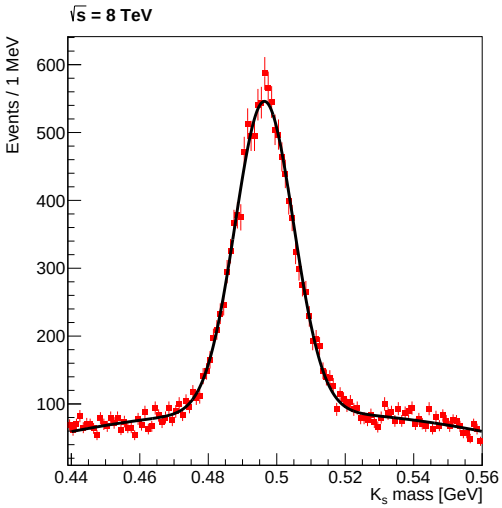
(p) Endcap bin $1.25 \leq \eta < 1.5$



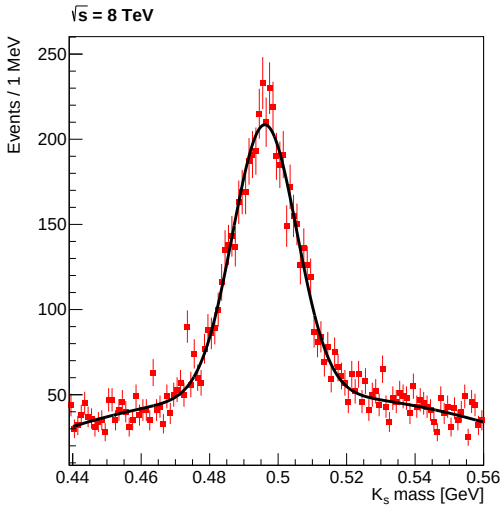
(q) Endcap bin $1.5 \leq \eta < 1.75$



(r) Endcap bin $1.75 \leq \eta < 2.0$



(s) Endcap bin $2.0 \leq \eta < 2.25$



(t) Endcap bin $2.25 \leq \eta < 2.5$

List of Figures

List of Tables

Bibliography

[1] Planck Collaboration, P. Ade *et al.*, "Planck 2013 results. I. Overview of products and scientific results", `arXiv:1303.5062 [astro-ph.CO]`.

[2] D. J. Griffiths, *Introduction to elementary particles.* John Wiley and Sons, Inc., 1987.

[3] T. S. Pettersson and P. Lefèvre, "The Large Hadron Collider: conceptual design", Tech. Rep. CERN-AC-95-05 LHC, CERN, Geneva, 1995.

[4] Particle Data Group, B. et al., "Review of particle physics", *Phys. Rev. D* 86 (Jul, 2012) 010001.

[5] ATLAS, CDF, CMS, and D0 Collaborations, "First combination of Tevatron and LHC measurements of the top-quark mass", *ArXiv e-prints* (Mar., 2014) , `arXiv:1403.4427 [hep-ex]`.

[6] Povh, Rith, Scholz, and Zetsche, *Teilchen und Kerne.* Springer, 2004.

[7] M. E. Peskin and D. V. Schroeder, *An Introduction to Quantum Field Theory (Frontiers in Physics).* Perseus Books, 2008.

[8] S. Catani *et al.*, "QCD", `arXiv:hep-ph/0005025`.

[9] N. J. Carron, *An Introduction to the Passage of Energetic Particles through Matter.* Taylor and Francis, Hoboken, NJ, 2007.

[10] D. E. Groom, N. V. Mokhov, and S. I. Striganov, "Muon stopping power and range tables 10 mev–100 tev", *Atomic Data and Nuclear Data Tables* 78 no. 2, (2001) 183–356.

[11] T. Hauth, "Dynamic Extension of Local Batch Systems with Cloud Resources and Measurement of the Jet Energy Scale with the CMS Detector", Master's thesis, Karlsruhe Institute of Technology (KIT), 2011.

IEKP-KA/2011-07
http://ekp-invenio.physik.uni-karlsruhe.de/record/45195
(19.12.2014).

[12] D. Piparo, *Statistical Combination of Higgs Decay Channels and Determination of the Jet-Energy Scale of the CMS Experiment at the LHC*. PhD thesis, Karlsruhe Institute of Technology (KIT), 2010.

[13] S. Holmes, R. S. Moore, and V. Shiltsev, "Overview of the Tevatron Collider Complex: Goals, Operations and Performance", *JINST* 6 (2011) T08001, arXiv:1106.0909 [physics.acc-ph].

[14] R. W. Assmann, M. Lamont, and S. Myers, "A Brief History of the LEP Collider", *Nucl. Phys. B, Proc. Suppl.* 109 no. CERN-SL-2002-009-OP, (Apr, 2002) 17–31. 15 p.

[15] ALEPH Collaboration, DELPHI Collaboration, L3 Collaboration, OPAL Collaboration, SLD Collaboration, LEP Electroweak Working Group, SLD Electroweak Group, SLD Heavy Flavour Group, S. Schael *et al.*, "Precision electroweak measurements on the Z resonance", *Phys.Rept.* 427 (2006) 257–454, arXiv:hep-ex/0509008 [hep-ex].

[16] P. W. Higgs, "Broken symmetries and the masses of gauge bosons", *Phys. Rev. Lett.* 13 (Oct, 1964) 508–509.

[17] F. Englert and R. Brout, "Broken symmetry and the mass of gauge vector mesons", *Phys. Rev. Lett.* 13 (Aug, 1964) 321–323.

[18] "Search for the Standard Model Higgs boson at LEP", *Physics Letters B* 565 no. 0, (2003) 61 – 75.

[19] CMS Collaboration, "Observation of a new boson with mass near 125 GeV in pp collisions at $\sqrt{s}$=7 and 8 TeV", *Journal of High Energy Physics* 2013 no. 6, (2013) .

[20] ATLAS Collaboration, "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC", *Physics Letters B* 716 no. 1, (2012) .

[21] LHCb Collaboration, B. Adeva *et al.*, "Roadmap for selected key measurements of LHCb", arXiv:0912.4179 [hep-ex].

[22] "CMS Public Lumi Results Webpage".
https://twiki.cern.ch/twiki/bin/view/CMSPublic/LumiPublicResults
(19.12.2014).

[23] A. Breskin and R. Voss, *The CERN Large Hadron Collider: Accelerator and Experiments*. CERN, Geneva, 2009.

[24] P. Skowronski *et al.*, "Optics Performance of the LHC during the 2012 Run", Tech. Rep. CERN-ACC-2013-0042, CERN, Geneva, May, 2013.

[25] "LHC Luminosity Plots for the 2012 Proton Run". `http://lpc.web.cern.ch/lpc/lumiplots_2012.htm` (19.12.2014).

[26] H. et al., "Optics Design and Lattice Optimisation for the HL-LHC", Tech. Rep. CERN-ACC-2013-0134, CERN, Geneva, May, 2013.

[27] CMS Collaboration, "CMS Physics: Technical Design Report Volume 2: Physics Performance",.

[28] CMS Media website. `http://cms.web.cern.ch/cms/Media/` (19.12.2014).

[29] T. C. Collaboration, "The cms experiment at the cern lhc", *Journal of Instrumentation* 3 no. 08, (2008) S08004.

[30] M. Lapka, "Interactive Slice of the CMS detector". `https://cms-docdb.cern.ch/cgi-bin/PublicDocDB/ShowDocument?docid=4172` (19.12.2014).

[31] R. Cittolin and Sphicas, *CMS trigger and data-acquisition project: Technical Design Report*. CERN, Geneva, 2002.

[32] CMS Collaboration, V. Karimaeki, M. Mannelli, P. Siegrist, H. Breuker, A. Caner, R. Castaldi, K. Freudenreich, G. Hall, R. Horisberger, M. Huhtinen, and A. Cattai, *The CMS tracker system project: Technical Design Report*. Technical Design Report CMS. CERN, Geneva, 1997.

[33] B. G. Streetman and S. K. Banerjee, *Solid state electronic devices; 6th ed.* Prentice-Hall, Englewood Cliffs, NJ, 2005.

[34] J. R. Chelikowsky and M. L. Cohen, "Electronic structure of silicon", *Phys. Rev. B* 10 (Dec, 1974) 5095–5107.

[35] F. Hartmann, *Evolution of Silicon Sensor Technology in Particle Physics*. Springer, 2009.

[36] Fruehwirth, Regler, Bock, Brote, and Notz, *Data Analysis Techniques for High-Energy Physics*. Cambridge University Press, 2nd ed. ed., 2000.

[37] CMS Collaboration, E. Migliore, "CMS Tracker alignment and material budget measurement", Tech. Rep. CMS-CR-2011-164, CERN, Geneva, 2011.

[38] CMS Collaboration, *CMS Physics: Technical Design Report Volume 1: Detector Performance and Software.* Technical Design Report CMS. CERN, Geneva, 2006.

[39] CMS Collaboration, S. Taroni, "Operation of the CMS silicon tracker", Tech. Rep. CMS-CR-2013-227. CERN-CMS-CR-2013-227, CERN, Geneva, Aug, 2013.

[40] C. Amsler *et al.*, "Mechanical Design and Material Budget of the CMS Barrel Pixel Detector", *Journal of Instrumentation 4* (2009) P05003.

[41] CMS Collaboration, "Studies of Tracker Material", Tech. Rep. CMS-PAS-TRK-10-003, 2010.

[42] D. Lange, "The CMS reconstruction software", Tech. Rep. CMS-CR-2011-002, CERN, Geneva, Jan, 2011.

[43] R. Brun and F. Rademakers, "ROOT - An Object Oriented Data Analysis Framework", *Proceedings AIHENP'96 Workshop* no. DOE-ER-40389-69, (1997) 81–86.

[44] T. Sjostrand, S. Mrenna, and P. Z. Skands, "PYTHIA 6.4 Physics and Manual", *JHEP* 0605 (2006) 026, `arXiv:hep-ph/0603175 [hep-ph]`.

[45] G. Corcella, I. G. Knowles, G. Marchesini, S. Moretti, K. Odagiri, P. Richardson, M. H. Seymour, and B. R. Webber, "HERWIG 6: an event generator for Hadron Emission Reactions With Interfering Gluons (including supersymmetric processes)", *J. High Energy Phys.* 01 (Nov, 2000) 93.

[46] M. L. Mangano, M. Moretti, F. Piccinini, R. Pittau, and A. Polosa, "ALPGEN, a generator for hard multiparton processes in hadronic collisions", *J. High Energy Phys.* 07 (Jun, 2002) 35.

[47] J. Alwall, P. Demin, S. de Visscher, R. Frederix, M. Herquet, F. Maltoni, T. Plehn, D. L. Rainwater, and T. Stelzer, "MadGraph/MadEvent v4: the new web generation", *Journal of High Energy Physics* no. 9, (2007) 28.

[48] T. Gleisberg, S. Hoeche, F. Krauss, M. Schonherr, S. Schumann, *et al.*, "Event generation with SHERPA 1.1", *JHEP* 0902 (2009) 007, `arXiv:0811.4622 [hep-ph]`.

[49] R. Field, "Early LHC Underlying Event Data - Findings and Surprises", `arXiv:1010.3558 [hep-ph]`.

[50] S. Agostinelli *et al.*, "G4 a simulation toolkit", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506 no. 3, (2003) 250–303.

[51] C. Rovelli, "The detailed simulation of the CMS detector", Tech. Rep. CMS-CR-2007-082, CERN, Geneva, Oct, 2007.

[52] J. A. et al., "Cms computing operations during run 1", *Journal of Physics: Conference Series* 513 no. 3, (2014) 032040.

[53] CMS WLCG Structure: Thanks to A. Scheurer for providing the picture.

[54] Grewal and Andrews, "Applications of Kalman Filtering in Aerospace 1960 to the Present", *Control Systems, IEEE* 30 (2010) .

[55] R. Frühwirth, "Application of Kalman filtering to track and vertex fitting", *Nuclear Instruments and Methods in Physics Research A* 262 (Dec., 1987) 444–450.

[56] CMS Collaboration, A. Tropiano, "Tracking and vertexing performance in CMS", Tech. Rep. CMS-CR-2012-384, CERN, Geneva, Dec, 2012.

[57] D. Giordano and G. Sguazzoni, "CMS reconstruction improvements for the tracking in large pile-up events", *Journal of Physics: Conference Series* 396 no. 2, (2012) 022044.

[58] E. Chabanat and N. Estre, "Deterministic annealing for vertex finding at CMS", *Proceedings of Computing in High Energy Physics and Nuclear Physics 2004* (2005) 287.

[59] M. Hildreth, "CMS Full Simulation: Evolution Toward the 14 TeV Run", *CHEP 2013 Conference* (2013) .
`http://indico.cern.ch/event/214784/session/3/contribution/159` (19.12.2014).

[60] T. Kuhr, "Computing at Belle II", *Journal of Physics: Conference Series* 331 no. 7, (2011) 072021.

[61] M. Bahr, S. Gieseke, M. Gigg, D. Grellscheid, K. Hamilton, *et al.*, "Herwig++ Physics and Manual", *Eur.Phys.J.* C58 (2008) 639–707, `arXiv:0803.0883 [hep-ph]`.

[62] C Debenedetti and the Atlas Collaboration, "Concepts for fast large scale Monte Carlo production for the ATLAS experiment", *Journal of Physics: Conference Series* 513 no. 2, (2014) 022006.

[63] B. Carpenter *et al.*, "The MUSCLE report: the computing needs of the LEP experiments", Tech. Rep. CERN-DD-88-1, CERN, Geneva, 1988.

[64] A. G. Holzner, "First Workshop on Data Preservation and Long Term Analysis in HEP - Data Conservation at LEP (ALEPH, DELPHI, L3)",. `http://indico.cern.ch/event/42722/session/3/contribution/12` (19.12.2014).

[65] S. Amerio, "CHEP2012 Session on Data Preservation and Long Term Analysis in HEP: Data Preservation at CDF",. `https://indico.cern.ch/event/171962/` (19.12.2014).

[66] T. Hara, "Fourth Workshop on Data Preservation and Long Term Analysis in HEP - Data Preservation at Belle/KEK",. `http://indico.cern.ch/event/95512/` (19.12.2014).

[67] "CMS Storage Overview - retrieved 23.5.2014". `http://transferteam.web.cern.ch/transferteam/StorageOverview/latest/main.html`.

[68] T. Abe, I. Adachi, K. Adamczyk, S. Ahn, H. Aihara, K. Akai, M. Aloi, L. Andricek, K. Aoki, Y. Arai, and et al., "Belle II Technical Design Report", *ArXiv e-prints* (Nov., 2010) , `arXiv:1011.0352 [physics.ins-det]`.

[69] D. Giordano and G. Sguazzoni, "CMS reconstruction improvements for the tracking in large pile-up events", *Journal of Physics: Conference Series* 396 no. 2, (2012) 022044.

[70] R. J. Langenberg, "Preparing the Track Reconstruction in ATLAS for a high multiplicity future", *Journal of Physics: Conference Series* 513 no. 5, (2014) 052027.

[71] O. S. Bruening, "HL-LHC parameter space and scenarios", *Proceedings of Chamonix 2012 workshop on LHC Performance* (2012) 10.

[72] S. Jarp, A. Lazzaro, and A. Nowak, "The future of commodity computing and many-core versus the interests of HEP software", *Journal of Physics: Conference Series* 396 no. 5, (2012) 052058.

[73] G. E. Moore *et al.*, "Cramming more components onto integrated circuits", *Electronics* 38 no. 8, (1965) 114–117.

[74] Intel, "Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor",. `ftp://download.intel.com/design/network/papers/30117401.pdf` (19.12.2014).

[75] M. J. Flynn, "Some Computer Organizations and Their Effectiveness", *Computers, IEEE Transactions on* C-21 no. 9, (Sept., 1972) 948 –960.

[76] "IEEE Standard for Floating-Point Arithmetic", tech. rep., Microprocessor Standards Committee of the IEEE Computer Society, 3 Park Avenue, New York, NY 10016-5997, USA, Aug., 2008.

[77] M. Clemencic, B. Hegner, P. Mato, and D. Piparo, "Introducing concurrency in the gaudi data processing framework", *Journal of Physics: Conference Series* 513 no. 2, (2014) 022013.

[78] T. Hauth, V. Innocente, , and D. Piparo, "Development and Evaluation of Vectorised and Multi-Core Event Reconstruction Algorithms within the CMS Software Framework", *Journal of Physics: Conference Series* 396 no. 5, (2012) 052065.

[79] Nvidia, "Kepler GK110 Whitepaper",. `http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf` (19.12.2014).

[80] Nvidia, "Tesla K40 GPU Active Accelerator - Board Specification",. `http://www.nvidia.com/content/PDF/kepler/Tesla-K40-Active-Board-Spec-BD-06949-001_v03.pdf` (19.12.2014).

[81] D. Funke, "Parallel Triplet Finding for Particle Track Reconstruction", Master's thesis, Karlsruhe Institute of Technology (KIT), 2013. EKP-2013-00027.

[82] "The GNU Compiler Collection". `http://gcc.gnu.org/` (19.12.2014).

[83] "Libm". `http://sourceware.org/newlib/libm.html` (19.12.2014).

[84] "Intel Math Kernel Library website". `https://software.intel.com/en-us/intel-mkl` (19.12.2014).

[85] "ACML - AMD Core Math Library". `http://developer.amd.com/tools-and-sdks/cpu-development/amd-core-math-library-acml/` (19.12.2014).

[86] S. Moshier, "Cephes C and C++ language special functions math library". `http://www.moshier.net/#Cephes` (19.12.2014).

[87] "LGPL". `https://www.gnu.org/licenses/lgpl.html` (19.12.2014).

[88] "vdt website". `https://svnweb.cern.ch/trac/vdt` (19.12.2014).

[89] D. Piparo, V. Innocente, and T. Hauth, "Speeding up hep experiment software with a library of fast and auto-vectorisable mathematical functions", *Journal of Physics: Conference Series* 513 no. 5, (2014) 052027.

[90] "Geant4 10.0 Release Notes", tech. rep., Dec, 2013. `http://geant4.web.cern.ch/geant4/support/ReleaseNotes4.10.0.html` (19.12.2014).

[91] "cuBLAS". `https://developer.nvidia.com/cuBLAS` (19.12.2014).

[92] Y. L. Dokshitzer, G. Leder, S. Moretti, and B. Webber, "Better jet clustering algorithms", *JHEP* 9708 (1997) 001, `arXiv:hep-ph/9707323 [hep-ph]`.

[93] S. Gorbunov, U. Kebschull, I. Kisel, V. Lindenstruth, and W. F. J. Muller, "Fast simdized kalman filter based track fit", *Comp. Phys. Commun* 178 (2008) 374–383.

[94] D. Funke, T. Hauth, V. Innocente, G. Quast, P. Sanders, and D. Schieferdecker, "Parallel track reconstruction in CMS using the cellular automaton approach", *Journal of Physics: Conference Series* 513 no. 5, (2014) 052010.

[95] R. Fruehwirth, A. Strandlie, W. Waltenberger, and J. Wroldsen, "A review of fast circle and helix fitting", *Nucl. Instrum. Methods Phys. Res., A* 502 (2003) 705–707.

[96] J. Mattmann and C. Schmitt, "Track finding in ATLAS using GPUs", *Journal of Physics: Conference Series* 396 no. 2, (2012) 022035.

[97] CMS Collaboration, "Measurement of Momentum Scale and Resolution of the CMS Detector using Low-mass Resonances and Cosmic Ray Muons", Tech. Rep. CMS-PAS-TRK-10-004, CERN, 2010. Geneva, 2010.

[98] P. Arco, "Error propagation for track reconstruction inside the geant4 framework", *CHEP 2006 Conference* . `https://indico.cern.ch/event/048/session/3/contribution/85/material/paper/0.pdf` (19.12.2014).

[99] T. Hauth, "Geant4 Bugreport: Memory leak in G4ErrorPropagator",. `http://bugzilla-geant4.kek.jp/show_bug.cgi?id=1466` (19.12.2014).

[100] "CMSSW V0 Producer website". `https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideV0Producer` (19.12.2014).

[101] R. Field, "QCD@LHC 2011 Workshop: CMS MC Tunning effort",. https://conference.ippp.dur.ac.uk/contributionDisplay.py?contribId=31&sessionId=15&confId=311 (19.12.2014).

Acknowledgements

First of all, I want to thank Professor Günter Quast for giving me the opportunity to work as part of his team and supervising me during my stay at CERN. The thesis presented here would not have been possible without his continuous support and advice. He managed to form a working group with a pleasant and supportive culture and allowed me to develop new ideas and provided guidance if needed.

Alike, I sincerely thank Professor Michael Feindt for being the co-referee of this thesis and providing me with valuable comments on my work and the thesis.

I want to thank my CERN supervisor Dr. Andreas Pfeiffer for hosting my stay at CERN and giving me the opportunity to present my work at international conferences.

A special thanks to Dr. Danilo Piparo, Dr. Vincenzo Innocente and Dr. Benedikt Hegner for our close collaboration on software optimization project.

I want to deeply thank the members of the working group: Dr. Klaus Rabbertz for his help in preparing talks, Dr. Fred-Markus Stober for his valuable programming advice, Georg Sieber for proofreading of this thesis, Daniel Funke for working on the GPU implementations with me and the rest of the group for providing such a pleasant and friendly environment for me to work in.

I want to thank Prof. Dr. Thomas Müller for managing the Institute of Experimental Nuclear Physics and Bärbel Bräunling for her kindness and helpfulness whenever administrative work had to be done. I want to thank the Institute for supporting my work and the Wolfgang-Gentner-Programme for providing me the opportunity to live in Geneva and work at CERN for three years.

I also want to thank the administrators of the local computing infrastructure and of the National Analysis Facility (NAF) at DESY in Hamburg.

But most of all, I want to deeply thank my family for their support and encouragement during my studies.

Hiermit versichere ich, die vorliegende Arbeit selbtständig verfasst und nur die angegebenen Hilfsmittel verwendet zu haben.

Thomas Hauth
Karlsruhe, den 20. 1. 2015