CERN-THESIS-2009-259

Abstract

The field of high energy physics is on the verge of several remarkable scientific breakthroughs due to the imminent startup of the Large Hardron Collider (LHC) at CERN. This machine will provide proton collisions at 14 TeV – an energy scale that no other man–made accelerator could reach so far and thus allowing a deeper insight into the smallest particles of which matter is composed as well as better understanding of the creation of the universe itself.

The energy scale of the accelerator is by far not the only unique thing about this extraordinary machine. The data that will be produced by the LHC amounts to 15 petabytes (15 million gigabytes) per year. This posed and still poses a challenge not only for all software used to reconstruct the collisions (events), be it the trigger, that will decide if an event is kept for further reconstruction or discarded, or the reconstruction algorithms themselves but also for the storage of the accumulated data.

This work starts giving a general overview over CERN and the LHC collider, zooming then inwards, introducing one of the LHC experiments, namely the CMS detector, in greater detail. Moving in further, the trigger software handling the decision about whether to keep events or not is presented. Finally the core of the work, the presentation and performance studies of a vertex reconstruction algorithm, the Multi Vertex Fitter (MVF) for the CMS inner tracker is introduced. Implemented optimizations in this algorithm are presented, their effects on the performance of the algorithm are discussed and compared to another already well–tested reconstruction algorithm, the Adaptive Vertex Reconstructor (AVR).

Alltough a considerable increase in the performance of the MVF algorithm could be obtained it will be shown that this algorithm cannot be better than the already very well–working AVR. Nevertheless, the MVF delivers an output that is at least in some settings comparable to the AVR.

Kurzfassung

Die bevorstehende Inbetriebnahme des bislang größten Teilchenbeschleunigers, des Large Hadron Collider (LHC), wird zu wesentlichen Erkenntnissen im Gebiet der Hochenergiephysik führen. Diese einzigartige Maschine wird Protonen mit einer Energie von je 7 TeV zur Kollision bringen und bewegt sich damit in einem Energieniveau, das kein von Menschenhand gebauter Beschleuniger je erreicht hat. Dadurch wird nicht nur ein tieferer Einblick in den Aufbau der Materie möglich, sondern es werden auch wesentliche Informationen über die Entstehung des Universums zugänglich sein.

Neben den nie dagewesenen Energien, die im LHC erzeugt werden, war und ist auch die Verarbeitung und Speicherung der anfallenden Daten eine Herausforderung an die beteiligten Wissenschaftler. Das von LHC erzeugte Datenvolumen beläuft sich auf 15 Petabyte (15 Millionen Gigabyte) pro Jahr. Sowohl die zur Auswahl interessanter Kollisionen (Events) verwendeten Algorithmen als auch die verwendeten Rekonstruktionsalgorithmen müssen enorm schnell und hoch effektiv sein um die für die Physik interessanten Events zu gewinnen.

Die vorliegende Arbeit gibt als Einleitung einen Überblick über den Beschleuniger selbst und beschreibt dann den Aufbau und die verwendeten Detektortechnologien eines der vier großen Experimente – des Compact Muon Solenoid (CMS) Experiments. Darauf folgt ein kurzer Überblick über die Triggersoftware, die die Entscheidung trifft, ob ein Event zur weiteren Rekonstruktion behalten wird oder nicht. Nach einer kleinen, allgemeinen Einführung in das Gebiet der Statistik wird dann der Kern der Arbeit, die Präsentation und schrittweise Optimierung eines Algorithmus (des Multi Vertex Fitters – MVF) zur Rekonstruktion von Vertices, erreicht. Die Ergebnisse werden mit denen eines bereits gut funktionierenden Rekonstruktionsalgorithmus, des Adaptive Vertex Reconstructors (AVR), verglichen und auf dieser Basis evaluiert.

Contents

Chapter 1

Introduction

1.1 The Large Hadron Collider (LHC)

At the European Nuclear Research Center (Conseil Européen pour la Recherche Nucléaire — CERN) in Geneva, Switzerland, the probably most important particle accelerator has been constructed — the Large Hadron Collider, which has gone in operation for the first time in September 2008. It is going to produce data in an energy region exceeding the highest energy level reached by any other man-made accelerator so far. This introduction will give a short insight into the requirements and efforts made in order to make this extraordinary machine possible. The interested reader is referred to [5, 15, 26] or [27] for more details.

The LHC reuses the tunnel of the former LEP (Large Electron Positron) collider, located 50 – 175 m underground. It has a circumference of 27 km spanning nearly from Mount Jura (France) to Lac Leman (Switzerland) (Fig. 1.1).

Unlike LEP, where electrons and positrons were accelerated, LHC uses two counter rotating beams of protons or, alternatively, lead ions. The protons are accelerated to an energy of 7 TeV per beam, resulting in a center of mass energy of 14 TeV (lead ions: 1150 TeV at center of mass). At this energy the protons are traveling with 99.999999 % of the speed of light. In order to achieve this velocity, the protons are preaccelerated in the already existing preacceleration chain of CERN (Fig. 1.3), discussed in the following subsection. All the information presented in the following subsection are taken from [35].

Figure 1.1: Location of the LHC (Source: `http://www.phys.ufl.edu/~matchev/LHCJC/lhc.html`)

1.1.1 The accelerator chain - from source to collision

The protons are generated using a dual plasma source [35] (Fig. 1.2). The source ionizes hydrogen, leaving a proton plasma which is then ejected with 90 kV into the RFQ (radio frequency quadrupole).

The RFQ is basically a linear collider of 1.75 m length, that focuses the protons (using electrical quadrupoles) and accelerates them to an energy of 750 keV. The protons are then injected into the LINAC2, a linear accelerator (30 m length), after which they reach the first circular accelerator, the Proton Synchrotron Booster (PSB), at an energy of 50 MeV. After being accelerated to 1.4 GeV, they are injected into the Proton Synchrotron (PS) and then (at 25 GeV) into the SPS (Super Proton Synchrotron). The SPS has a circumference of 6912 m and is able to accelerate the protons to an energy of 450 GeV. Via two transfer lines, the protons are finally fed into the LHC. For further details on the accelerator chain see Fig. 1.3 and [35].

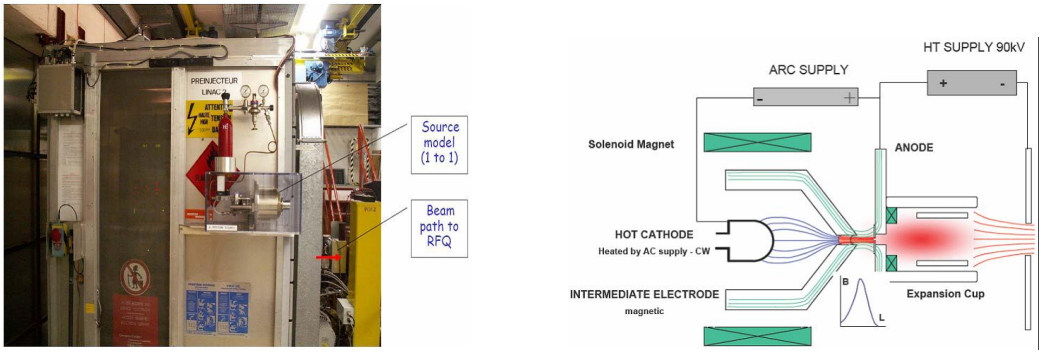


Figure 1.2: The dual plasma source (left) and a schematic drawing of it (right). (Source: [35])

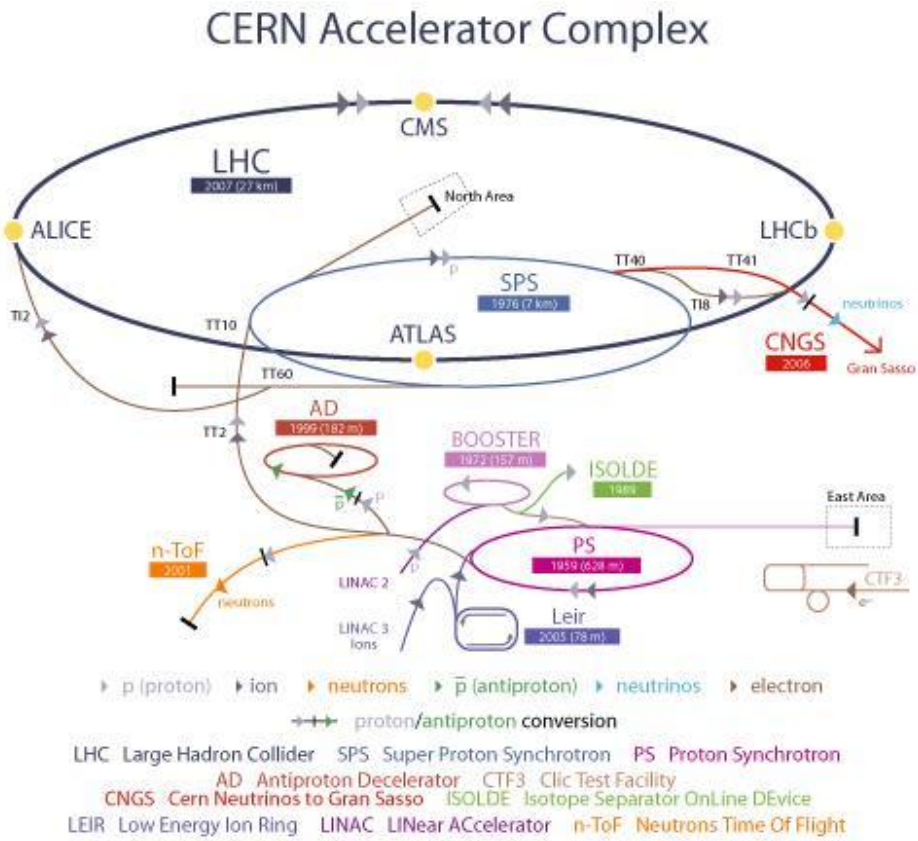


Figure 1.3: The accelerator complex at CERN (Source: `http://public.web.cern.ch/Public/en/Research/AccelComplex-en.html`)

1.1.2 The Beam

At the LEP accelerator a single beam tube for the electrons and positrons was sufficient, since these particles are of equal mass and opposite charge. The LHC has to use two separate beam tubes, one for each proton beam (Fig. 1.4).

Figure 1.4: An LHC dipole with two beam pipes (Source: `http://mediaarchive.cern.ch/MediaArchive/Photo/Public/1998/9809007/9809007/9809007-Icon.jpg`)

Electrons are much smaller than protons and can be regarded as pointlike particles, so electron/electron (as well as electron/positron) collisions always are head-on collisions. Protons on the other hand do have an intrinsic structure, so it is not granted that every collision is a hard (head-on) one. The direct consequence is that not every collision is interesting for physics analysis — the relevant ones have to be filtered out. To increase the probability for a head-on collision, the beam is focused to 16 μm at the interaction point (IP).

The main reason for the LHC using protons is that the effect of synchrotron radiation is inversely dependent on the mass ($\propto \mathrm{m}^{-4}$). Due to their small mass, synchrotron radiation is the main effect for energy loss of electrons. The center of mass energy of the LHC would be impossible to reach by accelerating electrons in a circular collider. For protons, the energy loss due to synchrotron radiation in a ring is smaller by a factor of $\mathcal{O}(10^{13})$. Therefore protons can be accelerated to far higher energies.

To keep the beam focused and bent, 1104 superconducting dipole magnets, operating at a temperature of 1.9 K and providing a field of 8.4 T, as well as 736 quadrupole magnets are used along the beam line.

When referring to a beam of protons, one has to keep in mind that the beam is not continuous. It consists of 2808 proton bunches, spaced with

25 ns. Each bunch contains the amount of $1.15 * 10^{11}$ protons. The bunches cross at the IP at a rate of $4 * 10^7$ times per second. Since not all bunches are filled, an effective event rate of 32 MHz is obtained.

Those facts were a real challenge since not only new solutions for treating such an enormous amount of data had to be found, but also the design of a detector fit for working under such harsh conditions as well as materials chosen for detector and electronics were no simple matter. The next section and chapters will illustrate how these challenges were met.

1.2 The CMS Experiment

The goal of collider experiments is to measure the energies and momenta of the particles created in the collisions. These quantities have to be extracted from the detector data e. g. the momenta of charged particles are computed from the curvature of the particle tracks in a magnetic field. Since this curvature becomes smaller with higher energy, it is crucial to measure the tracks as accurately as possible. Huge detectors, mostly in cylindrical form, are needed to measure all the created particles in sufficient detail. The CMS Detector (Fig. 1.5) described here has such a cylindrical form, with a length of 22 m and a diameter of 15 m. It weighs approximately 12 500 t. In the following subsections the subdetectors (inner tracker, calorimeters, solenoid, muon chambers) of the CMS Experiment and the technologies used for them are presented. In [8, 9] the CMS experiment is discussed in greater detail.

1.2.1 The Inner Tracker

The inner tracker is closest to the IP. Its purpose is to detect all the charged particle tracks originating at the primary and all secondary vertices in its volume with the highest possible resolution.

The CMS inner tracker is 5.8 m long and has a diameter of 2.5 m. The detector consists of several layers of high resolution semiconductive silicon detectors. They are radiation hard, and since they are based on the same production process as microchips, the detectors are quite easy and cheap to produce compared to other detector technologies used in CMS.

The innermost three layers of the CMS inner tracker barrel consist of pixel detectors (Fig. 1.7). The next four layers form the inner barrel with two double-sided and two single-sided silicon strip detectors, and the outermost six layers form the outer barrel, with again two double-sided and four single-sided silicon strip detectors. The endcaps consist of two discs with pixel detectors and ten discs with silicon strip detectors (Fig. 1.6).

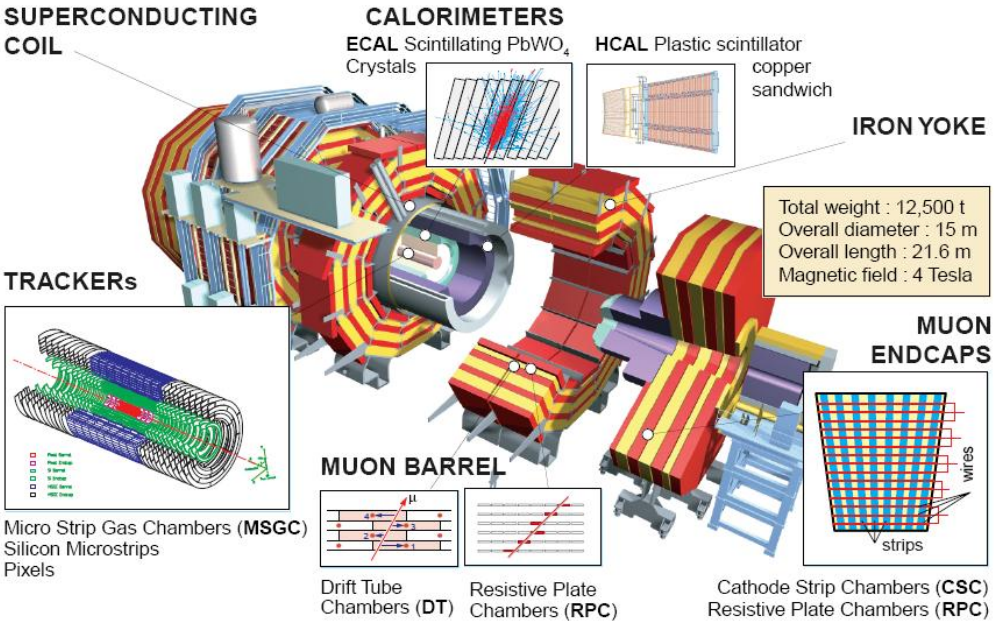


Figure 1.5: Overview of the CMS experiment (Source: `http://esmane.physics.lsa.umich.edu/wl/umich/phys/um-cern-reu/2004/20040805-umwlap002-08-wagner/real/sld003.htm`)

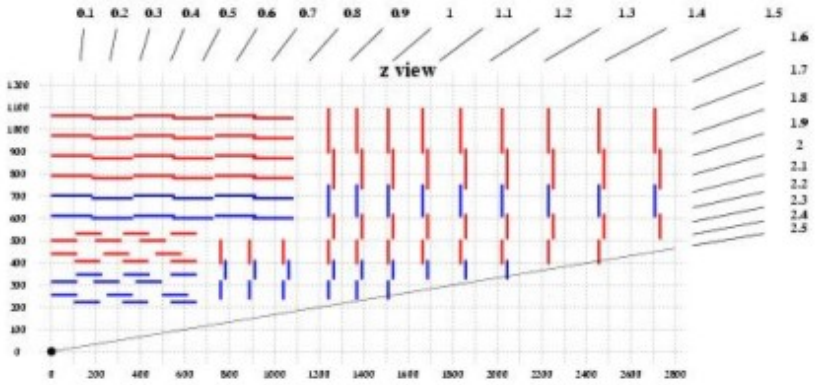


Figure 1.6: Cross section of one inner tracker barrel quarter and an endcap half. Double-sided silicon strip detectors are depicted in blue, the single-sided ones in red. (Source: `http://www.ba.infn.it/~zito/cms/tvis.html`)

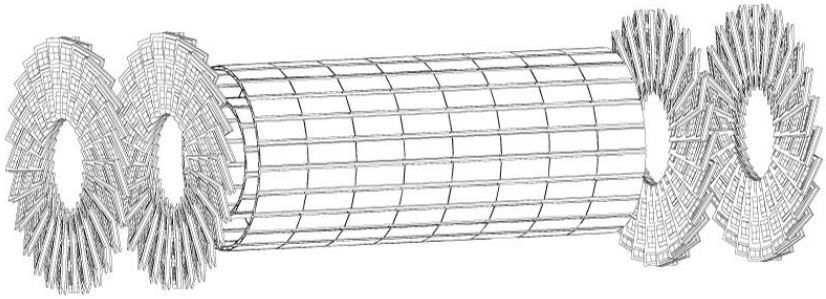


Figure 1.7: The CMS pixel detector. (Source: `http://cmsinfo.cern.ch/outreach`)

With a total of 1440 pixel detector modules and 15148 strip detector modules ($\cong$ 200 m^2 of active silicon) it is the largest silicon tracker ever built.

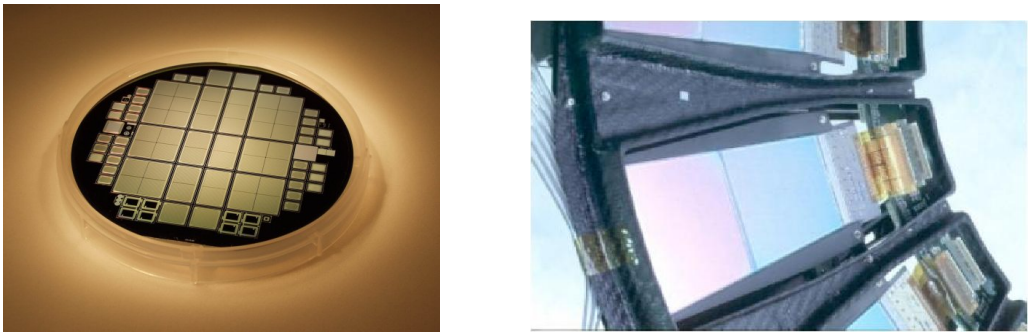


Figure 1.8: A single pixel detector (left) and a closeup of a silicon strip detector from an endcap disc (right). (Sources: `http://cmsinfo.cern.ch/outreach` and [23])

1.2.2 The Calorimetry

In the calorimeters the energy of the incoming particles is measured by complete absorption. In a high energy physics experiment, two different types of calorimeters are used — the electromagnetic and the hadronic calorimeter.

The Electromagnetic calorimeter

The electromagnetic calorimeter (ECAL) detects photons as well as electrons and positrons. When these particles enter the calorimeter, they create an

electromagnetic cascade or shower (Fig. 1.9). Electrons (and positrons) that enter the dense material of the calorimeter emit bremsstrahlungsphotons. These photons can then create electron/positron pairs if they have enough energy (more than 1.022 MeV) which will again emit bremsstrahlungsphotons. The shower stops if the energy of the photons falls below the threshold for pair production. A photon that enters the calorimeter immediately starts the showering with pair production.

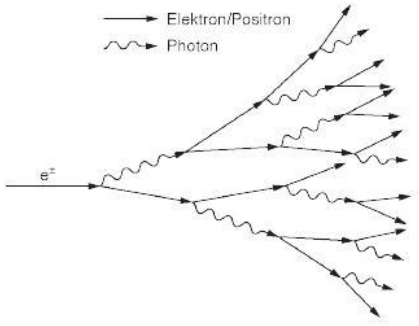
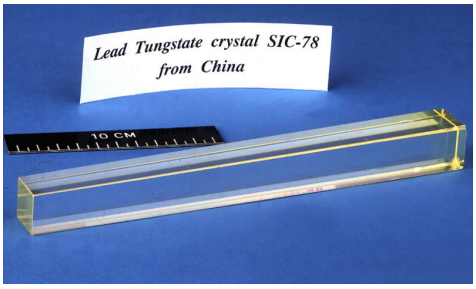


Figure 1.9: Schematic drawing of an electromagnetic shower. (Source: [24])

In CMS the ECAL is made of 80 000 $PbWO_4$ monocrystals (Fig. 1.10). These scintillating crystals have a short radiation length (0.89 cm), a small molière radius (2.2 cm) and a high density (8.28 g/cm^3) making the calorimeter very compact. Furthermore $PbWO_4$ is very radiation hard although the crystals get opaque when exposed to radiation. The reason for this are color centers forming inside the crystals which absorb a part of the transmitted light. At a temperature of 18°C[1] these centers anneal again establishing an equilibrium in optical transmission depending on the dose rate. The change in optical transmission will be 1 to 2 % in the barrel at low luminosity and a few 10 % in the endcaps at high luminosity. To account for the performance loss due to the behavior of the crystals, the optical properties of the crystals are monitored via a light-injection system. This system sends laser pulses via optical fibers into the crystals and measures their optical transmittance.

[1]This is the working temperature of the calorimeter.



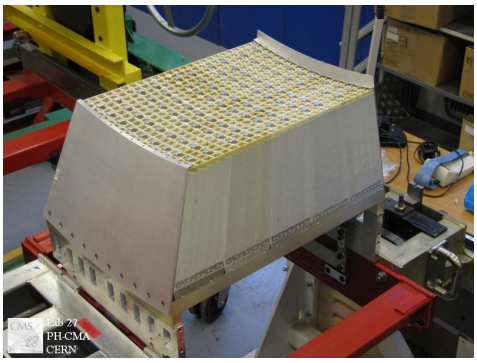


Figure 1.10: A single $PbWO_4$ crystal (left) and a module of crystals (right). (Sources: `http://cmsinfo.cern.ch/outreach` and `http://doc.cern.ch//archive/electronic/cern/others/PHO/photo-cms/oreach//oreach-2005-003_08.jpg`)

Figure 1.11: Crystal modules in the detector. (Source: `http://doc.cern.ch//archive/electronic/cern/others/PHO/photo-cms/oreach/oreach-2006-019.jpg`)

The Hadronic calorimeter

The second calorimeter is the hadronic calorimeter (HCAL). It absorbs the hadrons created in the collisions by creating a hadron shower. This shower is much more complex than its electromagnetic equivalent, because more types of interactions are occurring. Since exploiting these in greater detail would exceed the scope of this thesis the interested reader is referred e. g. to [24] for more information.

The CMS detector uses a sampling calorimeter (Fig. 1.12 and 1.13). In this type of calorimeter the absorbing and the active detector layers are arranged alternately. The absorbers are solid copper (thickness of each absorber: 50 mm) and the active detector layers are made of scintillators (thickness of each scintillator: 4 mm).

The HCAL needs to be thick enough to contain the entire hadronic shower. The design of CMS is such that both calorimeters are fitted inside a solenoid magnet (see next section). Since copper has a very high density, most of the showers are contained in the calorimeter although its width is limited by the size of the magnet. Nevertheless, especially in the central region of the CMS detector the thickness of both calorimeters together is not enough to stop the particles. Therefore an outer (outside the solenoid) hadronic calorimeter exists. This outer detector is a scintillator embedded in the first layer of the return yoke using the solenoid as absorber.

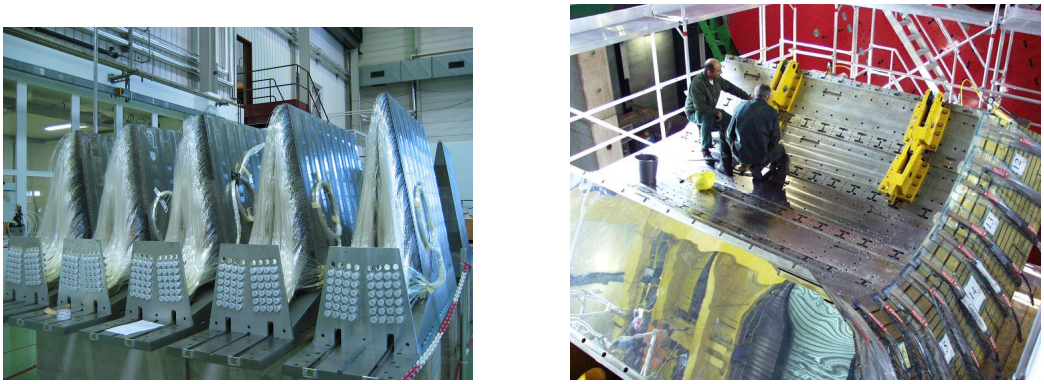


Figure 1.12: The single wedges of the HCAL (left) and the wedges in the HCAL barrel (right). (Source: `http://doc.cern.ch//archive/electronic/cern/others/PHO/photo-cms/hcal/hcal-2003-009.jpg` and `http://doc.cern.ch//archive/electronic/cern/others/PHO/photo-cms/hcal/hcal-2002-007.jpg`)

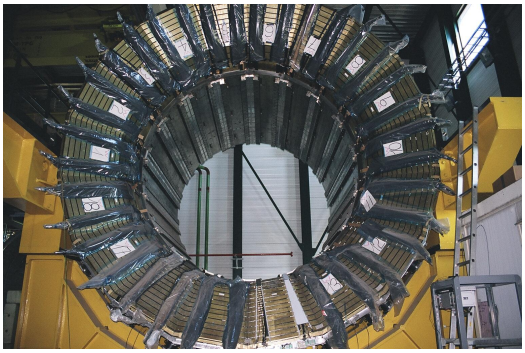


Figure 1.13: Whole HCal barrel. (Source: `http://doc.cern.ch//archive/electronic/cern/others/PHO/photo-cms/hcal/hcal-2005-001.jpg`)

1.2.3 The Solenoid

As said before, one of the most important characteristics of a particle is its momentum. In order to measure it, a strong magnetic field is crucial. In CMS the magnetic field is provided by a gigantic superconducting solenoid (Fig. 1.14 (left)). It is the largest superconducting solenoid ever built, with a length of 12.5 m, a diameter of 6 m and a weight of 220 t. It is cooled with liquid helium to 3 – 4 K and provides a magnetic field of 4 T in its center. The solenoid is supported by the middle ring of the magnet (or return) yoke (Fig. 1.14 (right)). The yoke consists of five iron rings (Fig. 1.15) and two endcaps with three discs each. It has a mass of 10 000 t.

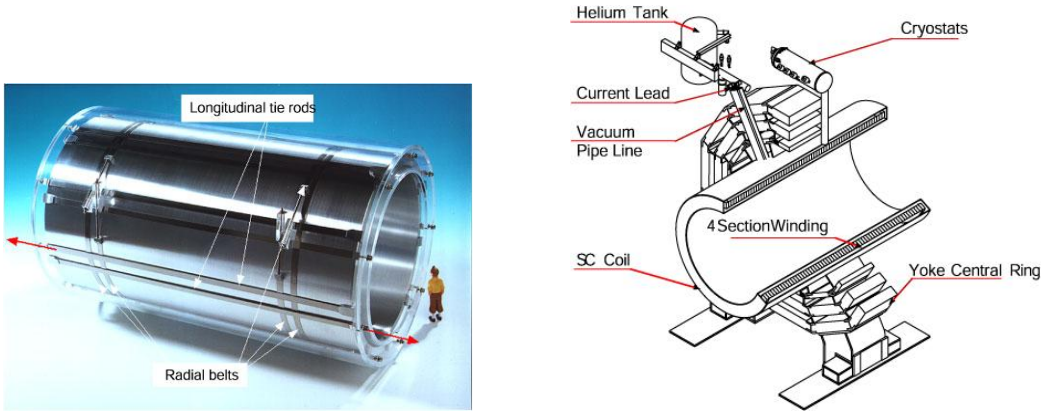


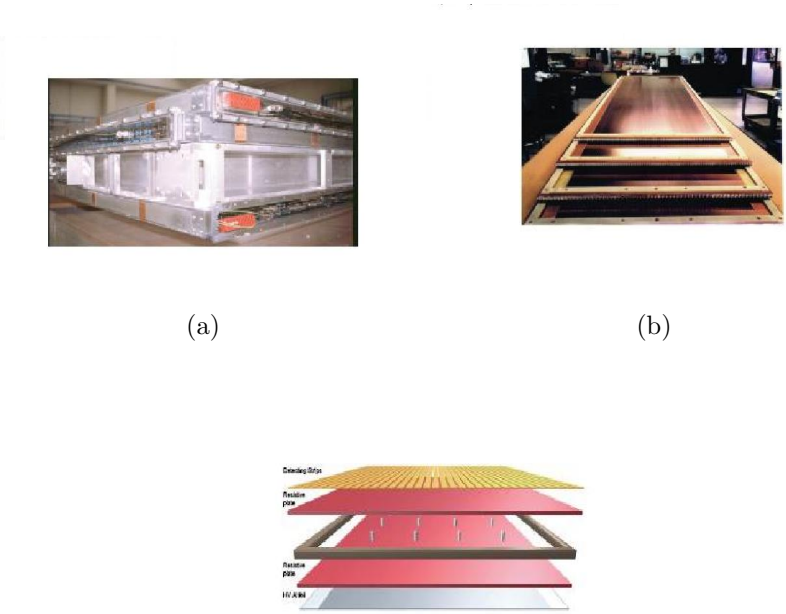
Figure 1.14: The superconducting solenoid (left) is held by a yoke ring (right). (Source: `http://cmsinfo.cern.ch/outreach`)

Figure 1.15: One of the five yoke rings. (Source: `http://cmsinfo.cern.ch/outreach`)

1.2.4 The Muon Chambers

Muons are the only detectable particles that are able to traverse both calorimeters. They are heavier than electrons and therefore the probability for an interaction via emission of bremsstrahlung is much lower, but it increases with the thickness of the traversed material. Therefore, the muon chambers are the outermost and hence the largest part of the CMS detector. They are embedded in the rings of the magnet yoke.

Three different types of detectors are used in the experiment: Drift tubes (DT) (Fig. 1.16 left) and Resistive Parallel Plate Chambers (RPC) (Fig. 1.16 right) are used in the barrel region, and Cathode Strip Chambers (CSC) (Fig. 1.16 middle) as well as RPCs are used in the endcaps.



(a) (b)

(c)

Figure 1.16: The drift tubes (a), the cathode strip chambers (b) and the resistive parallel plate chambers (c). (Source: `http://cmsinfo.cern.ch/outreach`)

Chapter 2

Data Analysis

2.1 Collisions - What now?

The creation of the accelerated particles and the detection of the particles produced in events was discussed in the previous chapter. This chapter will give an overview how events interesting for physics (e.g. creation of particles or jets with high transverse energies) are selected and how they are reconstructed and analyzed to gain the desired physics information. In CMS, the selection of an interesting event is done in real time via a trigger system consisting of two different steps, using algorithms that need to fulfill different requirements than those used in offline reconstruction. CMS software frameworks are presented that provide not only the algorithms for online and offline reconstruction, but also the tools for full as well as fast event simulation. The final section of this chapter will briefly present the different simulation possibilities of the CMS software and will show the importance of simulation in high energy physics.

2.2 Online Analysis

2.2.1 Event Selection - The CMS Trigger

Due to the intrinsic structure of protons not all collisions are going to be hard (head-on) ones. Since only those are interesting to physics analysis, a filter is needed that discards the unwanted and keeps the interesting events. This has to be done in real time at a rate of $\approx 40\,\mathrm{MHz}$ and therefore fast algorithms are needed that provide a minimum dead-time as well as maximum flexibility for new and hence unforeseen physics events. The filter used in CMS is the so-called trigger (Fig. 2.1). It is organized into two steps:

- a hardware implemented **Level 1 trigger**, using low level analysis on custom processors, and
- a **High Level Trigger**, which is a software filter executed in a multi-processor farm.

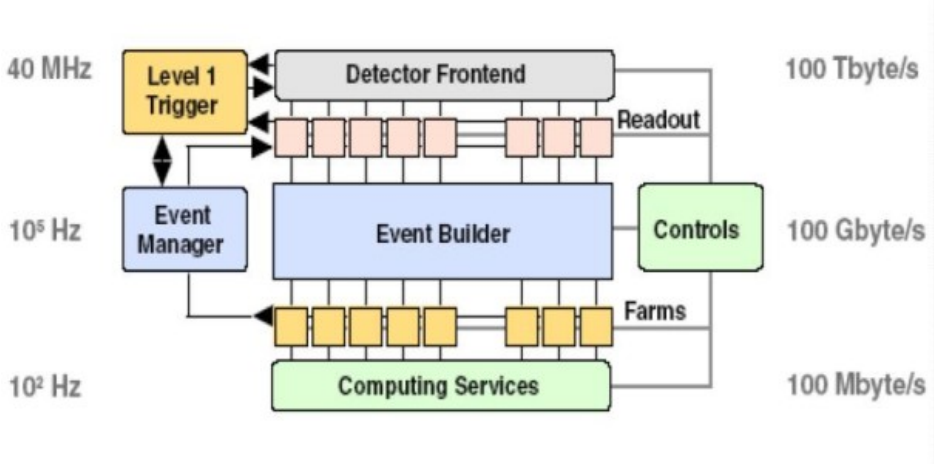


Figure 2.1: Structure of the CMS trigger and DAQ system (Source: [6])

Both trigger systems do not use the whole detector information of the event to reach their decision whether to keep an event for further analysis or not. Instead they use selective information coming from all subsystems of the CMS detector. The trigger then relates this information to physics objects. There are four basic objects that are already defined at Level 1 which the trigger uses to make its decision:

a) electrons/photons

b) muons

c) τ - jets

d) jets combined

For a detailed discussion and performance studies the reader is referred to [2, 14].The two steps of event selection are now briefly discussed in the following section. For further details see [13].

Level 1 Trigger

The Level 1 (LV1) Trigger (Fig. 2.2) reduces the data by a factor of $\mathcal{O}$ (1000) from 40 MHz to 100 kHz, needing 3 μs of latency for each decision. It consists of three major subsystems

1. LV1 muon trigger
2. LV1 calorimeter trigger
3. LV1 global trigger

The LV1 muon trigger itself consists of four further subsystems, three representing each one of the three different muon detectors used in CMS (see previous chapter), and one global muon trigger. This global muon trigger combines all the information from the three subsystems into consistent data and sends this information to the LV1 global trigger.

The LV1 calorimeter trigger consists of a local, a regional and a global trigger. The local calorimeter trigger computes the energy deposit in the different calorimeter sections. The regional calorimeter trigger (RCT) uses the output of the local trigger and finds electron and positron candidates as well as candidates for jets and isolated hadrons. It also calculates the transverse energy sums. The RCT transmits the sums and the candidates to the global calorimeter trigger (GCT). It computes the total transverse energy, the total missing energy and jet multiplicities for different thresholds. The information obtained by the GCT is then passed on to the global LV1 trigger.

The LV1 global trigger decides if an event is accepted or discarded and generates the corresponding L1 Accept signal (L1A) in the first case. It reaches its decision by synchronizing the data coming from the subsystems at different times and runs up to 128 algorithms in parallel. The final acceptance decision is a logical combination of the data sent by the global muon and calorimeter triggers. The trigger decision is sent to all detector subsystems and readout systems.

All the data used by the Level 1 trigger, the complete event data as well as all trigger objects, even those not responsible for the L1A are transmitted to the Data Acquisition system (DAQ) where they are stored temporarily for further processing.

High Level Trigger

The High Level Trigger (HLT) is a software filter running on a farm consisting of a cluster of about 7200 PCs, each using dual quadcore processors. It reduces the event rate once again by a factor of $\mathcal{O}$ (1000), leaving an output

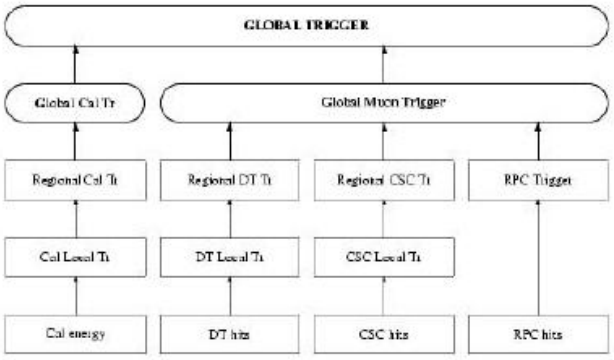


Figure 2.2: Schematic drawing of the LV1 trigger (Source: [6])

of 10^2 Hz that has to be stored for further physics analysis. The CPU time available to reach a HLT decision is about 1 s.

Since nearly all of the events discarded by the HLT are lost forever to analysis (few discarded events are kept to monitor the performance of the HLT), the requirements the HLT has to fulfill are high efficiency and high acceptance for possible unexpected situations. The algorithms used must therefore be able to reconstruct the events with nearly equal or equal quality as the offline software, which consumes a considerable amount of processing time. In order to save CPU time, the algorithms used reconstruct the event regionally and conditionally. This means that the events are reconstructed only in regions already considered interesting by the LV1 trigger, and that the reconstruction is stopped as soon as enough information has been gained to discard the event. This way it is possible to reject an event as fast as possible without sacrificing the quality of the reconstruction.

Because the HLT is not really a multi-level filter executed in several levels but is executed on a single processor farm there is no sharp distinction between a Level 2 (LV2) and a Level 3 (LV3) trigger. Nevertheless it is common to speak of a LV2 or LV3 at the HLT.

By convention Level 2 refers to algorithms based on the faster subdetectors, e.g. the muon chambers and the calorimeters, while Level 3 refers to

the full reconstruction of tracks in the tracker. Since CPU time is a key issue here and reconstruction of the total tracker information consumes a lot of time due to the high number of channels and the resulting complex pattern recognition, it is reasonable to use only partial information of the tracker. Algorithms that do so are referred to as Level 2.5.

Once an event is accepted by the HLT, it is stored for offline physics analysis. All stored events are tagged to indicate the reason of its selection in order to aid the offline reconstruction algorithms.

2.3 Offline Analysis

The process of offline analysis reconstructs all the events selected by the trigger delivering as output the desired physics results. All detector information is used to build the total event, and since the data analyzed is stored permanently the execution time of the algorithms used is not as important as in the online reconstruction. The results obtained by the offline reconstruction are interpreted and taken into account to verify or falsify theories and therefore an excellent performance and reconstruction quality is crucial.

In the subsequent section the software frameworks used in CMS offline event reconstruction are presented and discussed.

2.3.1 CMS Software

Since the development of the CMS software framework started about a decade ago, the algorithms provided by this framework were first based on FORTRAN. Today all of the framework is implemented in the object orientated language C++, although the FORTRAN based simulation tool CMSIM is still used by the CMS community [36], since its physics output can still be called more reliable than its object oriented successor. For further details on CMS software see [7, 10] or [11].

COBRA

The Basis of the software framework is formed by COBRA (Coherent Object orientated Base for Reconstruction, Analysis and simulation). It provides a few helper classes as well as various interfaces e.g. for magnetic field, data acquisition and event generators.

CARF

The heart of COBRA is the application framework CARF (CMS Analysis and Reconstruction Framework). It offers not only tools for the offline reconstruction but also for the HLT. To make it suitable for this task CARF operates on the "action on demand principle", so nothing is done unless it is needed.

ORCA

ORCA (Object orientated Reconstruction for CMS Analysis) is a software framework which can be used for the complete reconstruction chain. Functionally, it is based on CARF. Alltough a successor, CMSSW, exists, ORCA is mentioned in this thesis because parts of it (mainly the vertex subsystem) are reused in CMSSW and other frameworks such as RAVE (see below) that are important for this thesis.

Figure 2.3: From ORCA to CMSSW (Source: `http://wwwhephy.oeaw.ac.at/p3w/ilc/talks/Projektberichte/WW_060608.pdf`)

CMSSW

CMSSW [12] is a collection of software implementing a software bus model. Instead of several executables, as it was the case in the former frameworks, it has got only one executable, cmsRun and many plug-in modules. Modules simply are a piece of CMSSW code providing a specific functionality. For example, there is an output module that creates a ROOT file to analyse and store the data. The executable cmsRun is the same for detector data as well as simulated data.

CmsRun is configured by the user via a configuration file in run time. Through this file the user can specify the source (an event generator, a file etc.), the output (a ROOT file), the modules and services that shall be loaded while execution as well as the order of execution. It can also be specified if other parameters than the default values shall be used.

IGUANA

IGUANA (Interactive Graphics for User ANAlysis) [1, 21] is a generic toolkit implemented in C++. It is mainly used for developing high performance two and three dimensional graphics, as they are needed in interactive detector and event visualisation. Due to its architecture a wide variety of external tools can be integrated smoothly. IGUANA provides e.g. visualisation programs to display simulated or real events with the full or with parts of the detector. An example of an event visualized with IGUANA is given in Fig. 2.4.

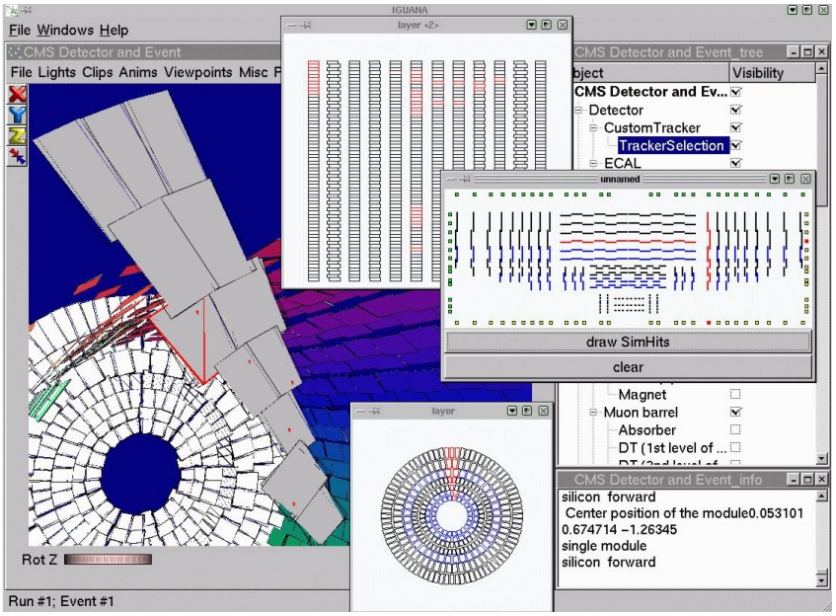


Figure 2.4: Snapshot of an event visualized with IGUANA (Source: `http://iguana.web.cern.ch/iguana/gallery.html`)

ROOT

ROOT is an analysis framework that provides a lot of tools especially for high energy physics. Like IGUANA, ROOT is implemented in the object oriented programming language C++. It was developed by high energy physicists for high energy physics, and therefore offers useful tools for the data analysts, such as histogramming and fitting, creating 2 and 3 dimensional graphics or even writing a Graphical User Interface.

The data analyst can either use ROOT from command line or write C/C++ macros that can be executed in ROOT.

2.4 Simulation

This section is dedicated to one of the most important aspects of collider experiments — simulation. An explanation why and how it is used in online and offline reconstruction is given and CMS simulation tools are presented. The reader is referred to [36] for a more detailed overview.

2.4.1 Why Simulation?

The simulation of realistic data is crucial not only for testing online and offline algorithms, but also for physics analysis. In the following a brief explanation as to why this is the case is given.

Online analysis

In online analysis separate simulations for each of the four trigger objects (see 2.2.1) are created in order to test the performance of the HLT reconstruction algorithms. Special care has to be taken here when simulating the expected background, since e. g. many processes produce a muon that can be mismeasured as a high p_T muon due to multiple scattering in the return yoke. For further detail see [14].

Offline analysis

Simulated data is created using different defined physics models, and it can be decided which model is the most accurate or probable by comparing the real data to the simulated.

For developing and testing reconstruction algorithms physics realism is unimportant. It is necessary to have data without or with low, controlled background where the user has complete control over the simulated data. CMS software offers the VertexSimPackage, where the user can define the location of the decay vertices, the multiplicity of these vertices (number of tracks originating in the vertex), the track momenta, and the reconstruction method of the tracks.

In this package reconstruction is done by smearing the simulated tracks according to a statistics model, giving them Gaussian or non-Gaussian errors. The non-Gaussian errors are obtained by combining two or more Gaussian distributed errors.

2.4.2 CMS Software

PYTHIA

The standard event generator in CMS is PYTHIA. It creates data according to various standard model processes as well as many physics scenarios beyond the standard model such as e.g. supersymmetry or string theory. The simulation applications described in the following use the output of an event generator as input.

CMSIM

CMSIM is a non object-orientated application for detector simulation written in FORTRAN and based on GEANT3. CMSIM simulates events using the description of the detector geometry, the materials used in the detector as well as information of the magnetic field.

Although an object orientated successor (OSCAR – see below) already exists, it has been used until very recently, and nearly all papers referenced in this thesis use data produced via CMSIM.

OSCAR

OSCAR (Object oriented Simulation for CMS Analysis and Reconstruction) [4] is the object orientated successor of CMSIM and is written in C++. OSCAR traces particles through the detector using the GEANT4 package enabling it e.g. to track a charged particle through a magnetic field.

FAMOS

It is not always necessary to simulate an event in depth bearing in mind that full simulation is a very CPU intensive task. Since simulation is not one large application running from beginning to end but is performed in distinct steps, shortcuts in the simulation chain do exist. These are implemented in FAMOS (Fast Monte Carlo Simulation).

2.4.3 Frameworks and data used in this Thesis

Although CMS software provides the data analyst with a huge amount of tools other applications that are independent of the underlying detector hardware do exist. For this thesis, two such applications were used and are presented in this section. The interested reader is referred to [30, 31, 37] or [37] for further information.

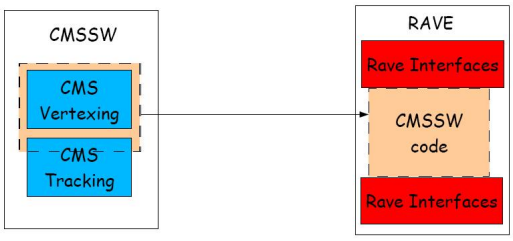


Figure 2.5: From CMSSW to RAVE (Source: `http://wwwhephy.oeaw.ac.at/p3w/ilc/talks/Projektberichte/WW_060608.pdf`)

RAVE

RAVE (Reconstruction in Abstract Vertexing Environment) is an upgradeable toolkit for Vertex Reconstruction that is independent of an underlying detector setup. It includes vertex finding as well as vertex fitting algorithms (see), using a given set of reconstructed tracks as input.

Since RAVE inherited its algorithms from the CMSSW vertex package (Fig. 2.5), it is fully compatible with with the CMS vertexing routines, but it is also easily embeddable in a variety of other software environments.

VERTIGO

VERTIGO (Vertex reconstruction and Interfaces to Generic Objects) is a stand-alone framework offering the user quick implementation, debugging and analysis of RAVE algorithms (Fig. 2.6). In VERTIGO the user has complete control over the reconstructed tracks as well as the track multiplicity. Different sources can be chosen, e.g. different vertex guns that are provided by VERTIGO itself or files containing simulated or real data.

These vertex guns can create different environements using Monte Carlo (MC) simulation, ranging from very simple events with just one vertex and a fixed number of tracks (track multiplicity) to a more "dirty" environment with two or more vertices, a variable track multiplicity including outlayer tracks (tracks that do not belong to any vertex). When using these guns the "true" values of the vertex positions and track parameters are known, making VERTIGO a convenient testing and debugging tool for reconstruction algorithms.

The user can now apply constraints as for example the beam spot constraint (primary vertex position and covariance ellipsoid), or set the number of events to be generated by the guns. One can also choose between different types of output. The reconstructed parameters can be displayed on

the screen, visualized in a graphical interface or simply stored in a ROOT file. Since all these features are accessible via commandline, VERTIGO is a comfortable and highly useful tool for the data analyst.

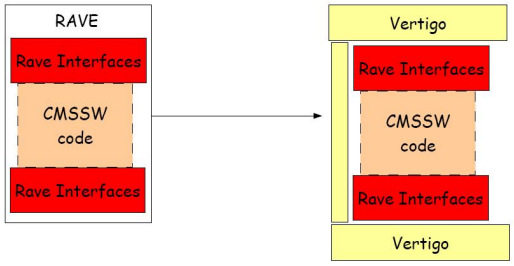


Figure 2.6: From RAVE to VERTIGO (Source: `http://wwwhephy.oeaw.ac.at/p3w/ilc/talks/Projektberichte/WW_060608.pdf`)

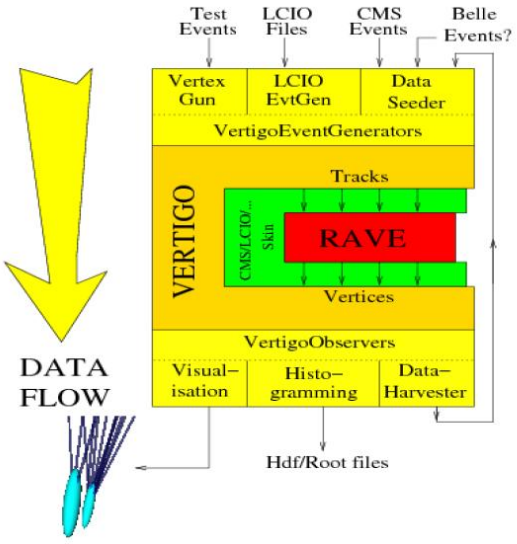


Figure 2.7: The data flow in VERTIGO and RAVE. Since both frameworks are independent of the underlying detector, the input data may originate from different sources (Source: `http://wwwhephy.oeaw.ac.at/p3w/ilc/talks/Projektberichte/WW_060608.pdf`)

For this thesis all data used for tests and performance studies on the MultiVertexFitter have been created using VERTIGO standard vertex guns as well as vertex guns implemented in VERTIGO by the author.

Chapter 3

Vertex Fitting

3.1 A Little Statistics

In this section a few statistical terms and estimation methods are explained, since they are needed in order to understand the working principle of a vertex finding respectively fitting algorithm. The interested reader may find the basic definitions in any book on statistic, e.g. [3] or books on measurement techniques in high energy physics such as [18] or [28].

3.1.1 Definitions

Random Process

A random process is a process whose outcome can not be predicted with certainty. Each possible result of a random process is represented by a random variable commonly denoted as x. Random variables can either be continuous or discrete. An example for such a process is rolling a (perfect) die since the outcome of each cast can not be predicted. In this case x is discrete and ranges from 1 to 6.

Probability Density Function

A random process is described by a probability density function (pdf) $w(x)$. The pdf gives the frequency with which an outcome of a random process is expected to occur. Considering rolling a die again $w(x)$ would be $\frac{1}{6}$ for each x (Fig. 3.1).

Depending on the random variable the pdf can be discrete or continuous (Fig. 3.2 left and right).

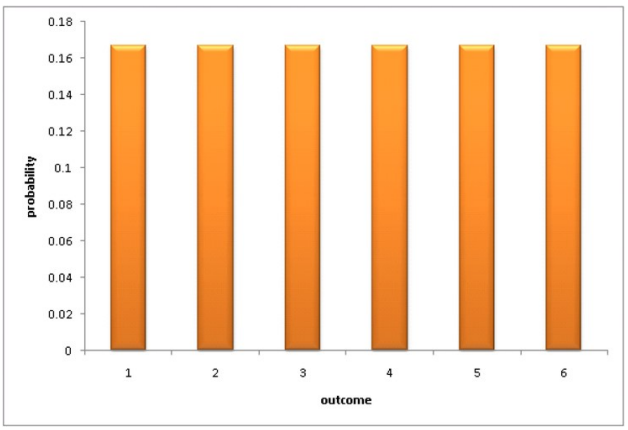


Figure 3.1: Probability distribution for rolling a perfect die.

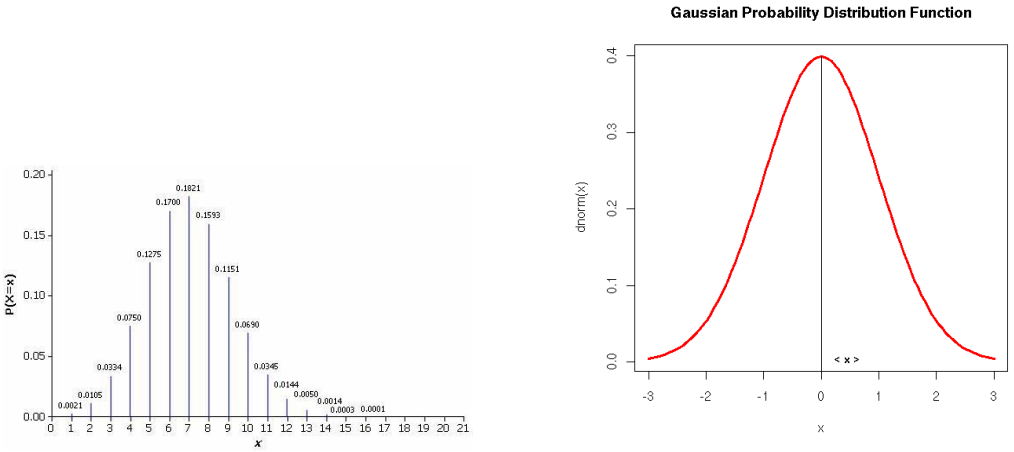


Figure 3.2: Examples of a discrete in this case a binomial (left) and a continuous (gaussian) probability density function (right). (Sources: `http://www.rossmanchance.com/iscam/exampleCh3.html` and `http://zoonek2.free.fr/UNIX/48_R/07.html`)

If it is discrete then $w(x_i)$ gives the probability of the value x_i. For a continuous distribution it does not make sense to ask for the frequency of a single value. Instead the probability of finding x in a finite interval e.g. $(x, x + \Delta x)$ is of interest. It equals to:

$$W(x \leq x' \leq x + \Delta x) = \int_{x}^{x+\Delta x} w(x')\,\mathrm{d}x'. \tag{3.1}$$

The pdf is normalized by convention:

$$\int w(x)\,\mathrm{d}x = 1, \tag{3.2}$$

or in the case of a discrete distribution

$$\sum_i w(x_i) = 1. \tag{3.3}$$

Furthermore a pdf is characterized by its moments. The n-th moment about a point x_0 of a pdf is defined as

$$\langle (x - x_0)^n \rangle = \int (x - x_0)^n w(x)\,\mathrm{d}x. \tag{3.4}$$

The most important moments are the first and the second moments of the distribution.

Since continuously distributed random variables are more important to this thesis, all further definitions will assume a continuous underlying distribution. The definitions can be easily rewritten for the discrete case by replacing the occurring integrals by sums.

Mean

The mean or expectation value of a random variable x is the first moment about zero of its pdf. It is defined as

$$\langle x \rangle = E[x] = \int x w(x)\,\mathrm{d}x. \tag{3.5}$$

This definition can also be applied to obtain the expectation of a function $f(x)$:

$$E[f(x)] = \int f(x) w(x)\,\mathrm{d}x \tag{3.6}$$

If the underlying distribution is symmetric, the mean value is the most probable value (Fig. 3.3).

Variance

The second moment about the mean is the variance

$$\sigma^2 = E[(x - \langle x \rangle)^2] = \int (x - \langle x \rangle)^2 w(x)\, \mathrm{d}x. \tag{3.7}$$

The variance is the expected squared deviation of x to the mean value and gives the width of a distribution (Fig. 3.3).

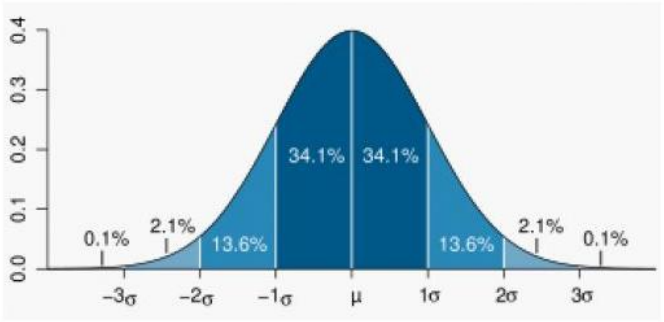


Figure 3.3: Gaussian Distribution. The mean and the standard deviation are shown. (Source: `http://hubpages.com/hub/Probability_Glossary`)

Higher Moments

The third moment about the mean of a pdf is a measure of the symmetry of a distribution. It is referred to as skewness, but since this moment or the higher ones have not much importance in practice they are not further mentioned here.

Median

The median is the middle value of an ordered list of data or of a pdf that splits the list or the pdf in an upper and an lower half of same size. To illustrate this, it is useful to consider a set of discrete data for example $M = \{2, 6, 1, 3, 5, 4, 7\}$. The ordered set would be $M = \{1, 2, 3, 4, 5, 6, 7\}$ and the median of the set is 4.

The median does not need to be a true element of a data set. Consider a set consisting of an even number of data the median is defined as the arithmetic mean of the two central values: e.g. if $M = \{1, 2, 3, 4, 5, 6\}$ the median is $\frac{3+4}{2} = 3.5$

In the case of continuous data, the median is defined as $F_r(x_\alpha) = 0.5$ where F is the integrated probability density function, the so called cumulative distribution function (cdf)

$$F(x) = \int_{\infty}^{x} w(x')\,\mathrm{d}x'. \tag{3.8}$$

In this case the median is always unique.

Mode

The mode of a pdf is the global maximum of the pdf and hence the most probable value. The task of finding the mode of a set of data being either discrete or continuous is not an easy one since the mode is not necessarily an unique value and the underlying pdf is not always known. As mentioned above the mode differs from the expectation value if the underlying pdf is asymmetric. A good example is the Landau distribution where the mean value is shifted off the maximum because of the pdf's asymmetric 'tail' (Fig. 3.4).

Covariance

So far only distributions of a single random variable have been discussed. But since a random process might depend on several variables $x_1, x_2, ..., x_n$ one has to consider multivariate distributions $w(x_1, x_2, ..., x_n)$. In this case besides the mean and the variance of the distribution (calculated as before just the integration has to be done over each variable now) a third important quantity can be defined, the covariance:

$$cov(x_i, x_j) = E[(x_i - \langle x_i \rangle)(x_j - \langle x_j \rangle)] \tag{3.9}$$

This quantity is a measure for the linear correlation of two variables. If the covariance is divided by the standard deviations (the square root of the variance) of the two values one gets the correlation coefficient ρ. If the variables are not correlated, the covariance and the correlation coefficient are zero. In case of a perfect linear correlation $\mid \rho \mid$ equals 1. All the possible covariances can be written as an $n \times n$ dimensional matrix — the so called covariance matrix.

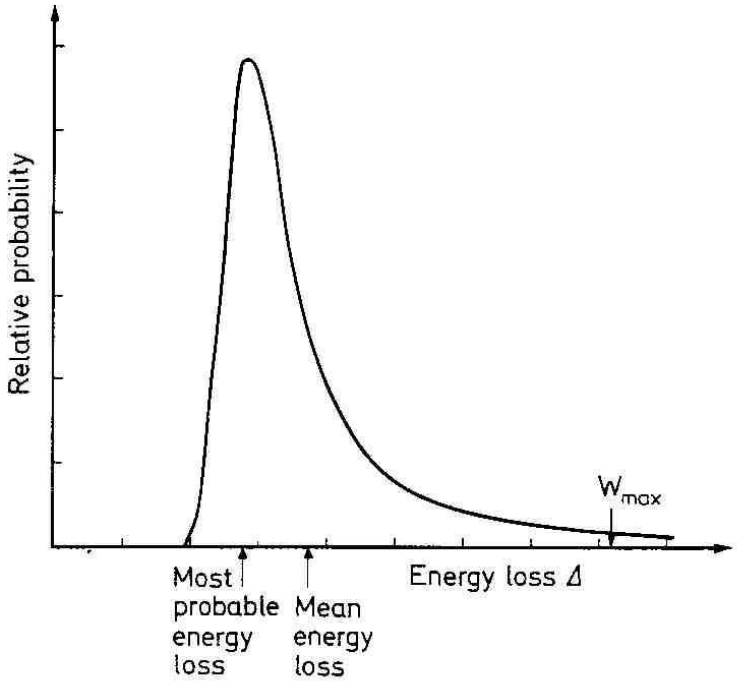


Figure 3.4: Landau distribution. (Source: [28])

3.1.2 Parameter Estimation

The task of fitting an unknown parameter (or unknown parameters) to measurement data is widely used in experimental physics in general and especially in experimental particle physics. For example in collider experiments, the track parameters of the created particles have to be estimated from the detector hits, and the coordinates of a vertex have to be determined from the reconstructed tracks.

The goal of such an estimation is to find the best value of the parameter, this being the value minimizing the variance of the estimate with respect to the true value. The determination of this value consists of two parts:

1. finding the best estimate and

2. determinating its error.

Various principles exist to accomplish this task. In the following the two most important ones for this thesis are presented.

The Maximum Likelihood Method

The Maximum Likelihood Method (MLM) applies if the distribution underlying the data is known. Consider a sample of n independent measurements $(x_1, x_2, ..., x_n)$ from which it is known (or assumed) that the underlying distribution is $w(x \mid \vartheta)$ and ϑ is the parameter that is estimated. Then the likelihood function equals

$$L(\vartheta \mid x) = \prod_{i=1}^{n} w(x_i \mid \vartheta). \tag{3.10}$$

This function can be interpreted as the probability to observe the sequence $x_1, x_2, ..., x_n$, given ϑ. To illustrate this, consider five measurements with an underlying Gaussian distribution where the mean value of this distribution should be estimated. The likelihood function in this case would be

$$L(\vartheta \mid x) = \prod_{i=1}^{5} \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x_i - \langle\vartheta\rangle)^2}{2\sigma^2}) \tag{3.11}$$

This probability should now be a maximum, given the observed values, and therefore the parameter has to satisfy

$$\frac{\mathrm{d}L}{\mathrm{d}\vartheta} = 0 \tag{3.12}$$

or

$$\frac{\mathrm{d}\ln L}{\mathrm{d}\vartheta} = 0, \tag{3.13}$$

since $\ln L$ and L have the same maximum. Solving this equation yields the estimated value $\hat{\vartheta}$. The hat indicates that this value is the estimate. If more parameters are estimated from the sample one has to take partial derivatives and solve the resulting system of equations.

Applying this principle renders the "best" value and therefore the first part of the estimation problem is solved. The simplicity of the method might strike the reader but it has to be said that solving the resulting system of equation often can't be done analytically. In the majority of cases the equations have to be solved numerically.

If we consider now another set of measurements with the same underlying distribution and apply again the MLM principle, another value will be obtained for $\hat{\vartheta}$. Thus $\hat{\vartheta}$ itself follows a distribution (which is of course immediately evident since $\hat{\vartheta}$ depends on the x_i). The error of the estimate is then the square root of the variance of the distribution of $\hat{\vartheta}$:

$$\sigma^2(\hat{\vartheta}) = \int (\hat{\vartheta} - \vartheta)^2 L(\vartheta \mid x)\, \mathrm{d}x_1\, \mathrm{d}x_2 \cdots \mathrm{d}x_n \tag{3.14}$$

So the second part of our estimation problem is solved.

Again the solution sounds quite easy and straightforward, but again σ^2 is analytically obtainable only in very few rather special cases. To make things easier and a lot less CPU intensive an approximation of σ^2 can be made that works very well in the limit of large n:

$$\sigma^2 \cong -\left[\frac{\mathrm{d}^2 \ln L}{\mathrm{d}\vartheta^2}\right]^{-1} \tag{3.15}$$

This formula is called the Rao–Cramer theorem. For deduction and further information on this theorem, the interested reader is referred to any handbook on statistics, for example [3]. If more than one parameter is fitted, then the diagonal of the inverse matrix of the derivatives yields the error for each parameter:

$$U_{ij} = -\frac{\partial^2 \ln L}{\partial\vartheta_i \partial\vartheta_j} \tag{3.16}$$

$$\sigma^2(\hat{\vartheta}_i) \cong (U^{-1})_{ii} \tag{3.17}$$

If the expectation value of $\hat{\vartheta}$ is the true value, the estimation is said to be unbiased. In case of the MLM this is in general the case only for infinite size of the sample, hence for $n \to \infty$.

A useful property of a MLM estimator is its invariance under transformation. Consider for example a differentiable parameter transformation $\vartheta = \varphi(\vartheta')$ for which $\frac{d\varphi}{d\vartheta'} \neq 0$ holds then

$$L(\vartheta) = L(\varphi(\vartheta')) = L'(\vartheta') \tag{3.18}$$

Proof:

Since

$$\frac{\mathrm{d}L'}{\mathrm{d}\vartheta'} = \frac{\mathrm{d}L(\varphi(\vartheta'))}{\mathrm{d}\vartheta'} = \frac{\mathrm{d}L}{\mathrm{d}\varphi}\frac{\mathrm{d}\varphi}{\mathrm{d}\vartheta'} \tag{3.19}$$

it follows that

$$\frac{\mathrm{d}L}{\mathrm{d}\varphi} = 0 \Leftrightarrow \frac{\mathrm{d}L'}{\mathrm{d}\vartheta'} = 0 \tag{3.20}$$

leading to

$$\hat{\vartheta} = \varphi(\hat{\vartheta}') \tag{3.21}$$

The Least Squares Method

Sometimes one is not interested in the distribution of a sample but rather in a function $f(x_1, x_2, ...)$ (model) depending on the sample data. Probably the best known example for such a case is a linear fit or regression line through

a set of data points. The model is then a linear function and two parameters have to be estimated. This special case will be discussed now to illustrate the Least Square Method (LSM).

Consider measurements of variable y at n points $(x_1, x_2, ..., x_n)$. So $y_i (i = 1, ..., n)$ with the error σ_i are obtained. A function $\hat{y}_i = f(x, a_1, a_2, ..., a_m)$ is now fitted where the $a_j (j = 1, ..., m)$ are the parameters to be estimated. Of course the condition $m < n$ has to be satisfied since it is impossible to estimate more parameters than there are data points.

According to the LSM the best estimate of the parameters a_j are the ones for which the sum of the squared difference of the y_i from the estimate $\hat{y}_i$ weighted by the error of the variable y_i is minimal[1]:

$$S = \sum_{i=1}^{n} \left[\frac{y_i - f(x_i, a_j)}{\sigma_i} \right]^2 \rightarrow \min \tag{3.22}$$

The function S is the so called objective function. To determine the parameters a_j one has to solve the following system of equations:

$$\frac{\partial S}{\partial a_j} = 0, j = 1, ..., m. \tag{3.23}$$

As in case of the MLM those equations usually have to be solved numerically, since only for a few functions $f(x)$ an analytic solution exists.

To illustrate this estimation problem, consider a set of n coordinates x_i that will be fitted with a linear function $f(x, a_1, a_2) = a_1 x + a_2$.It is obvious that we need at least two points in order to determine the two parameters. Applying the LSM yields

$$S = \sum_{i} \left[\frac{y_i - (a_1 x + a_2)}{\sigma_i} \right]^2 \tag{3.24}$$

and therefore the system of equations

$$\frac{\partial S}{\partial a_1} = -2 \sum_{i} \frac{(y_i - a_1 x_i - a_2) x_i}{\sigma_i^2} = 0 \tag{3.25}$$

$$\frac{\partial S}{\partial a_2} = -2 \sum_{i} \frac{(y_i - a_1 x_i - a_2)}{\sigma_i^2} = 0 \tag{3.26}$$

[1]The squared differences are also known as residuals while the weighted differences are called the standardized residuals. So the best estimate is the minimum of the sum of the standardized residuals.

Solving this equations one finds:

$$a_1 = \frac{\sum \frac{x_i y_i}{\sigma_i^2} \sum \frac{1}{\sigma_i^2} - \sum \frac{y_i}{\sigma_i^2} \sum \frac{x_i}{\sigma_i^2}}{\sum \frac{x_i^2}{\sigma_i^2} \sum \frac{1}{\sigma_i^2} - (\sum \frac{x_i}{\sigma_i^2})^2} \tag{3.27}$$

$$a_2 = \frac{\sum \frac{x_i^2}{\sigma_i^2} \sum \frac{y_i}{\sigma_i^2} - \sum \frac{x_i y_i}{\sigma_i^2} \sum \frac{x_i}{\sigma_i^2}}{\sum \frac{x_i^2}{\sigma_i^2} \sum \frac{1}{\sigma_i^2} - (\sum \frac{x_i}{\sigma_i^2})^2} \tag{3.28}$$

The first part of the estimation problem, finding the best values, is solved now.

The errors on the parameters a_j form the covariance matrix:

$$(V^{-1})_{ij} = \frac{1}{2} \frac{\partial^2 S}{\partial a_i \partial a_j} = \begin{pmatrix} \sum \frac{x_i^2}{\sigma_i^2} & \sum \frac{x_i}{\sigma_i^2} \\ \sum \frac{x_i}{\sigma_i^2} & \sum \frac{x_i^2}{\sigma_i^2} \end{pmatrix} \tag{3.29}$$

The least squares method is optimal for linear functions which means that it is the linear unbiased estimator with the least variance if the covariance matrix of the errors is non-singular. (Gauss-Markov theorem). Unfortunately it is not robust to outliers or noise (Fig. 3.5).

Robustifications of the LSM exist and a few algorithmic realizations to solve the problem of mode finding are discussed in the following.

Test for Goodness of a Fit — the Chi-Square Distribution

The chi-square of n random variables x_i following Gaussian distributions with mean value μ_i and standard deviation σ_i is defined as

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{x_i - \mu_i}{\sigma_i} \right)^2 . \tag{3.30}$$

The distribution of the χ^2 is

$$P(\chi^2)\,\mathrm{d}\chi^2 = \frac{(\frac{\chi^2}{2})^{\frac{\nu}{2}-1} e^{-\frac{\chi^2}{2}}}{2\Gamma(\frac{\nu}{2})}\,\mathrm{d}\chi^2 \tag{3.31}$$

Here the only parameter of the distribution, ν, is the number of degrees of freedom of the distribution, and $\Gamma(\frac{\nu}{2})$ is the gamma function. The degrees of freedom are simply the number of independent random variables in the χ^2 sum.

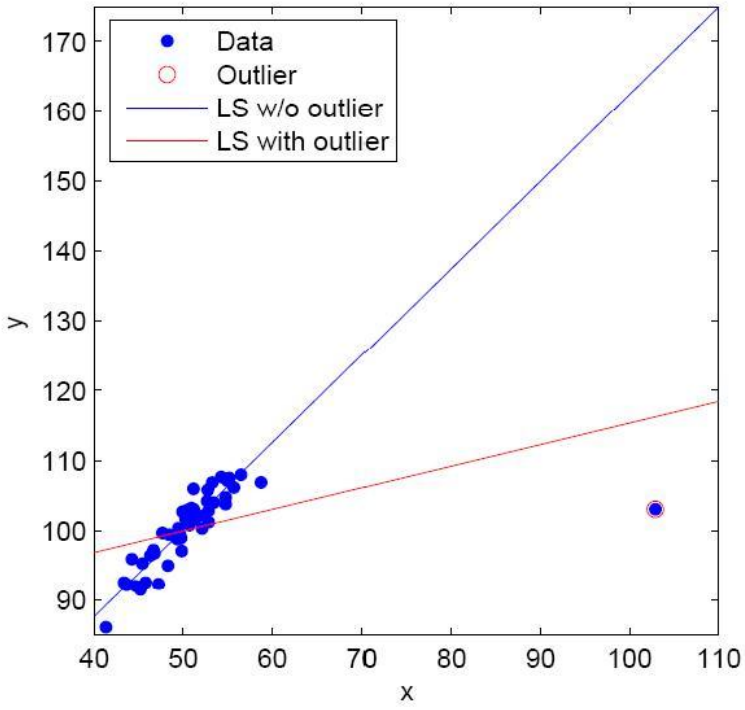


Figure 3.5: Linear regression through a set of data points. It can be seen how an outlayer distorts the estimate. (Source: [16])

Since the χ^2 is the sum of standardized residuals, the χ^2 distribution provides a measurement of the fluctuations of the data x_i. When the standardized residuals of the x_i and a theoretical mean value and standard deviation are calculated, the chi square provides a measure of the goodness of a fit.

In case of the least squares method parameter estimation, the value of the function (3.22) at the minimum is the χ^2. A commonly used test is to form the reduced χ^2, which is just the χ^2 divided by its number of degrees of freedom ν. In case of n data points and m parameters estimated from these, $\nu = n - m$. For a good fit the reduced χ^2 should be close to 1. One can also calculate the probability of $P(\chi^2 \geq S)$ (S being the calculated value of the χ^2). If it is greater than 5%, the fit is acceptable. To illustrate the use of the χ^2, here an example similar to the one in [28].

Consider a set of given data points through which the best linear function should be fitted:

x	0	1	2	3	4	5
y	0.92	4.15	9.78	14.46	17.26	21.9
σ	0.5	1.0	0.75	1.25	1.0	1.5

The parameters of such a linear fit are calculated using equations 3.27 and 3.28. In case of the data above, the parameters are found to be

$$\begin{aligned} a_1 &= 4.227 \\ a_2 &= 0.878. \end{aligned}$$

So the fit function is

$$f(x) = 4.227x - 0.878.$$

The χ^2 is then calculated to be

$$\chi^2 = \sum_i \left(\frac{y_i - f(x_i)}{\sigma_i} \right)^2 = 2.078.$$

The set consists of 6 data points and for the linear fit, 2 parameters had to be estimated. So the degrees of freedom of the χ^2 distribution are 6 - 2 = 4. The reduced χ^2 equals

$$\frac{\chi^2}{\nu} = \frac{2.078}{4} \approx 0.5.$$

The probability $P(\chi^2 > 2.07)$ with $\nu = 4$ is

$$P(\chi^2 > 2.07) = \approx 0.975 = 97.5\%$$

Since in this case the reduced χ^2 is close to one and the probability P is much greater than 5 %, the fit can be considered as good.

3.2 Mode Finding

Mode finding [36] is an estimation problem, where one parameter, the maximum of a data sets unknown distribution, is to be estimated. For this thesis the problem of mode finding can be divided into two classes

1. mode finding in one dimension
2. mode finding in three dimension

For greater details on the algorithms in one as well as three dimensions presented in the following, the reader is referred to [36].

3.2.1 Algorithms for mode finding in one dimension

One approach for solving the problem of mode finding in one dimension is the **Least Median of Squares Method** (**LMS**). It minimizes the median of the residuals (see Eq. 3.22):

$$S = \text{med}\left[\frac{y_i - f(x_i, a_j)}{\sigma_i}\right]^2 \rightarrow \min \tag{3.32}$$

But, like the LSM, it is not robust to noisy data and a general and fast algorithmic solution for (3.32) is not known, except splitting the data points into subsets and trying out every possible combination. One can imagine of course, that this is a very CPU intensive task and therefore not applicable in high energy physics. Faster solutions exist for example selecting few combinations of data points at random (**Least Trimmed Squares** – **LTS**).

Another approach is the so called **Half Sample Mode** (**HSM**) which is a recursive LMS. Iteratively an LMS is performed on each points in the interval that has been found in the step before. A generalization of this algorithm is the **Fraction of Sample mode with Weights** (**FSMW**). In this algorithm weights are associated with the data points and the "weighted" interval (length of the interval divided by the sum of all weights of the contained data points) covering $k\,\%$ of the data is determined.

A very simple algorithm is the **Maximum Two Values** (**MTV**) algorithm. Here too, weights are assigned to the data points. The estimated mode is then the weighted mean of the two consecutive data points with the highest sum of weights. There are similar algorithms that use only one or three such data points — the former being called **Maximum Single Value** (**MSV**) the latter **Maximum Three Values** (**M3V**).

3.2.2 Algorithms for mode finding in three dimensions

One has to distinguish between true three–dimensional mode finders and coordinate–wise mode finders. The latter splits the three dimensional (more general an n dimensional) mode finding into three (n) one–dimensional problems.

Only one family of true three–dimensional mode finders does exist: the **Small Median of Squares** (**SMS**) algorithm. This algorithm computes the median of the distances for each data point to the other data points. Then the data points are sorted according to this median distance. The mode is then the mean of the top $k\,\%$ of the data points, k being user–defined.

A recursive SMS algorithm does exist, the **Iterated Median of Squares** (**ISMS**). It is the three–dimensional analogon to the HSM, performing iteratively the SMS algorithm using the former output as new input. Introducing weights in the latter algorithm one gets the **Iterated Small Median of Squares with Weights** (**ISMSW**) algorithm. The weight, which simply is the sum of the weights of two data points, are associated with the distance between these two points. The distances are then sorted according to the length for each data point. The weights of the distances are now summed up, starting from the top of the list until the sum is > 50 % of the total sum of all weights. This is then the average weighted distance by which the data points are finally sorted. The median is now the weighted mean of $k\,\%$ of the data points.

3.3 Vertex Finding

Vertex Finding is the task of grouping all tracks into subsets that have common points of origin. This section will give an overview of concepts and algorithms used for vertex finding. The interested reader is referred to [36] for further details.

The input for any vertex finding algorithm is a set of tracks. It has to be said that for the minimum distances between these tracks, the triangle inequality is generally violated. So since these tracks have such unpleasant metric properties different concepts do exist on how to deal with the input tracks. One such concept is the introduction of apex points and it will briefly be discussed here.

The apex points are simple points in a three–dimensional Euclidean space that represent the tracks. So the problem of vertex finding is split into two problems: Finding these representative points and then searching for a vertex in the set of them. Since the apex points exist in ordinary three dimensional

space, all algorithms presented in the last section can be applied. Since there is no track information such as track direction in the apex points, this concept simply trades information for metricity.

Of course track information can be introduced in the apex points via a covariance matrix. But the distances between two tracks weighted with these covariance matrices again do not form a metric space.

The algorithms used for vertex finding can be divided into two classes:

- hierarchic and
- non–hierarchic algorithms.

Hierarchic algorithms have a strict clustering hierarchy while non–hierarchic algorithms lack such a hierarchy. The former can be subdivided further into agglomerative and divisive clusterers. Only a brief overview over these algorithms is given here. For more information on the implementation of those clusterers in CMS software, the reader is referred to [36].

Agglomerative clusterers start by assigning each track to an own subset. These subsets are then called singleton groups. Singletons that are most compatible are then merged iteratively, until a stopping condition is reached. Of course it is assumed implicitly that at least two tracks have to be compatible enough with each other. The definition of compatibility is a direct consequence of the employed metric.

Divisive clusterers work the other way round. The start by assigning all tracks to one big cluster. This cluster is split up into smaller ones that are again split up and so on till the stopping condition is reached. The implicit assumption here is that at least two tracks must be incompatible with the primary vertex in order to find a secondary vertex. Furthermore those incompatible tracks have to be compatible enough with each other.

As mentioned above, non–hierarchic clusteres lack a strict clustering hierarchy. There will be four algorithms briefly discussed here.

The **vector quantization** algorithm [20] uses code vectors or vertex prototypes as a representation of the input data vectors. In the case of fitting, these data vectors are the apex points. The code vectors are taken from a code book and do not necessarily live in the same space as the data vectors. In the fitting process however, the prototypes live in the apex space too. They are attracted by the apex points and the final position of them are the locations of the vertex candidates. It is possible to introduce different learning procedures to improve the algorithm. For details on this, see [36].

The **KMeans** algorithm works in a similar fashion as the vector quantization. The difference is, that there is a fixed number of code vectors to which the apex points are associated. The apex points belonging most likely

to one prototype form a cluster. In each cluster the center is computed via apex fit. The apex points are then reassociated to this new centers, new cluster are formed and their centers refitted. This procedure is repeated until convergence. For more information the reader is referred to [36].

Both of the afore mentioned algorithms need initial prototypes that provide good starting positions, or in short, they need seeding.

The next two algorithms discussed here use metaphors of physical models.

The **deterministic annealing** [33, 34] exploits a metaphor to a thermodynamical system consisting of data points that cools down. Clusters of those points move and split with each phase transition.

The **gravitational clustering** [25] assumes an attractive force between the weighted data points that follows an $1/r^2$ rule, r being the distance between two vincinated data points. When two data points clash, a larger point is obtained and its weight simply is the sum of the weights of the former points.

Another type of vertex finders are the Superfinders [36]. Those algorithms combine the results of other vertex finding algorithms into a new and improved solution.

The output of the superfinder can be simply the value that minimizes a formal criterion as for example the global association criterion (for further details on this, see [36]) or a real combination of the input results and hence a new solution. The latter is done by a voting algorithm that will be presented briefly now.

The voting algorithm

The voting algorithms starts by finding the two most similar clusters in the input solutions. Similarity is defined as

$$\mathrm{sim}(\alpha, \beta) \equiv \frac{n(\alpha \cap \beta)^2}{n(\alpha) n(\beta)} \tag{3.33}$$

where n(α) is the number of tracks in the cluster α. The two clusters are merged into a new cluster containing the tracks of both with assigned weights. Then the two previous clusters are erased and the process is repeated. The weights assigned in each step are calculated as follows. When the first two clusters are merged, each track present in both clusters receives a weight equal to one, while the others receive a weight of $\frac{1}{2}$. In the n-th step, the cluster with the tracks of $n-1$ clusters is assigned a weight of $\frac{n-1}{n}$ while the other is assigned a weight of $\frac{1}{n}$. To illustrate this, I will give a short example similar to the ones in [36]:

Consider three solutions s_1, s_2 and s_3, each consisting of three or four clusters containing one to four tracks[2]:

$$\begin{aligned} s_1 &= \{AB\}\{CDEF\}\{G\}\{H\} \\ s_2 &= \{ABC\}\{DEFG\}\{H\} \\ s_3 &= \{ABE\}\{CD\}\{FGH\} \end{aligned} \tag{3.34}$$

When the solutions s_1 and s_2 are merged, one yields

$$s_1' = \frac{1}{2}s_1 \oplus \frac{1}{2}s_2 = \left\{AB\frac{C}{2}\right\}\left\{\frac{C}{2}DEF\frac{G}{s}\right\}\left\{\frac{G}{2}\right\}\{H\} \tag{3.35}$$

Then the new cluster s_1' is merged with solution s_3 giving the new and hopefully better solution s_{end}:

$$s_{end} = \frac{2}{3}s_1' \oplus \frac{1}{3}s_3 = \left\{AB\frac{C}{3}\frac{E}{3}\right\}\left\{\frac{2C}{3}D\frac{2E}{3}\frac{2F}{3}\frac{G}{3}\right\}\left\{\frac{G}{3}\right\}\left\{\frac{F}{3}\frac{G}{3}H\right\} \tag{3.36}$$

For further details as for example resolving possibly occurring ambiguities[3] see [36].

3.4 Vertex Fitting

Vertex Fitting [36] is the task to estimate a vertex position and its covariance matrix out of the already determined subsets of tracks. Basically there are two types of fitters:

- non robust[4] and
- robust fitters.

Every fitter also needs an initial guess of the vertex position, the so called linearization point or seed. The necessary quality of the seed is determined by the type of algorithm used. Non-robust algorithms do not need a good seed since the tracks are relinearized if the calculated vertex candidate is too far off the linearization point. Robust fitters however are local optimization algorithms and therefore they need an accurate initial guess.

[2]The tracks are denoted with capital letters.

[3]This is the case when no unique pair of most similar clusters exists.

[4]In this thesis exclusively least square fitters are referred to as non robust fitters.

The input of each fitter is a set of tracks. In the first step, these tracks need to be linearized. Therefore they are re-parameterized. Although the choice of the parameters is not unique, only a few parameterizations are of practical use as for example the perigee parameters[5].

Algorithms that provide an initial guess are called Linearization Point Finders. They can be divided in two groups:

- those using the concept of apex points (see 3.3.1) and
- those that are based on so called crossing points between the tracks.

The crossing points are the arithmetic mean of the points of closest approach (PtCAs) between the tracks. Those crossing points are used as an input in a three dimensional mode finder. Algorithms that find crossing points do not solely provide the position of these points but also a weight. This weight is simply dependent on the distance[6] between two PtCAs.

Two non robust least square fitting methods and three robustifications of these are briefly discussed in the following. For further information the reader is referred to [36].

Non Robust Fitting methods

The **Linear Vertex Fitter** computes the impact points of the tracks with respect to the initial seed as well as their error. Should the calculated vertex change very much during an iteration a relinearization of the tracks is done. The tracks are linearized in the track phase space.

The **Kalman Vertex Fitter** uses tracks in perigee parametrization $(\epsilon, z_p, \Theta, \Phi_p, \rho)$ [29]. Fig. 3.6 illustrates these parameters. The tracks are linearized in these five parameters and then a Kalman filter technique is applied. The working principle of a Kalman filter in the task of vertex fitting will be described here briefly. The reader is referred to [18, 32, 17] or [38] for further details.

A Kalman filter is a recursive application of least squares estimators. It minimizes the standardized residuals of all tracks to the vertex candidate. The goal of such a fit is to determine the state vector, containing the estimated vertex position x and the momenta of all tracks belonging to the vertex candidate. Therefore the tracks are treated as so called virtual measurements, that contain the total information of the tracker.

[5]The perigee parameters will be discussed in greater detail in the following.

[6]It is $\propto \frac{1}{d^2}$ where d denotes the distance between the PtCAs.

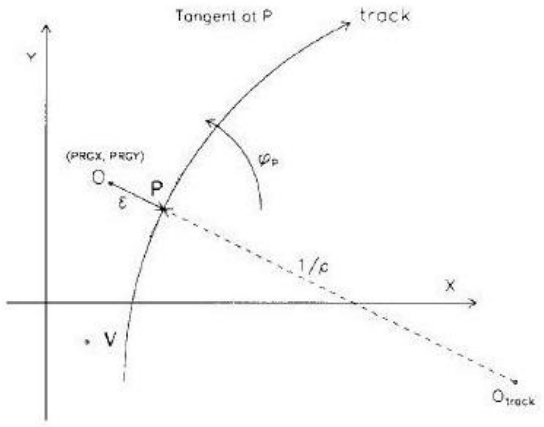


Figure 3.6: Illustration of the perigee parameters $(\epsilon, z_p, \Theta, \Phi_p, \rho)$ with respect to a reference position O (e.g. an initial vertex position) with the coordinates(PRGX / PRGY / PRGZ). The minimal distance between O and the track ($\overline{OP}$) is ϵ, z_p is the z coordinate of P, Θ is the trajectories' polar angle, Φ_p is the azimuthal angle of the trajectory at P and ρ is the inverse radius of the track curvature. (Source: [29])

At the beginning of the iteration the state vector consists only of an initial guess of the vertex position $\mathbf{x}_0$ and its covariance matrix. A first track is added, leaving the initial vertex position unaltered and adding the tracks momentum vector $\mathbf{q}_1$ to the state vector. The way the track parameters $\mathbf{p}_k$ depend on the vector is given by the equation of motion and described by a usually nonlinear measurement equation (see also [17]).

$$\mathbf{p}_k = \mathbf{h}_k\left(\mathbf{x}, \mathbf{q}_k\right) + \epsilon_k \tag{3.37}$$

with

$$cov(\epsilon_k) = \mathbf{V}_k \tag{3.38}$$

where $\mathbf{V}_k$ is the to the track model corresponding error matrix. This equation can be approximated by a linear Taylor expansion

$$\mathbf{h}_k \approx \mathbf{h}_k(\mathbf{x}_e, \mathbf{q}_{k,e}) + \mathbf{A}_k(\mathbf{x} - \mathbf{x}_e) + \mathbf{B}_k(\mathbf{q}_k - \mathbf{q}_{k,e}) = \mathbf{c}_{k,e} + \mathbf{A}_k\mathbf{x} + \mathbf{B}_k\mathbf{q}_k. \tag{3.39}$$

Where $\mathbf{A}_k$ and $\mathbf{B}_k$ are the matrices of derivatives evaluated at $(\mathbf{x}_e, \mathbf{q}_{k,e})$ and $\mathbf{c}_{k,e}$ is a constant matrix.

When a track is added, the vertex position is refitted and the momentum of the track is added to the state vector both according to the Kalman filter. For the addition of the track k to the state vector obtained by fitting with $k-1$ tracks, the Kalman filter would be

$$\widetilde{\mathbf{x}}_k = \mathbf{C}_k\left[\mathbf{C}_{k-1}^{-1}\widetilde{\mathbf{x}}_{k-1} + \mathbf{A}_k^T\mathbf{G}_k^B\left(\mathbf{p}_k - \mathbf{c}_{k,e}\right)\right] \tag{3.40}$$

$$\widetilde{\mathbf{q}}_k = \mathbf{W}_k\mathbf{B}_k^T\mathbf{G}_k\left(\mathbf{p}_k - \mathbf{c}_{k,e} - \mathbf{A}_k\widetilde{\mathbf{x}}_k\right) \tag{3.41}$$

$$\tag{3.42}$$

with

$$\begin{aligned}\mathbf{W}_k &= \left(\mathbf{B}_k^T\mathbf{G}_k\mathbf{B}_k\right)^{-1}\\ \mathbf{G}_k^B &= \mathbf{G}_k - \mathbf{G}_k\mathbf{B}_k\mathbf{W}_k\mathbf{B}_k^T\mathbf{G}_k.\end{aligned}$$

Here $\widetilde{\mathbf{x}}_k$ is the estimated vertex position fitted with k tracks, $\mathbf{C}_k$ is the covariance matrix of the parameters of track k and $\mathbf{G}_k$ is the inverse covariance matrix of the vertex position after adding track k.

This procedure is repeated until no more tracks are left.

Robustifications of Least Squares Fitting Methods

The **Trimming Vertex Fitter** uses only a certain fraction of the tracks. In order to return the global minimum of the objective functionall combinations of tracks should be tried out, but since we have an initial guess of the vertex position, one can always find a certain number of tracks most compatible with this seed. The compatibility of the tracks with the seed is determined by a χ^2 test. The vertex is then refitted with these tracks and again the most compatible tracks are found. This is repeated until convergence is reached.

The **Least Median of Squares (LMS) Fitter** does not minimize the sum of the squared residuals but their median. Neither a simple analytic solution nor a satisfactory algorithmic implementation exists to solve this problem.

The **Adaptive Vertex Fitter (AVF)** does not discard outlayers like the trimmer, but down-weights them. Weights are assigned to each track according to the weight function

$$w_i(\chi_i^2) = \frac{e^{\frac{-\chi_i^2}{2T}}}{e^{\frac{-\chi_i^2}{2T}} + e^{\frac{-\chi_{cutoff}^2}{2T}}} \tag{3.43}$$

where χ_i^2 denotes the χ^2 compatibility of track i with the vertex candidate and T is the temperature according to an annealing schedule. Since it is an iterative algorithm, the vertex is re-fitted and the tracks are re-weighted in each iteration step.

Also an annealing schedule to invoke deterministic annealing is used. So the AVF starts with the initial vertex candidate computed by a seeding algorithm. At an initial temperature a weight is assigned to each track. Then the vertex is refitted using a Kalman Vertex Fitter and the temperature is multiplied with a factor ≤ 1.[7] Then the tracks are re-weighted. This is repeated until convergence.

The Adaptive Vertex Reconstructor (AVR) uses the AVF method iteratively to fit one vertex at a time. It starts by refitting the initial vertex guess with the most compatible tracks, then reweighs those tracks with respect to the newly found vertex candidate and refits it again with its most compatible tracks. This is done until convergence is reached. Then the procedure is repeated using the tracks that have not been assigned to any vertex yet.

The AVR qualifies as a general purpose algorithm that performs well using

[7] According to the used annealing schedule.

pure data as well as noisy data. For greater details as well as performance studies of the AVR, the reader is referred to [36].

The **Multi Vertex Fitter (MVF)** can be regarded as a generalization of the adaptive vertex fitting method. The next chapter discusses this algorithm in detail as well as its performance.

Chapter 4

Optimization of the Multi Vertex Fitter

This chapter will present the Multi Vertex Fitter algorithm and its implementation details. Furthermore changes and extensions of the MVF implemented by the author are presented along with studies illustrating the results yielded by them. These results are then compared with the already well–tested Adaptive Vertex Reconstructor algorithm.

4.1 The Multi Vertex Fitter

4.1.1 The Algorithm

The Multi Vertex Fitter (MVF) is quite similar to the adaptive vertex fitter mentioned before, and in case of a single vertex to be reconstructed, the two are equivalent. The difference between the MVF and other algorithms is that it fits n vertices simultaneously. As it does not deliver only one vertex at a time it is technically not a vertex fitter at all, but its own category.

The iterative Multi Vertex Fitter does not use prototypes, instead it assigns a dynamic list to each track containing the assignment probabilities for each track to each vertex. These lists change with every iteration enabling the algorithm to keep track of the assignment probabilities. The clusters of tracks that are the input for the MVF are computed by a voting algorithm, as are the initial weights. The MVF uses a special seeding algorithms, the Multi Vertex B Seeder (MBS), that is supposed to work well in b-jet setups. During the optimization process of the MVF, the author has tested and expanded this seeder.

The Vertex Fitting Process

The input seeds are refitted with the initial clusters, with the assignment probabilities of the tracks as initial weights. The refitting is done using a set of weighted Kalman filters running in parallel. The weights are then recomputed with the new vertex positions.

So in each iteration, the Multi Vertex Fitter computes the weights, using the weight function[1]:

$$w_{ij} = \frac{e^{\frac{-\chi^2}{2T}}}{e^{\frac{-\chi^2_{cutoff}}{2T}} + \sum_{k=1}^{n} e^{\frac{-\chi^2_{ik}}{2T}}}, \tag{4.1}$$

where w_{ij} denotes the weight of the i-th track with respect to the j-th vertex, χ^2_{ij} is the χ^2 compatibility between the i-th track and the j-th vertex and T is the temperature set according to an annealing schedule. In the Multi Vertex Fitter there are two annealing schedules available, the step–by–step annealing and the geometric annealing schedule.The process of refitting and reweighing is then repeated until convergence is reached.

4.1.2 Implementation of the Multi Vertex Fitter

The Multi Vertex Fitter does not derive from any super class, but reuses classes already implemented for other fitters. As mentioned above, it reuses the annealing schedules implemented for the Adaptive Vertex Fitter and also many classes implemented for linear fitters, such as the Kalman Vertex Fitter.

The MVF is organized in methods[2] that represent the individual tasks of the algorithm. The central method is called **vertices**. This method calls **createSeed** and **fit**. In **createSeed** the seeding algorithm is called and the initial vertex positions and the initial track weights are returned. The method **fit** recalculates the vertices as well as the weights by calling **updateSeeds** and **updateWeights**. The method **fit** returns an STL vector with entries of the type CachingVertex that includes the reconstructed vertices, the tracks and their weights. Said vector is then returned as final result of the MVF.

Many other small methods exist inside the MVF code and a few important ones will be presented here. Method **minWeightFraction** fixes the minimal weight a track must have in order to belong to a certain vertex. Method **discardLightWeights** discards tracks that are beneath that threshold so that they are not used for the fit in the next iteration step. Furthermore

[1]This function can be motivated statistically, via effective energy or even by using a quantum mechanical analogy. For further details on this, see [36].

[2]The methods described here are printed boldly.

method **validWeight** checks that the weights do not exceed 1 or fall below 0.

4.2 Performance Studies

The data used to study the performance of the MVF were generated in VERTIGO, using VERTIGO's standard bjet gun as well as bjet–like guns implemented by the author. All the guns used create a primary vertex and a single bjet–like secondary vertex. The track multiplicity at the primary vertex is 9 to 11 tracks with an angular spread of 1.3 rad for all guns, and the multiplicity at the second vertex is 3 to 5 tracks with an angular spread of 262 mrad. The standard gun creates data with some contamination[3] while the guns implemented by the author create unbiased data. The distance between the two created vertices ranges from the actual flight distance of a b meson, which is about 2 mm up to an enormous distance of 1 meter. [4]

These setups, unrealistic as they may be, are merely a means to test the performance of the algorithms, starting with easily distinguishable vertices at very large distances, and systematically moving the two vertices closer to each other. The distances used in each case will be given explicitly in the appropriate sections.[5]

In this work the quality of the reconstruction will be determined by two criteria:

1. The number of reconstructed vertices that could be associated[6] with the Monte Carlo truth.

2. The standard deviation of the position of the reconstructed vertices.

The latter is obtained via a Gaussian fit through the cumulated deviations of every single event. In each presented scenario a number of 5001 events were simulated.

As mentioned before, within the VERTIGO framework, the user is free to provide the reconstruction algorithm with individually set parameters.

[3] Two outlayer tracks are created.

[4] Note that in this case the term distance refers to an average value corresponding to the mean value of the even distribution underlying the simulation algorithm.

[5] A general remark: In some of the graphs presented here the abbrevations wb and wob will occurr. In this thesis wb stands for with beamspot constraint and wob for without beamspot constraint.

[6] Association is done via the actual physical distance between the simulated and reconstructed vertex.

Among these only two parameters have an influence on the aforementioned weight function:

1. The σ_{cut} which is the square root of the χ^2_{cutoff} in Eq. (4.1), and
2. the temperature T.

The default values of these parameters are:

- $\sigma_{cut} = 9$
- $T = 8$

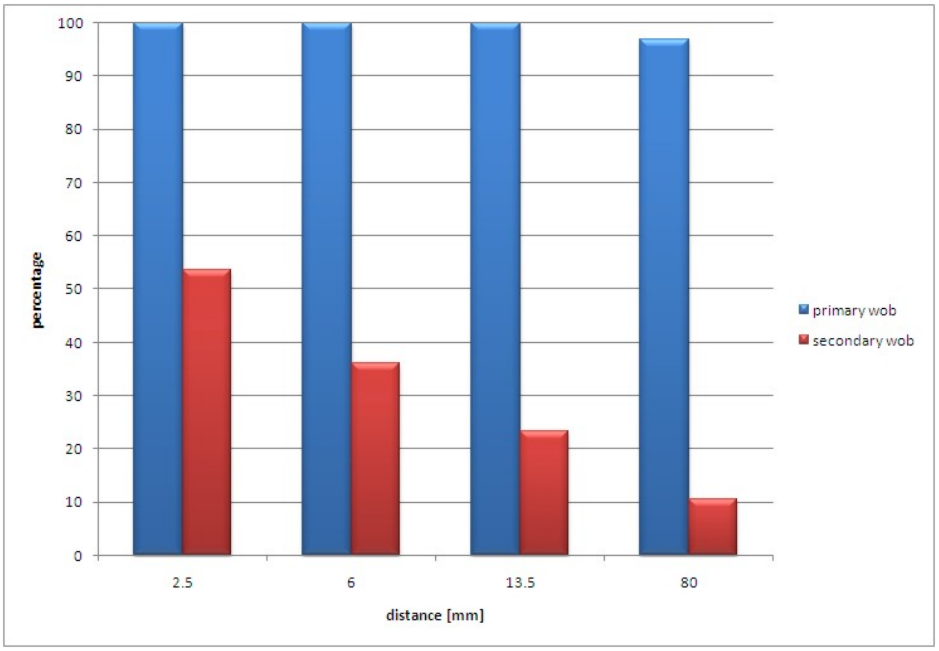
For testing the MVF algorithm the value $\sigma_{cut} = 2.5$ and the default value for T were chosen using the default values of the AVR as guideline. In section 4.7 the influence of the σ_{cut} on the reconstruction quality will be discussed in more detail.

4.3 The Original Algorithm

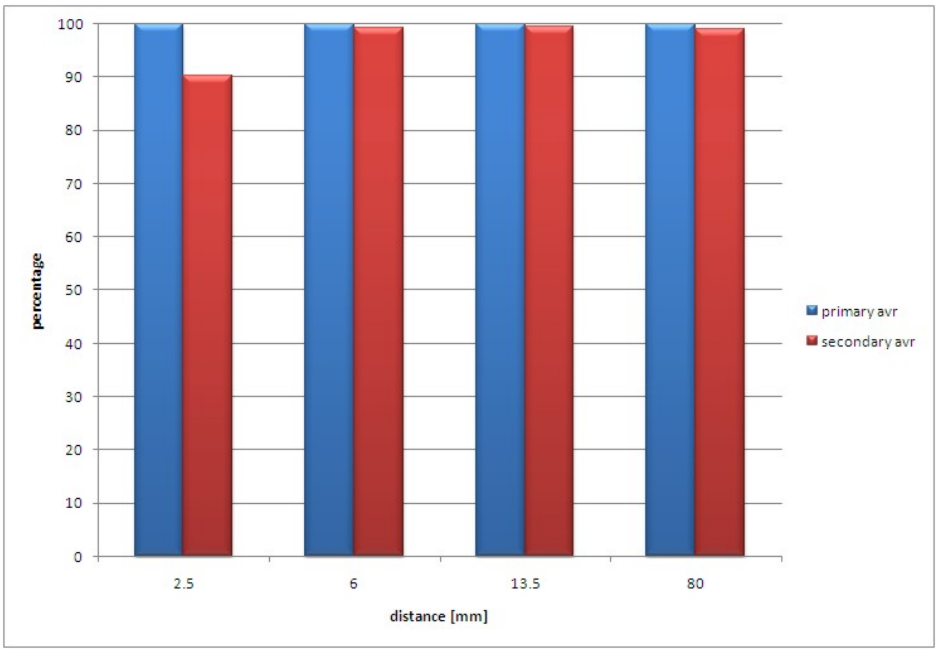
The performance of the original algorithm could in neither of the scenarios used for testing compete with the already optimized and thoroughly tested adaptive vertex fitter.

The graph in Fig. 4.1 (a) shows the amount of reconstructed and actually associated primary and secondary vertices (in percent) using the MVF reconstruction algorithm. It can clearly be seen that especially in the case of secondary vertex reconstruction the performance of the MVF is in no way competitive with the performance of the AVR reconstruction algorithm (Fig. 4.1 (b)). It can also be seen that the performance of the MVF drops when going to larger distances. This is a highly unintuitive behavior since the reconstruction should be facilitated at larger distances between the vertices.

In Fig 4.2 the position resolution in x direction (transverse to the beam direction) obtained with the MVF (left graph) is compared to the one obtained using the AVR. In both cases the resolution of the primary vertices are significantly better (nearly one order of magnitude) than for the secondary vertices. They both show a similar development going to larger distances although the resolution of the AVR is slightly better than in case of the MVF. Looking at the resolution of the secondary vertices the MVF again shows an unintuitive behavior since the resolution improves towards larger distances at first but gets worse again. In case of the AVR the resolution improves slightly but steadily with larger distances.

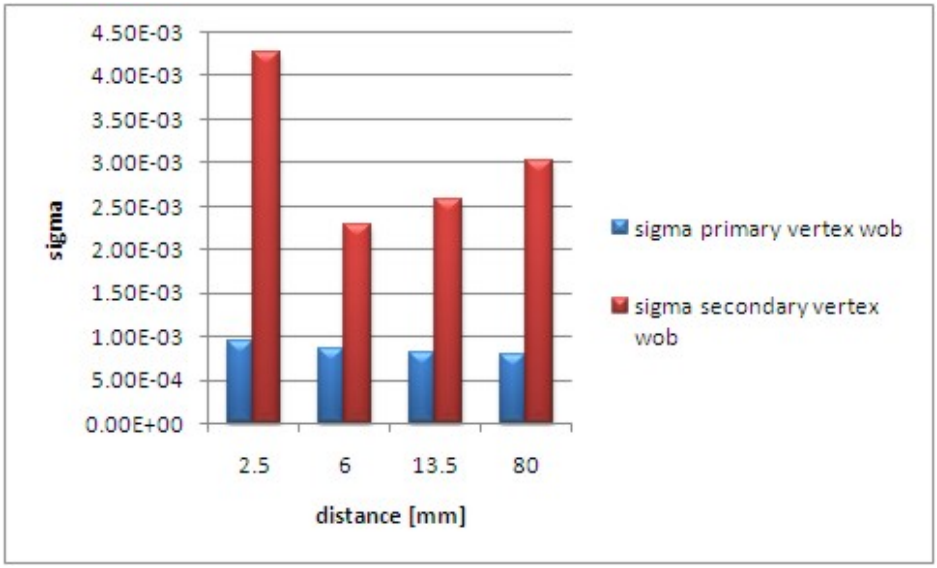


(a)

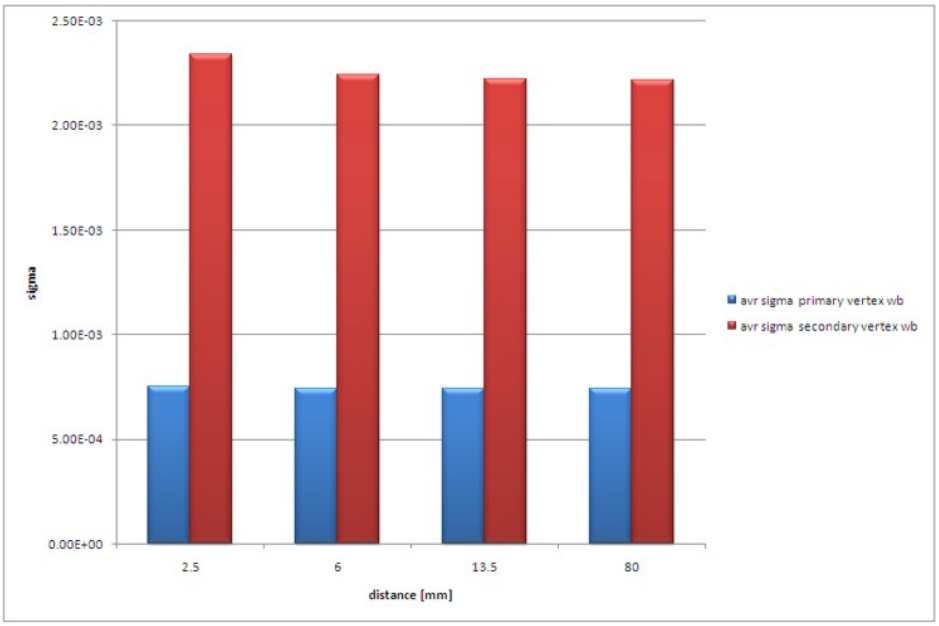


(b)

Figure 4.1: Performance of the MVF reconstruction algorithm (a) compared to the AVR algorithm (b).



(a)



(b)

Figure 4.2: Position resolution of the primary and secondary vertices obtained using MVF (a) and AVR (b)

4.4 Inclusion of Beamspot Constraint

The first means to improve the performance of the Multi Vertex Fitter was the inclusion of the beamspot constraint. This delivers the position of the primary vertex, with along covariance matrix. This information is derived from the beam interaction profile, depending on the accelerator itself. As mentioned before (see subsection 3.3.2) a good initial guess of the vertex position is crucial for any robust fitting method. So the inclusion of the beam spot constraint is expected to increase the performance of the MVF considerably.

4.4.1 Performance Studies

As can be seen in Fig 4.3, the inclusion of the beamspot constraint does not show any significant improvement in the performance of the MVF concerning the number of reconstructed and associable vertices. In Fig. 4.4 however the influence of the beamspot constraint is clearly visible. Not only an improvement in the position resolution of the primary vertex (Fig. 4.4 (a)) but also in the secondary vertex (Fig. 4.4 (b)) could be obtained.

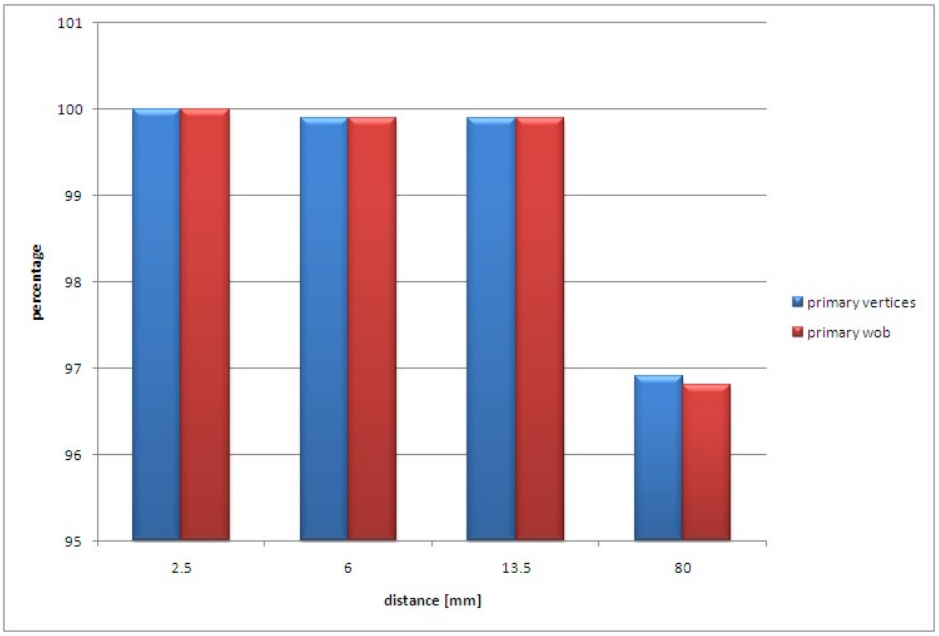
In order to gain more insight into the behavior of the MVF the author decided to evaluate the algorithm at larger distances.

Evaluation at larger distances showed that the number of associated and reconstructed primary vertices drops continuously (Fig. 4.5 (a)) while in case of the secondary vertices it reaches a minimum around a vertex distance of 150 mm (Fig. 4.5 (b)). At distances larger than 150 mm the number of associable reconstructed vertices rises again but does not exceed the number of such vertices at smaller distances. It can also be seen that this behavior is independent of the usage of the beamspot constraint, which again is very counterintuitive.

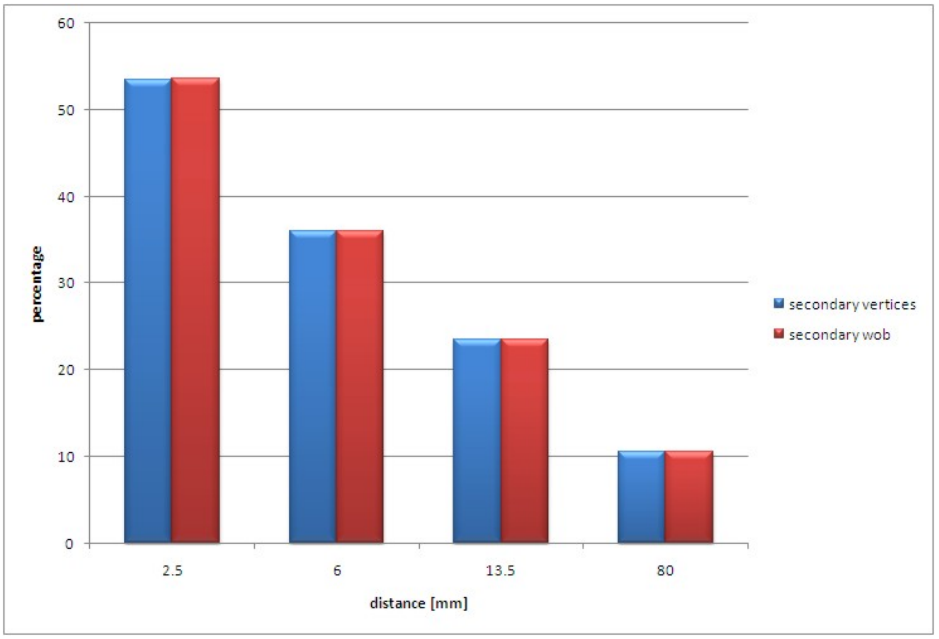
The position resolution again shows an improvement in case of the associated reconstructed primary vertices, whereas in the case of the secondary vertices no enhancement could be obtained (Fig. 4.6).

4.5 Comparison of Different Seeding Algorithms

As shown above, the inclusion of the beamspot constraint could not improve the number of associable reconstructed vertices. Only an improvement in the position resolution could be obtained. The counterintuitive behavior of the MVF with increasing distance however was not influenced. In order to

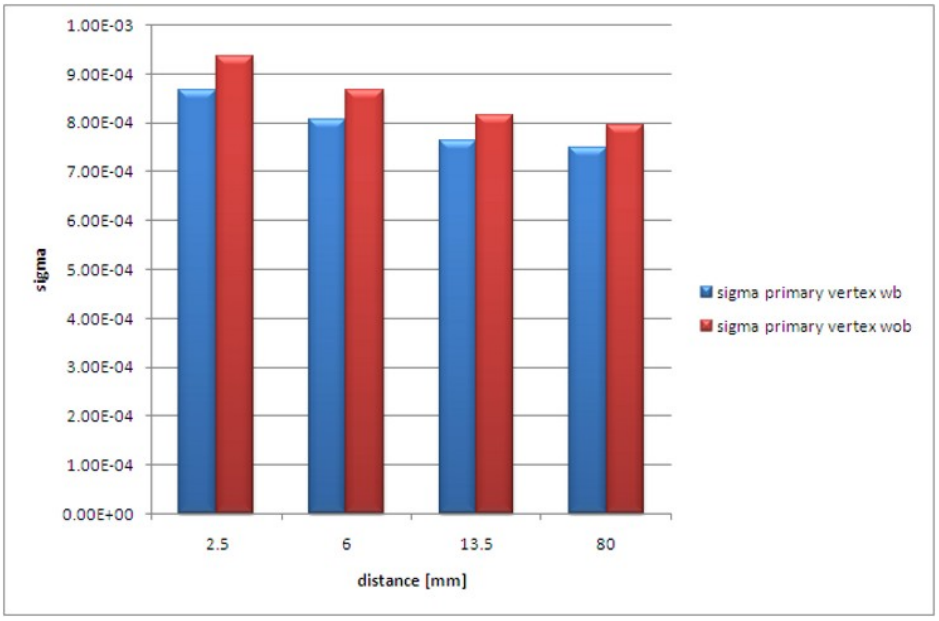


(a)

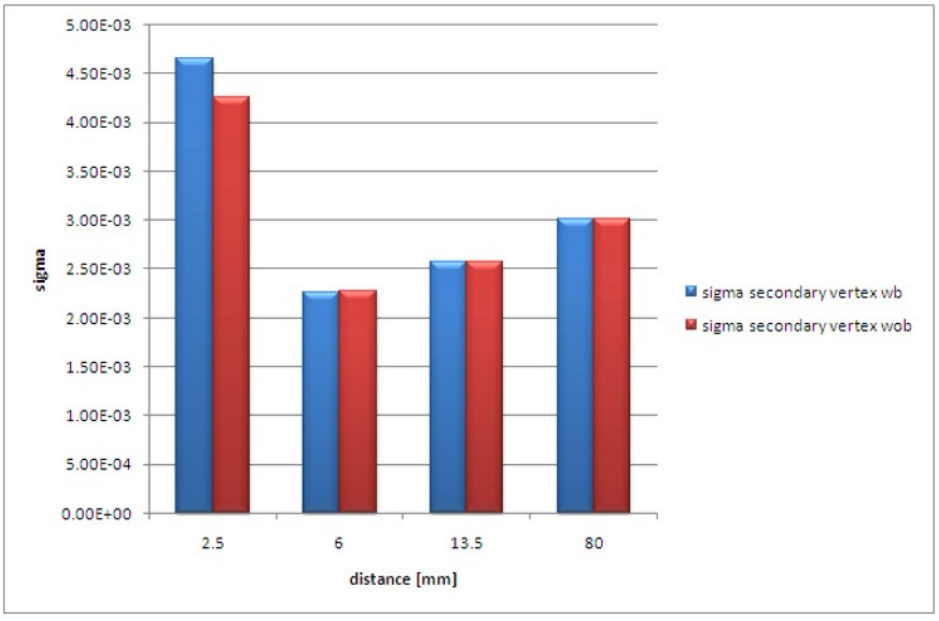


(b)

Figure 4.3: Comparison of the number of associated reconstructed primary (a) and secondary vertices (b) using the beamspot constraint and not using it.

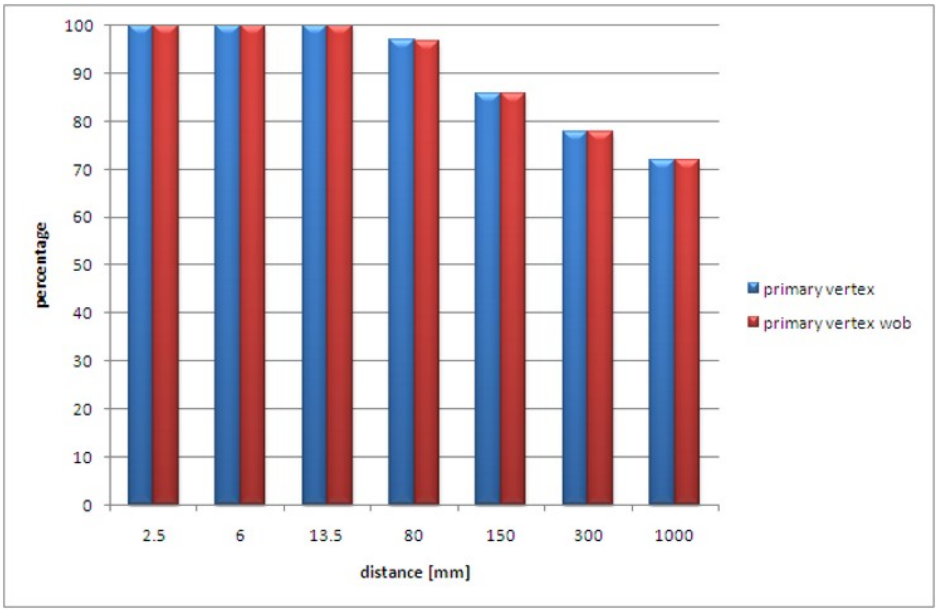


(a)

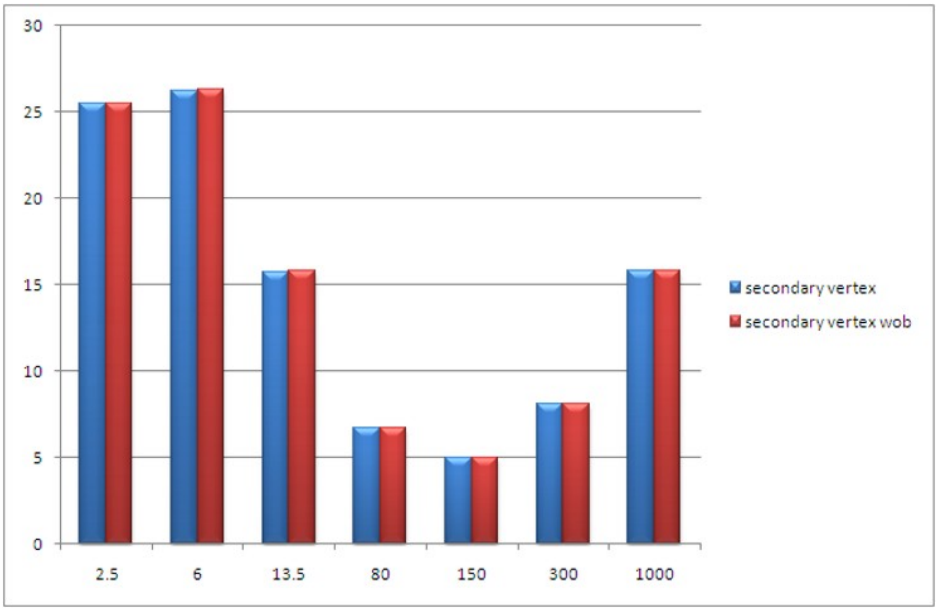


(b)

Figure 4.4: Comparison of the position resolution of the associated reconstructed primary (a) and secondary vertices using the beamspot constraint and not using it.

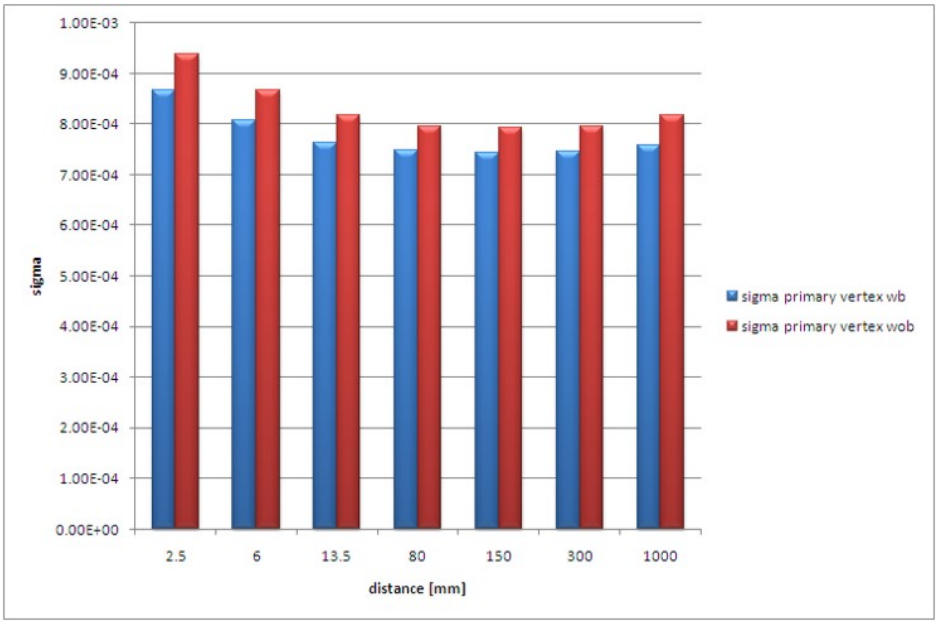


(a)

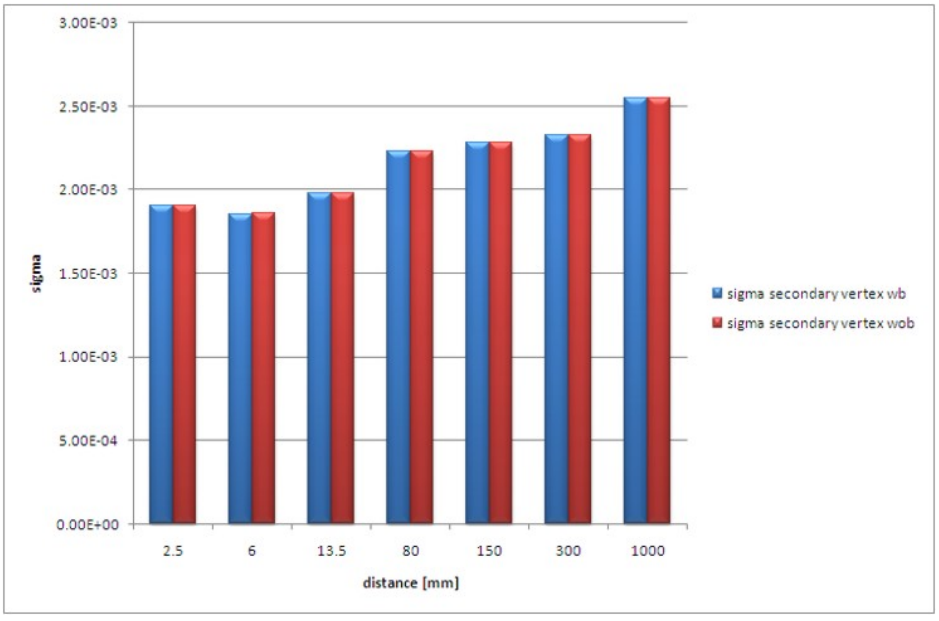


(b)

Figure 4.5: Comparison of the number of associated reconstructed primary (a) and secondary vertices (b) with and without using the beamspot constraint.



(a)



(b)

Figure 4.6: Comparison of the position resolution of the associated reconstructed primary (a) and secondary vertices with and without using the beamspot constraint.

explore this behavior the AVR was used as an alternative seeding algorithm for the MVF, and the result was compared to the outcome of the MVF using its default seeding algorithm, the MBS.

4.5.1 Performance Studies

In case of only four different distances as well as at seven different distances it can be seen that the improvement of the MVF using the AVR as seeding algorithm is huge. The number of associated reconstructed vertices is significantly increased in all scenarios. For the secondary vertex reconstruction it is even improved by an order of magnitude at certain distances (Fig. 4.7 and 4.8).

The position resolution is slightly increased in case of the primary vertex but for the secondary vertex an improvement is obtained(Fig. 4.9 and 4.10).

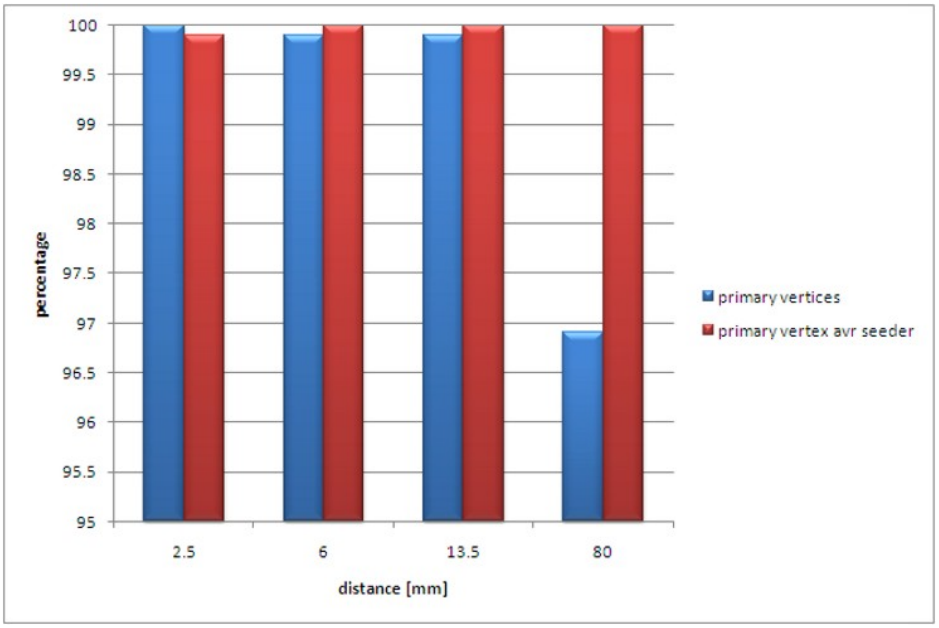
4.6 Test of the MBS Seeding Algorithm

Since using a different seeder did improve the performance of the MVF significantly, the functionality of the MBS seeding algorithm was tested and checked. Since the MBS is a reconstruction algorithm itself it could be tested in the same way as the MVF or AVR algorithms.

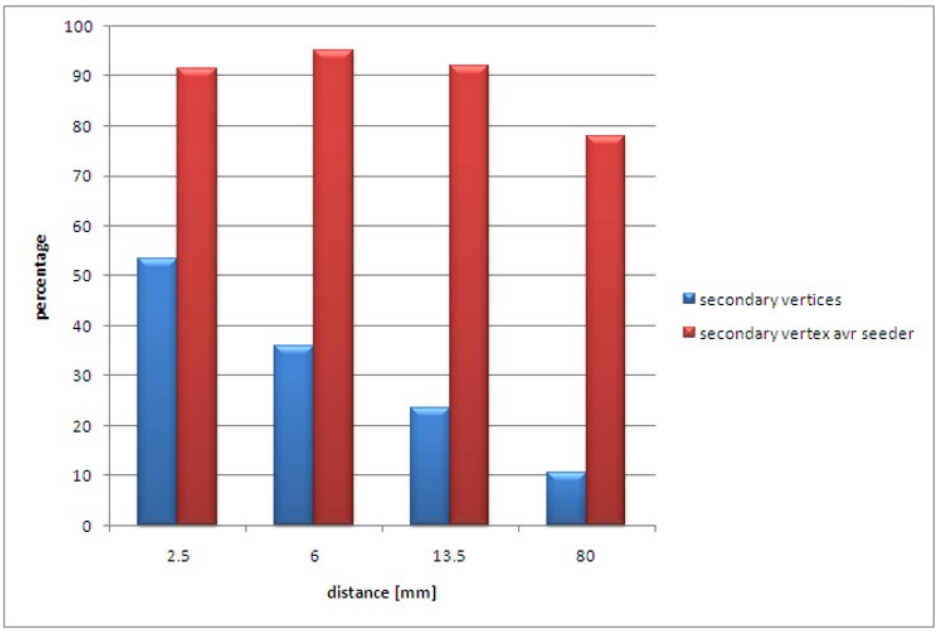
From Fig. 4.11 it is evident that the performance of the MBS is not satisfactory. The reconstruction of the primary vertices (Fig. 4.11 (a)) lies way below the reconstruction quality of the AVR. Not even for a single distance is the MBS able to associate all vertices. In case of the secondary reconstructed and associable vertices (Fig. 4.11 (b)) the MBS fails completely. In some cases not even a single secondary vertex is found (reconstructed and associated). Furthermore it can be seen that the unintuitive behavior already observed in case of the MVF is displayed here too and hence it can be assumed that the reason for it lies inside the MBS.

After thoroughly debugging the MBS the author found the reason for the bad performance of the algorithm inside the OutermostClusterizer method which was used as linearization point finder inside the MBS. This method should compute the points of closest approach between all tracks and should then pass on the two outermost points (impact points – IP's) along with the weighted tracks to a Kalman fitter. The result of the fit is then passed on to the MVF as initial guesses.

However, the method passed on two points that were very close around the primary vertex instead, even at a distance of 1 meter between the two simulated vertices. This of course explains nicely the reduction of recon-

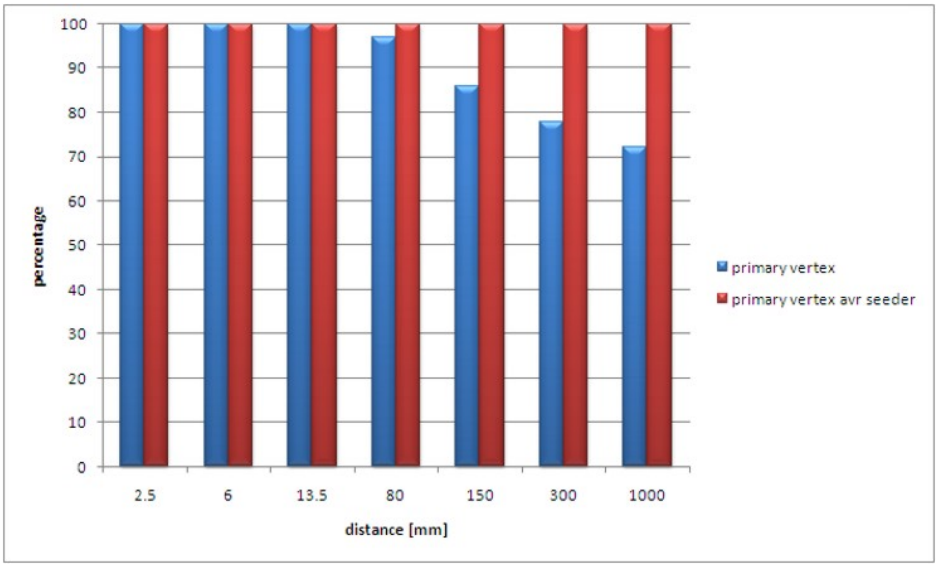


(a)

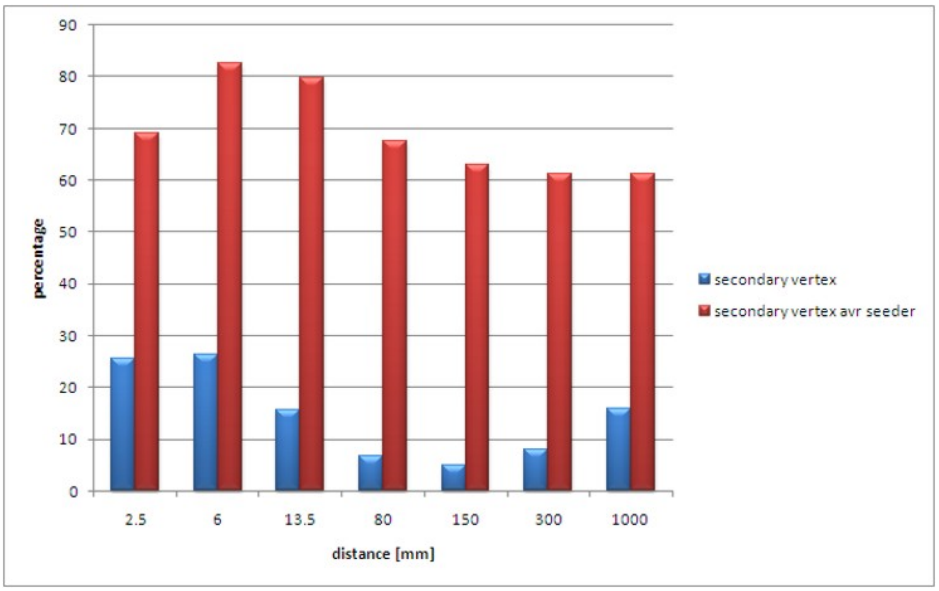


(b)

Figure 4.7: Comparison of the associated reconstructed primary (a) and secondary vertices (b) using the AVR (red) or the MBS (blue) as seeding algorithm.

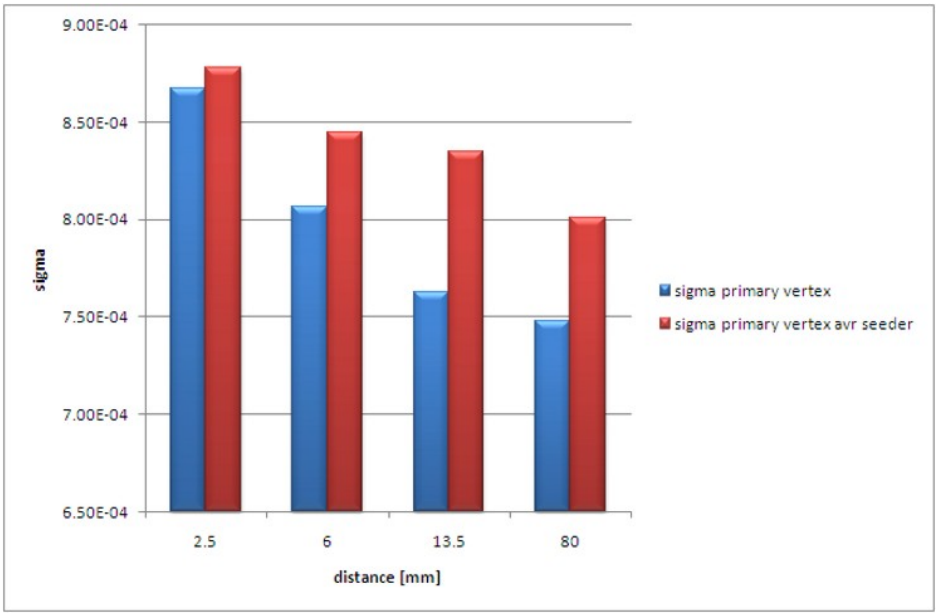


(a)

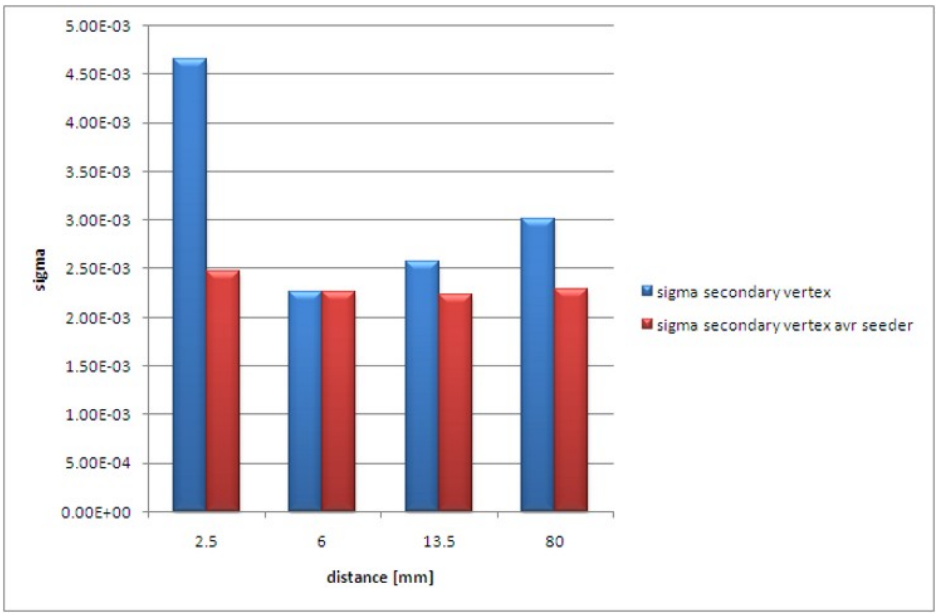


(b)

Figure 4.8: Same as 4.7 but for more distances.

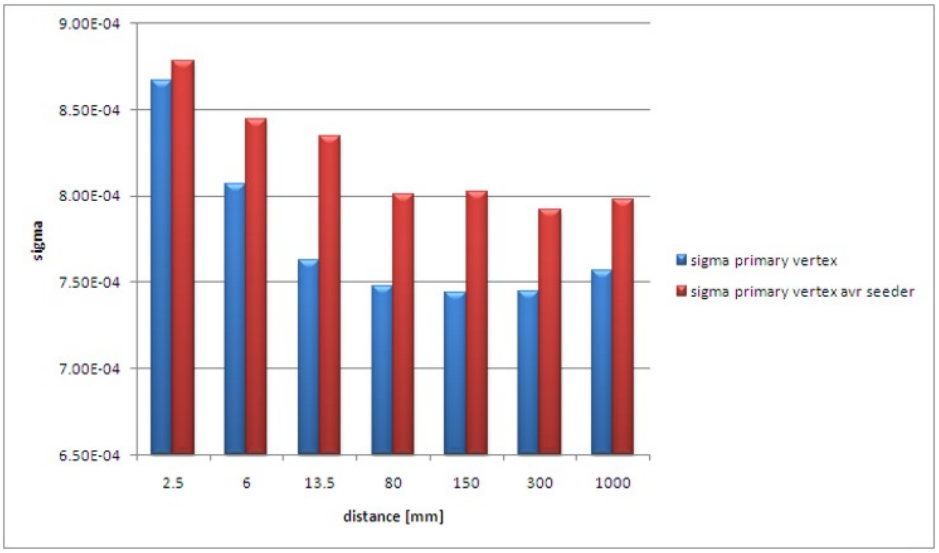


(a)

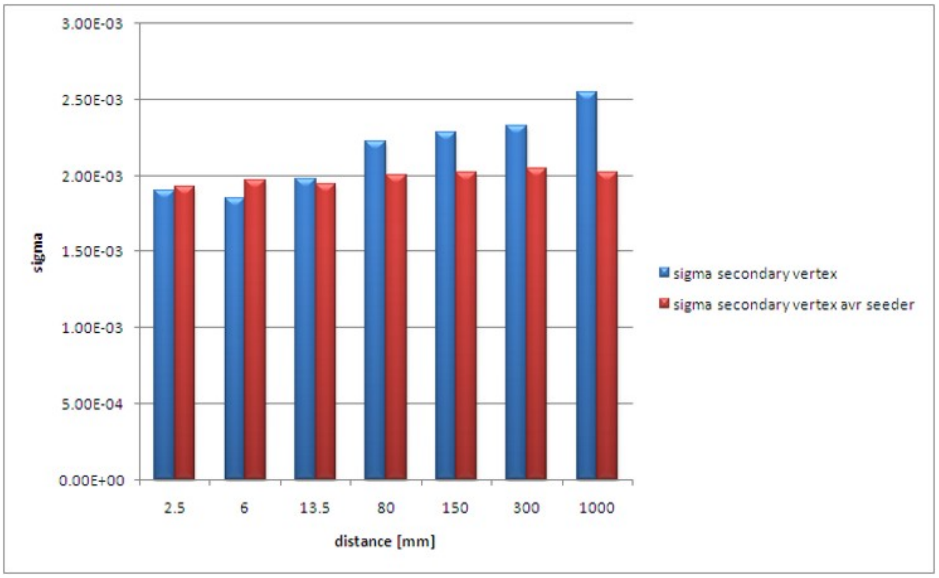


(b)

Figure 4.9: Position resolution of the associated reconstructed primary (a) and secondary vertices (b) using the AVR (red) or the MBS (blue) as seeding algorithm.

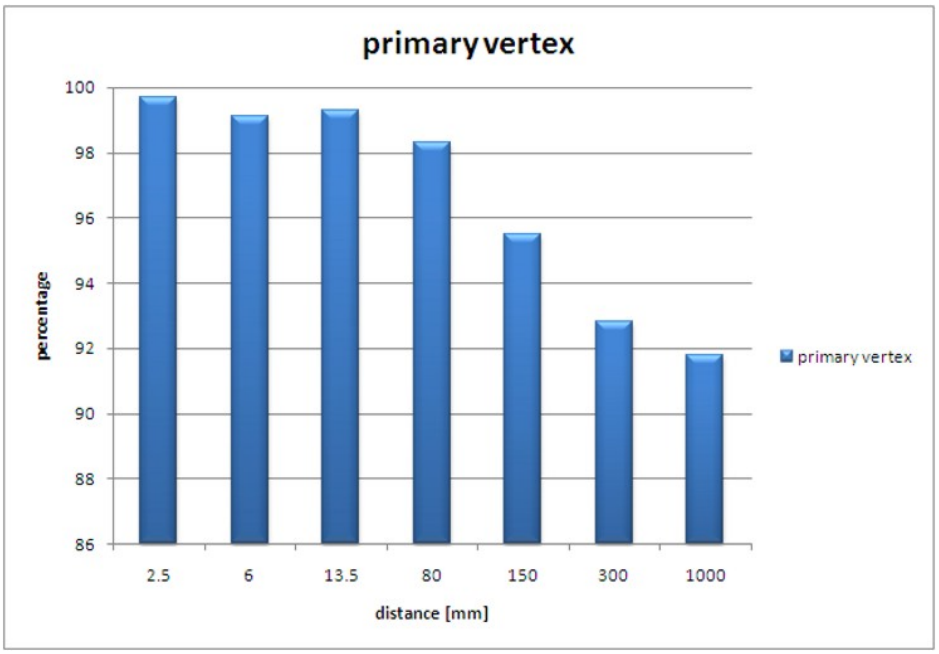


(a)

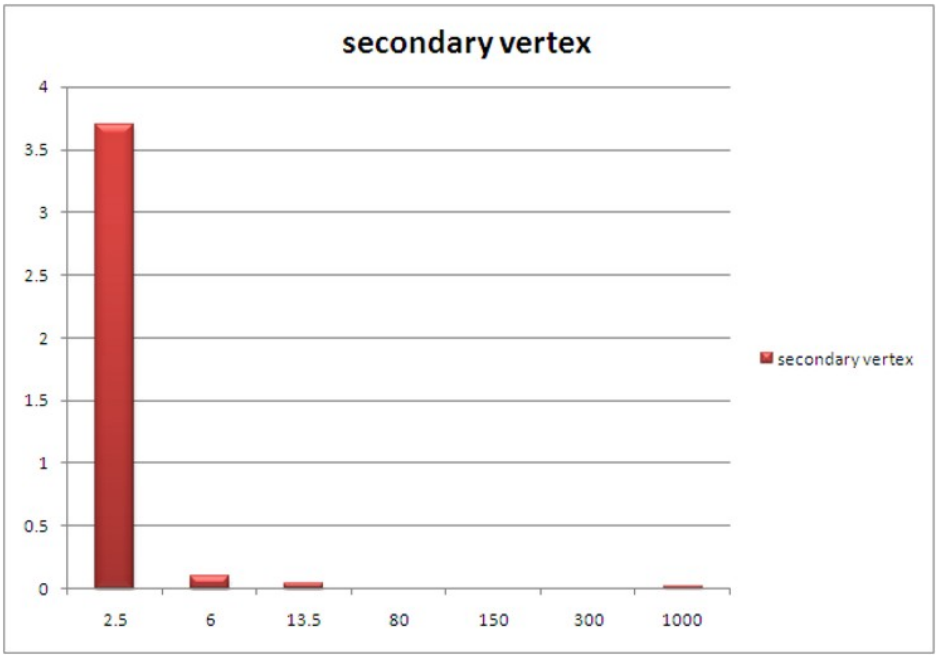


(b)

Figure 4.10: Same as 4.9 but for more distances.



(a)



(b)

Figure 4.11: The reconstructed and associated primary (a) and secondary (b) vertices using the MBS as reconstruction algorithm.

structed associated vertices when going to larger distances. The author fixed this problem by introducing a jet axis (ghost track). Details will be presented in the next section.

4.7 Inclusion of the Ghost Track Formalism

A ghost track simply is a jet axis originating in the primary vertex and going through the secondary vertex. The jet axis is directly produced by the guns implemented by the author. In case of real data, the direction and energy of the jet axis is extracted from data delivered by the calorimeter.

As mentioned above, the seeding algorithm of the MVF was found to deliver poor initial guesses. In order to improve these guesses, the impact points where now calculated using the ghost track.

Inside the MVF, the weights of the tracks with respect to this seed were calculated using the KalmanChiSquare method and the already implemented annealing schedule. The weighted tracks along with the initial seeds were then passed on as usual.

4.7.1 Performance Studies

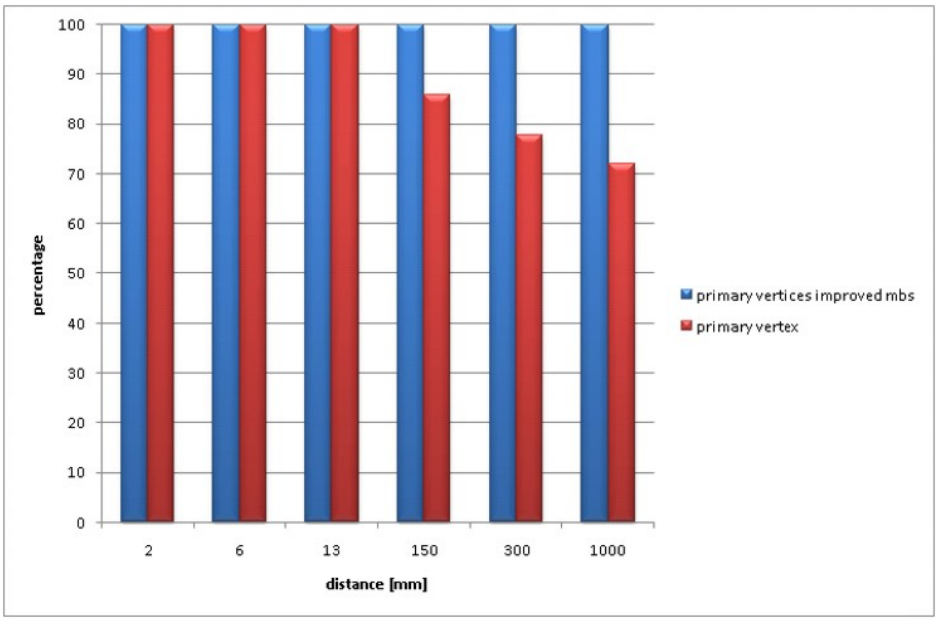
The performance of the MVF using the MBS seeder with the aforementioned implementations will be presented below.

The MVF using the improved MBS seeding algorithm clearly shows a steep increase in performance. The number of reconstructed and associated primary vertices is increased and the counterintuitive drop at larger distance disappears (Fig. 4.12 (a)). In case of the secondary reconstructed and associated vertices, the performance improvement is even more dramatic (Fig. 4.12 (b)). Furthermore no decrease going to larger vertex distances can be observed.

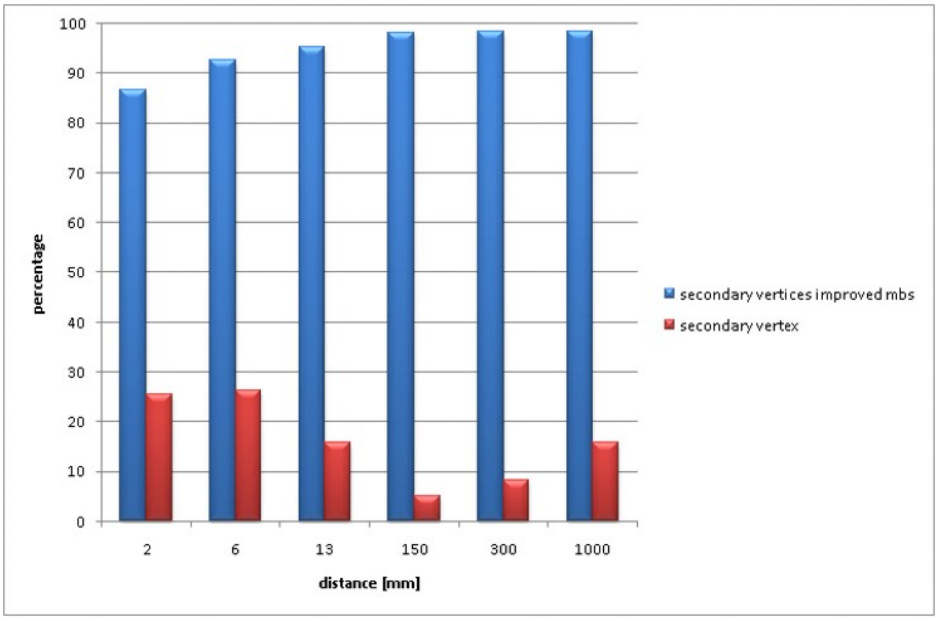
The position resolution is improved too, especially in the case of the primary vertices (Fig. 4.13). In case of the secondary vertices only an improvement at larger distances is observed. At smaller distances the position resolution is slightly deteriorated.

In Fig. 4.14 it can be seen that the MVF can not only compete but indeed beats the AVR in case of large distances. However in the interesting case of smaller distances, the improved MVF cannot outperform the AVR.

Comparing the position resolution of the original MVF to the MVF using the improved seeding algorithm and the AVR it can be seen that in case of the reconstructed and associated primary vertices the AVR delivers slightly

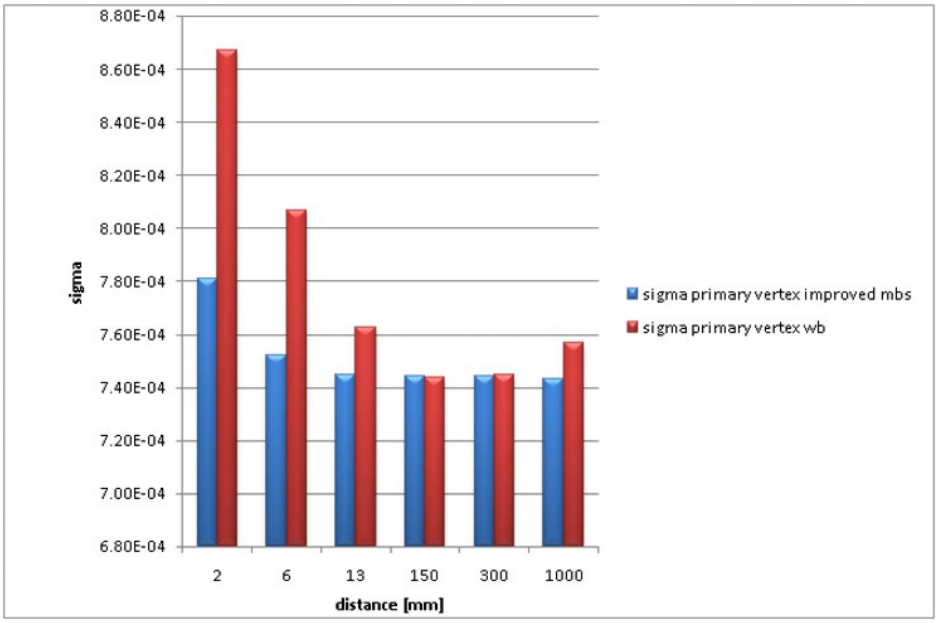


(a)

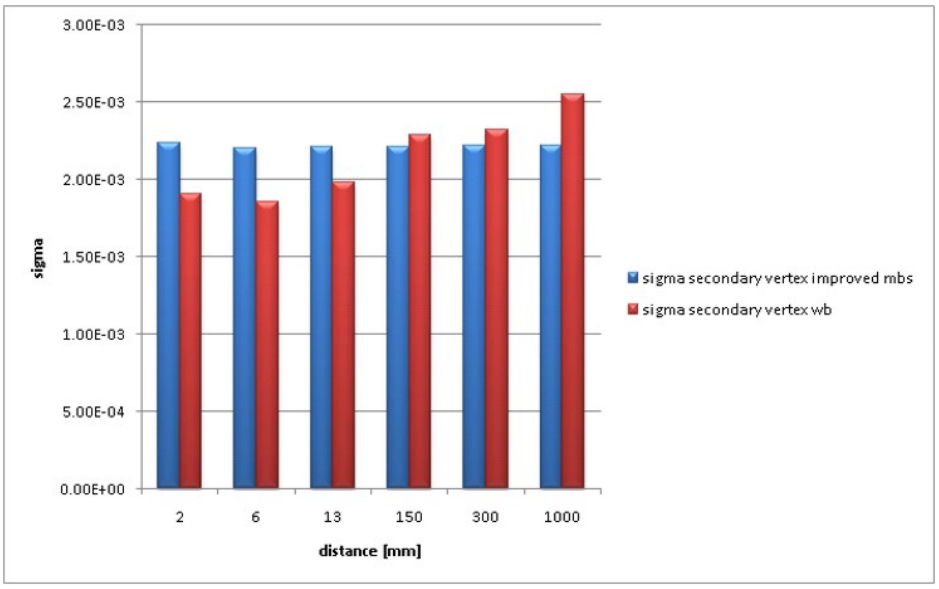


(b)

Figure 4.12: Comparison of the percentage of reconstructed and associated primary (a) and secondary vertices (b) between the former MVF and the MVF using the improved MBS.

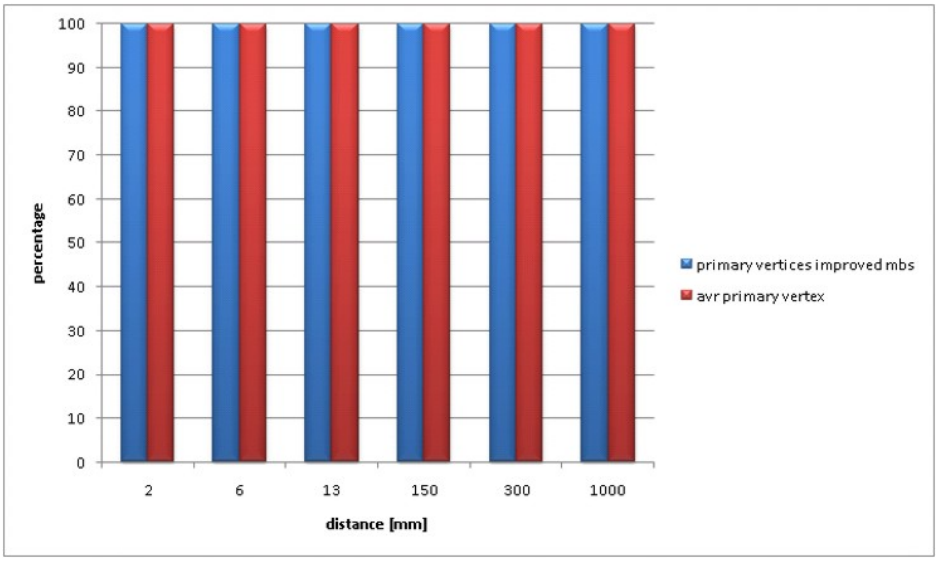


(a)

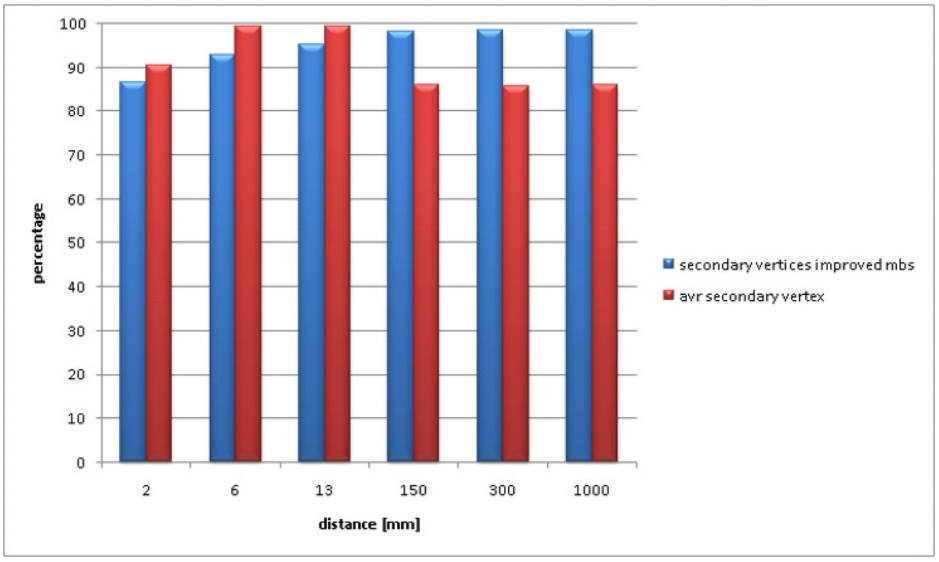


(b)

Figure 4.13: Comparison of the position resolution in the primary (a) and secondary vertices (b) between the former MVF and the MVF using the improved MBS.

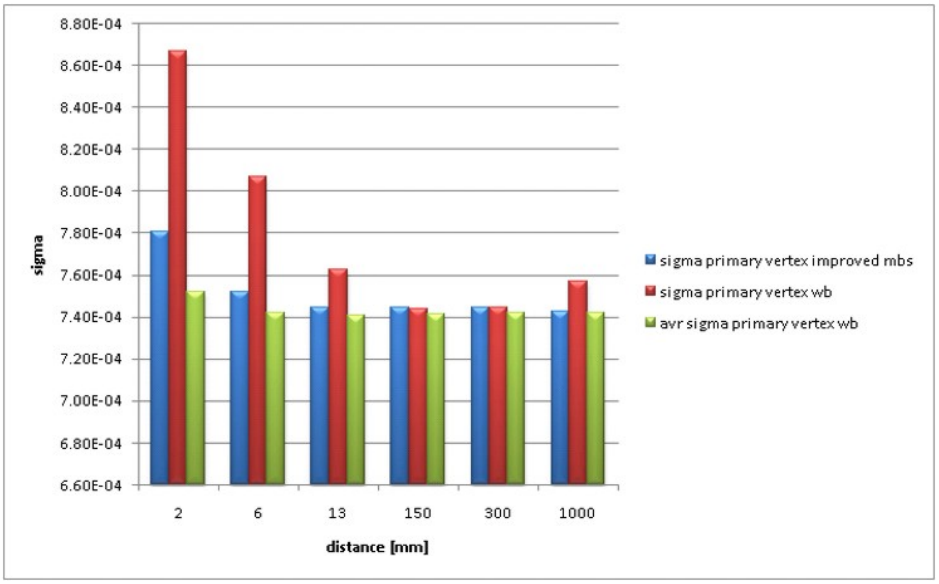


(a)

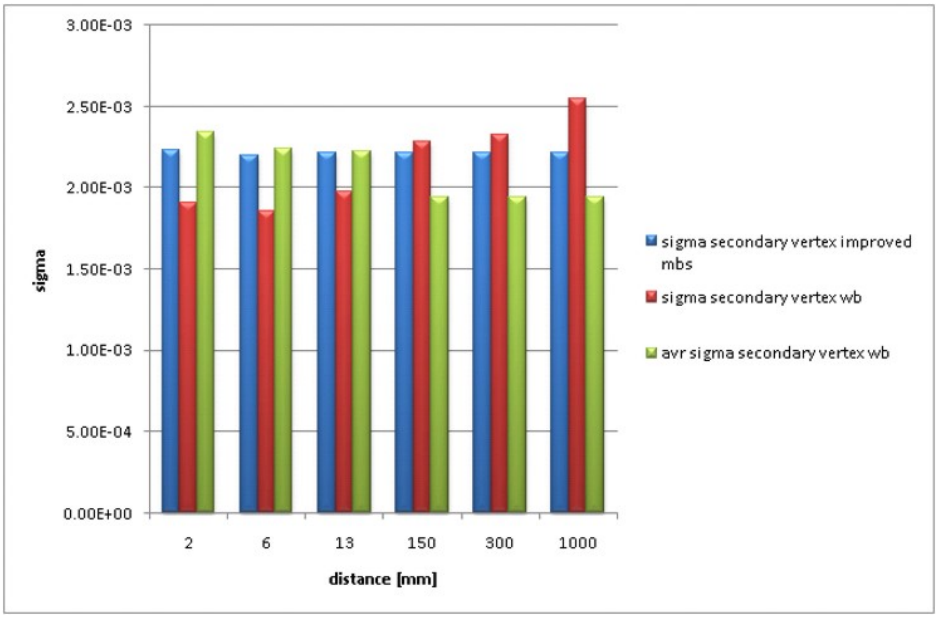


(b)

Figure 4.14: Comparison of the percentage of reconstructed and associated primary (a) and secondary vertices (b) between the MVF using the improved MBS and the AVR.



(a)



(b)

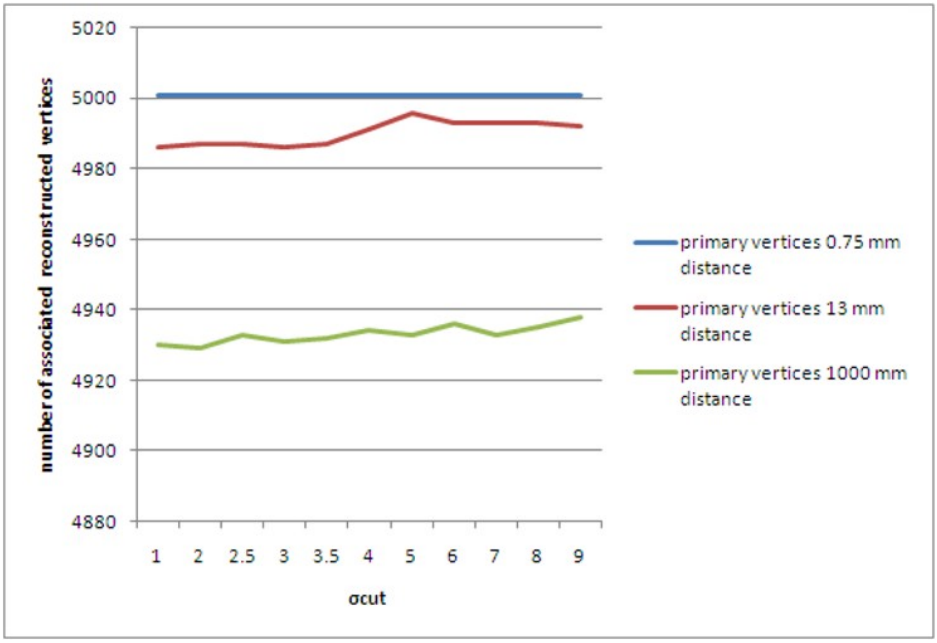
Figure 4.15: Comparison of the position resolution in the primary (a) and secondary vertices (b) between the MVF using the improved MBS and the AVR.

better results (Fig. 4.15 (a)). However the improved MVF shows better performance than the AVR considering the reconstructed associated secondary vertices. Only at large distances the AVR is slightly better again (Fig. 4.15 (a)).

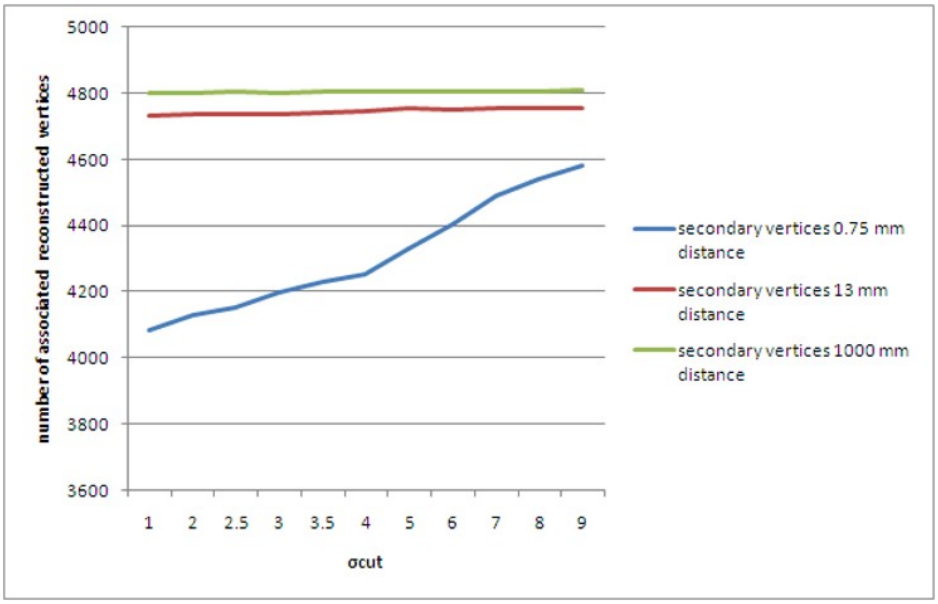
4.7.2 Adjustment of the σ_{cut}

The influence on the performance of different values for the σ_{cut} were now studied in order to find a suitable default value.

It can be seen that the number of reconstructed and associated primary and secondary vertices improves slightly when going to higher σ_{cut} values. This holds also true for the position resolution. Therefore the choice $\sigma_{cut} = 9$ as default value is justified.

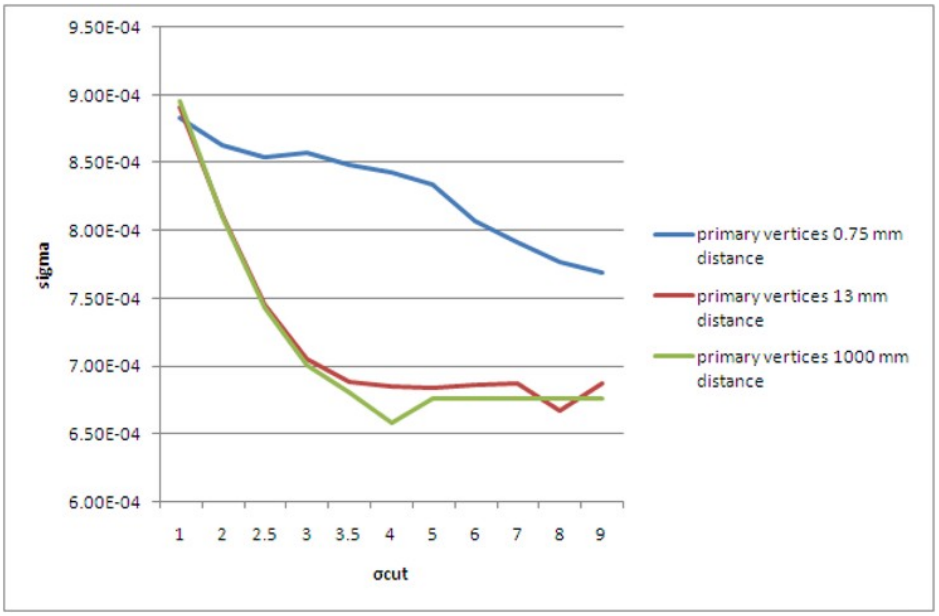


(a)

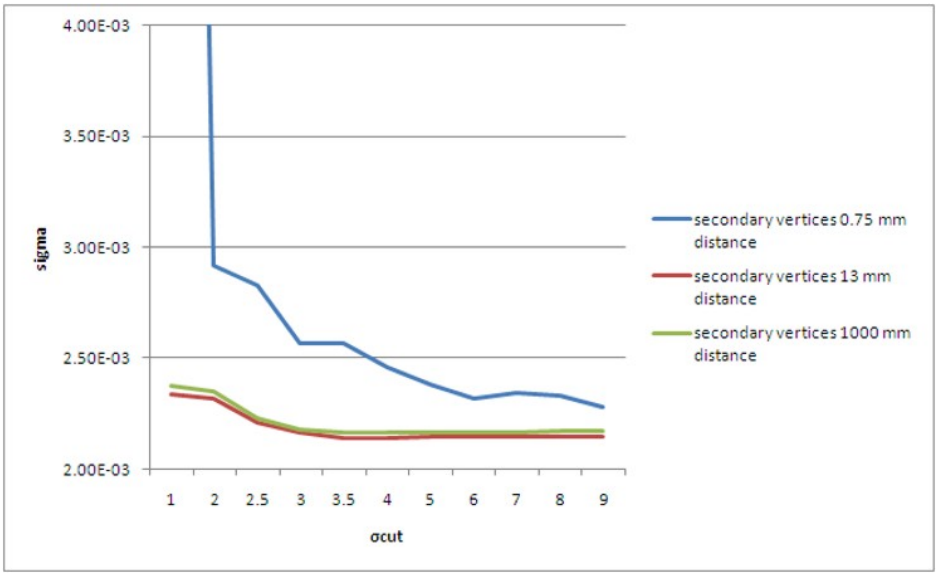


(b)

Figure 4.16: The influence of the σ_{cut} on the number of reconstructed and associated primary (a) and secondary vertices (b). It was studied for three different distances – 0.75 mm, 13 mm and 1 m.



(a)



(b)

Figure 4.17: The influence of the σ_{cut} on the position resolution of the primary (a) and secondary vertices (b). It was studied for three different distances – 0.75 mm, 13 mm and 1 m.

Chapter 5

Conclusion

The poor initial performance of the Multi Vertex Fitter algorithm could be identified as a consequence of the malfunctioning of a method used inside the MBS seeding algorithm. This caused the passing on of the wrong initial seeds from the MBS to the MVF. A considerable improvement in the performance of the MVF could be obtained after this problem was eliminated.

Nevertheless, the improvement thus obtained could increase the performance of the MVF over that of the AVR only in case of large distances between the two vertices. In the relevant case of small distances the improved MVF does not give better results than the AVR. Also, since Monte Carlo truth in form of a ghost track was used for improving the MBS, and consequently the MVF, it cannot be followed that the MVF will lead to better results than the AVR when handling real data.

Future research may include introducing more realistic information, e. g. in form of a ghost track with errors, to the MFV algorithm; while exceeding the scope of this thesis the principle of the MFV still holds interesting research possibilities.

Acknowledgments

First of all I want to thank my family for making it possible for me to follow my studies, never complaining or even mentioning the great personal efforts and austerity it caused them. Thank you for always supporting me 100%. Secondly I want to thank my supervisor Univ. Doz. DI Dr. R. Frühwirth and DI Dr. W. Waltenberger for their help and many valuable comments and discussions. I also want to thank all my colleagues and fellow students not only for countless fruitful discussions but also for not only conserving but increasing my interest in physics. Especially I want to thank M. Bartel for always helping me with the small and not so small problems that occurred writing this work. Thank you for your patience. Furthermore thanks to everybody that helped me and accompanied me on my way so far, be it academically or privately. You are the reason I have become the person that I am today.

Bibliography

[1] G. Alverson, G. Eulisse, S. Muzaffar, I. Osborne, L. Tuura and L. Taylor, The IGUANA Interactive Graphics Toolkit with Examples from CMS and D0. Proceedings of the 2003 Conference on Computing in High Energy and Nuclear Physics. http://www.slac.stanford.edu/econf/C0303241/proc/papers/MOLT008.PDF

[2] G. Bagliesi, on behalf of the CMS collaboration. CMS High Level Trigger Selection. Eur. Phys. J. C 33, Supplement 1 (2006) s1035. DOI: 10.1140/epjcd/s2004-03-1804-1.

[3] S. Brandt, Datenanalyse. BI Wissenschaftsverlag, Mannheim, 1992.

[4] T. Boccali. The Geant4 Simulation in CMS: OSCAR. Pisa, 2002. http://moby.mib.infn.it/~cmsweb/old_site/lectures/NicolaTommaso/OSCARTutorial_Boccali.pdf

[5] CERN. www.cern.ch

[6] CERN School of Computing, Gjøvik, Norway, 2008

[7] CMSDOC. http://cmsdoc.cern.ch/cms.html

[8] The CMS experiment at the CERN LHC. The CMS Collaboration, S. Chatrchyan *et al*. JINST 3 (2008) S08004. DOI: 10.1088/1748-0221/3/08/S08004.

[9] CMS Homepage. http://cms.cern.ch/

[10] CMS Software. http://cmsdoc.cern.ch/cms/cpt/Software/html/General/

[11] CMS Software Guide. https://twiki.cern.ch/twiki/bin/view/CMS/SWGuidePreface

[12] CMSSW Application Framework. https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookCMSSWFramework

[13] CMS, The TriDAS Project. Technical Design Report, Volume 1: The Trigger systems. CERN/LHCC 2000-38, CERN, Geneva, 2000. http://cmsdoc.cern.ch/cms/TDR/TRIGGER-public/CMSTrigTDR.pdf

[14] The CMS Trigger and Data Acquisition Group. The CMS High Level Trigger. Eur. Phys. J. C 46 (2006) 605.

[15] Lyndon Evans and Philip Bryant (eds). LHC Machine. JINST 3 (2008) S08001 DOI: 10.1088/1748-0221/3/08/S08001.

[16] R. Frühwirth. Statistische Methoden der Datenanalyse. Vorlesungsunterlagen, Wien, 2008. http://wwwhephy.oeaw.ac.at/u3w/f/fru/www/vorlesung/Folien_SS2008.pdf

[17] R. Frühwirth, P. Kubinec, W. Mitaroff, M. Regler. Vertex reconstruction and track bundling at the LEP collider using robust algorithms. Comput. Phys. Commun. 96 (1996) 189.

[18] R. Frühwirth, M. Regler, R. Bock, H. Grote and D. Notz, Data Analysis Techniques for High–Energy Physics. Cambridge University Press, Cambridge, 2000.

[19] R. Frühwirth, A. Strandlie, T. Todorov, M. Winkler. Recent Results on Adaptive Track and Multitrack Fitting in CMS. Nucl. Instrum. and Methods A 502 (2003) 702.

[20] A. Gersho and R. M. Gray. Vector Quantization And Signal Compression. Kluwer, Boston, 1992.

[21] IGUANA Homepage. http://iguana.web.cern.ch/iguana/

[22] Institut für Hochenergiephysik der Österreichischen Akademie der Wissenschaften. http://www.hephy.at/

[23] K. Klein. The CMS Silicon Strip Tracker — Overview and Status. Proceedings of the 2005 International Europhysics Conference on High Energy Physics. http://arxiv.org/ftp/physics/papers/0610/0610259.pdf

[24] M. Krammer. Detektoren in der Hochenergiephysik. Lecture notes, summer term 2007.

[25] S. Kundu. Gravitational clustering: A new approach based on the spatial distribution of the points. Pattern Recognition 32 (1999) 1149.

[26] The Large Hadron Collider. http://lhc-machine-outreach.web.cern.ch/lhc-machine-outreach/

[27] LHC Technical Design Report. http://ab-div.web.cern.ch/ab-div/Publications/LHC-DesignReport.html

[28] W. R. Leo, Techniques for Nuclear and Particle Physics Experiments A How–to Approach Second Revised Edition. Springer, New York Berlin Heidelberg, 1994.

[29] S. Qian. Letter to the Editor A steering procedure for the fast vertex fitting method. Nuclear Instruments and Methods in Physics Research A 350 (1994) 618.

[30] RAVE Users Guide. http://projects.hepforge.org/rave/trac/wiki/UserGuide

[31] The RAVE/VERTIGO Vertex Reconstruction Toolkit. http://wwwhephy.oeaw.ac.at/p3w/ilc/talks/06_SiLC_Liverpool/WM_RaveVertigo.pdf

[32] M. Regler, R. Frühwirth, W. Mitaroff. Filter Methods in Track and Vertex Reconstruction. Int. J. Mod. Phys. C 7 (1996) 521.

[33] K. Rose. Deterministic annealing for clustering, compression, classification, regression and related optimization problems. Proc. IEEE 86 11 (1998) 2210.

[34] K. Rose, E. Gurewitz, G.C. Fox. Statistical mechanics and phases transitions in clustering. Phys. Rev. Lett., 65 (1990) 8 945.

[35] J. Sammet, Ausarbeitung des Vortrags LHC Beschleuniger aus der Reihe Hadron–Kollider–Experimente bei sehr hohen Energien, 2006. http://web.physik.rwth-aachen.de/~hebbeker/lectures/sem0607/sammet_ausarbeitung.pdf

[36] W. Waltenberger. Development of Vertex Finding and Vertex Fitting Algorithms for CMS. Dissertation TU Wien, 2004.

[37] W. Waltenberger *et al.* The RAVE/VERTIGO vertex reconstruction toolkit and framework. J. Phys.: Conf. Ser. 119 (2008) 032037. DOI: 10.1088/1742-6596/119/3/032037

[38] W. Waltenberger, R. Frühwirth, P. Vanlaer. Adaptive vertex fitting. J. Phys. G: Nuclear and Particle Physics 34 (2007) N1.

Riedel. The RAVE/VERTIGO vertex reconstruction toolkit and framework. Wien, 2008

List of Figures

Curriculum Vitae

Name	Silke Federmann
Geburtsdatum	12. 01. 1979
Familienstand	verheiratet seit 23. 01. 1999
Adresse	Steudelgasse 42 1100 Wien
Berufserfahrung	1999 bis 2009 tätig als Nachhilfelehrerin im Nachhilfeinstitut ÖKLV 2003 bis 2004 freiberuflich tätig bei OVB Wien 1996 und 1997 Ferialpraxis im chemischen Labor der Enz–Caro Metallwerke
Ausbildung	2007 elf Wochen Praktikum am CERN als Sommer Studentin 2004 bis 2009 Diplomstudium Physik an der Universität Wien 1997 bis 2004 Lehramtsstudium Chemie/PPP an der Universität Wien, dann Studienrichtungswechsel auf Diplom Physik 1989 bis 1997 BG/BRG Baden Biondekgasse 1985 bis 1989 Volksschule Enzesfeld-Lindabrunn
Sprachen	Englisch fließend Französisch Grundkenntnisse