

┌

└

ATLAS Muon Reconstruction

**from a
C++ Perspective**

A Road to the Higgs

└

┌

┌

┐

└

┘

┌

ATLAS Muon Reconstruction

**from a
C++ Perspective**

A Road to the Higgs

ACADEMISCH PROEFSCHRIFT

TER VERKRIJGING VAN DE GRAAD VAN DOCTOR
AAN DE UNIVERSITEIT VAN AMSTERDAM
OP GEZAG VAN DE RECTOR MAGNIFICUS PROF. DR. J.J.M. FRANSE
TEN OVERSTAAN VAN EEN DOOR HET COLLEGE VOOR PROMOTIES INGESTELDE COMMISSIE
IN HET OPENBAAR TE VERDEDIGEN IN DE AULA DER UNIVERSITEIT
OP WOENSDAG 26 APRIL 2000 TE 10:00 UUR

door

Patrick John Hendriks
geboren te Nijmegen

└

Promotor : **Prof. Dr. F.L. Linde**
Universiteit van Amsterdam (UVA)

Co-Promotor : **Dr. K. Bos**
NIKHEF, Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica.

The work described in this thesis is part of the research programme of “het Nationaal Instituut voor KernFysica en Hoge-Energie Fysica (NIKHEF)” in Amsterdam. The author was financially supported by “de Stichting voor Fundamenteel Onderzoek der Materie (FOM)”.

Table Of Contents

| | | |
|------------------|---|-----------|
| | Introduction | 1 |
| CHAPTER 1 | The Standard Model | 3 |
| | 1.1 Elementary Particles and their Interactions | 3 |
| | 1.2 The Higgs Mechanism | 4 |
| | 1.3 Higgs Creation | 5 |
| | 1.4 Higgs Decay Channels | 6 |
| CHAPTER 2 | The ATLAS Experiment | 9 |
| | 2.1 The LHC | 9 |
| | 2.2 The ATLAS Detector | 10 |
| | 2.3 The Muon Spectrometer | 12 |
| | 2.4 ATLAS Computing | 15 |
| | 2.4.1 Object Orientation | 15 |
| | 2.4.2 Domain Decomposition | 17 |
| | 2.4.3 Arve | 19 |
| CHAPTER 3 | Software Design | 21 |
| | 3.1 Global Architecture | 21 |
| | 3.2 AMBER | 22 |
| | 3.2.1 Integration into Arve | 23 |
| | 3.2.2 Event | 25 |
| | 3.2.3 Detector Description | 27 |
| | 3.2.4 Graphics | 29 |
| | 3.3 Detector Reconstruction Toolkit | 31 |
| | 3.3.1 The Track Package | 31 |
| | 3.3.2 Track Propagation in a Magnetic Field | 36 |
| | 3.4 Generic Dataview Library | 38 |
| | 3.4.1 Core Implementation | 40 |
| | 3.4.2 Toolkit | 42 |
| | 3.5 Conclusion | 46 |

| | | |
|------------------|--|-----------|
| CHAPTER 4 | Reconstruction Algorithm | 49 |
| 4.1 | Trigger Chamber Reconstruction | 50 |
| 4.1.1 | The RPC Chambers | 50 |
| 4.1.2 | The TGC Chambers | 53 |
| 4.1.3 | Low- p_T Trigger | 54 |
| 4.1.4 | High- p_T Trigger | 56 |
| 4.2 | MDT Pattern Recognition | 57 |
| 4.2.1 | Local MDT Reconstruction | 57 |
| 4.2.2 | Pattern Recognition | 59 |
| 4.2.3 | Drift-Circle Fit | 61 |
| 4.3 | Global Reconstruction | 64 |
| 4.4 | The Global Fit | 65 |
| | | |
| CHAPTER 5 | Single-Chamber Performance | 69 |
| 5.1 | Simulation Environment | 69 |
| 5.2 | Reconstruction Algorithm | 71 |
| 5.3 | Reconstruction Efficiency | 71 |
| 5.4 | Fit Accuracy | 73 |
| 5.5 | Single-Tube Resolution | 74 |
| | | |
| CHAPTER 6 | DATCHA | 77 |
| 6.1 | Introduction | 77 |
| 6.1.1 | Data Runs | 80 |
| 6.1.2 | Event Simulation | 81 |
| 6.2 | Calibration | 81 |
| 6.2.1 | Time-of-flight Correction | 82 |
| 6.2.2 | Leading and Trailing Edges | 83 |
| 6.2.3 | R-T Calibration | 85 |
| 6.2.4 | Signal Propagation Velocity | 88 |
| 6.3 | Reconstruction | 90 |
| 6.3.1 | Single-Tube Efficiency | 91 |
| 6.3.2 | Segment Reconstruction | 93 |
| 6.3.3 | Segment Matching | 94 |
| 6.3.4 | Sagitta Measurement | 95 |
| 6.4 | Conclusion | 97 |
| | | |
| CHAPTER 7 | Muon Reconstruction Performance | 99 |
| 7.1 | Single Muons | 100 |
| 7.1.1 | Resolution | 102 |
| 7.1.2 | Efficiency | 106 |
| 7.1.3 | Charge Identification | 107 |

| | | |
|-------------------|--|------------|
| 7.2 | $Z \rightarrow \mu^+ \mu^-$ | 108 |
| 7.3 | $H \rightarrow ZZ^{(*)} \rightarrow \mu^+ \mu^- \mu^+ \mu^-$ | 109 |
| 7.4 | Conclusion and Outlook | 111 |
| CHAPTER 8 | Higgs to 4 Lepton Decay | 113 |
| 8.1 | Signal Reconstruction | 113 |
| 8.1.1 | $H \rightarrow ZZ^{(*)} \rightarrow e^+ e^- e^+ e^-$ | 114 |
| 8.1.2 | $H \rightarrow ZZ^{(*)} \rightarrow \mu^+ \mu^- \mu^+ \mu^-$ | 115 |
| 8.1.3 | $H \rightarrow ZZ^{(*)} \rightarrow e^+ e^- \mu^+ \mu^-$ | 115 |
| 8.2 | Background | 115 |
| 8.2.1 | Irreducible Background | 116 |
| 8.2.2 | Reducible Background | 117 |
| 8.3 | Conclusion | 119 |
| APPENDIX A | Notation | 121 |
| A.1 | Unified Modelling Language | 121 |
| A.1.1 | Package Diagrams | 122 |
| A.1.2 | Class Diagrams | 122 |
| A.1.3 | Interaction Diagrams | 124 |
| A.2 | Dataview Diagrams | 125 |
| APPENDIX B | Software Implementation | 127 |
| APPENDIX C | Glossary | 129 |
| | References | 133 |
| | Summary | 137 |
| | Samenvatting | 139 |
| | Acknowledgements | 141 |

┌

┐

└

┘

Introduction

Where do you want to go today?

Microsoft advertising slogan

For the past two decades Fortran 77 has been the dominant programming language in the high-energy physics community. At the time it was first introduced the typical experiments were small, both in terms of the physical size of the detectors and in the number of people working on them. But with every new generation of experiments both grew in size, placing ever more stringent demands not only on the detectors' hardware, but also on the software needed to simulate, reconstruct and analyse their physics events.

As a consequence, advancements in detector technology have been continuous and have led, among other things, to better resolutions and faster response times. Surprisingly however, any upgrades to the software development process have been few and far between. Despite major changes in the "real" world, the language of choice has remained to be Fortran. It wasn't until very recently that this anachronism was acknowledged and efforts were undertaken to bring the latest software techniques to the high-energy physics community.

The one technique that has had the largest impact has been the change from procedural to object-oriented programming, with the complementary adaptation of C++ as the implementation language. The first steps along this line were taken about 5 years ago, but a truly widespread acceptance has happened only much more recently.

The work described in this thesis is aimed at designing, implementing and testing a full-size object-oriented program. After some consideration, the choice was made to work on the reconstruction of events in the muon spectrometer of the ATLAS detector¹. The standard steps to take in the creation of such a program are [1]:

- **Analysis.** Chapter 1 describes the Standard Model and the theoretical foundation of the Higgs mechanism. Finding and studying the Higgs boson(s) is one of the major goals of the ATLAS detector. The experimental setup is described in chapter 2, in which special emphasis is placed on the muon spectrometer and the software infrastructure available within ATLAS.

1. The various acronyms used in this thesis are explained in the glossary (appendix C).

- **Design.** A detailed description of the design of the muon reconstruction software is given in chapter 3. First the various ATLAS-specific subdomains like the detector description and the event representation are explored. They are followed by an account of two general-purpose packages that are used to implement the reconstruction algorithm. The first is the Detector Reconstruction Toolkit, which defines general reconstruction classes like tracks and error cones, and performs tasks like track fitting and track propagation through a magnetic field. The second package is the Generic Dataview Library, which provides a framework in which data-driven algorithms can be implemented in a straightforward and intuitive way.

The reconstruction algorithm itself is described in chapter 4. It starts with the reconstruction of regions of activity from the hits in the trigger chambers. Within these so-called roads, the pattern recognition in the precision chambers is performed, followed by a global matching of the individual track segments. It is then concluded by a global fit through all precision and trigger hits.

- **Implementation and testing.** The first test of the algorithm, viz. the evaluation of its segment-reconstruction performance in a single precision chamber is described in chapter 5. Various levels of detector inefficiency and background are used to analyse the robustness of the algorithm.

This is followed in chapter 6 by a study of a complete tower of the muon spectrometer in the form of the DATCHA cosmic ray test setup. DATCHA is primarily a testbed for the muon alignment system, but is also well suited for testing the performance of the particle reconstruction.

And ultimately the performance of the whole spectrometer for single-muon, dimuon and 4-muon event topologies is explored in chapter 7.

In the last chapter of this thesis we return to the Higgs; to study its decay into a final state of four leptons and to ascertain its discovery potential of the ATLAS detector. In contrast to all other chapters, these final results are not obtained using my own software, but with the standard ATLAS production tools instead.

Keep it simple; as simple as possible, but no simpler.

Albert Einstein

1.1 Elementary Particles and their Interactions

Elementary-particle physics is the study of the fundamental building blocks of nature and the interactions between them. As far as we know, there exist 12 different building blocks, viz. six quarks and the same number of leptons, all of which are spin- $\frac{1}{2}$ fermions. They are distributed over three families:

| Family | 1 | 2 | 3 |
|----------------|---------------------|-----------------------------|------------------------------|
| Quarks | up down | charm strange | top bottom |
| Leptons | electron ν_e | muon (μ) ν_μ | tau (τ) ν_τ |

Table 1.1 The three quark and lepton families.

All of these particles are subject to one or more of the four fundamental interactions or forces:

- The **gravitational interaction**, which attracts all particles that have a mass. In the energy domain of elementary-particle physics this force is so weak that it can safely be neglected;
- The **electromagnetic interaction**, which is felt by all charged particles. It is mediated by the exchange of massless photons and is described by the theory of Quantum Electrodynamics (QED);
- The **weak interaction**, which influences all quarks and leptons and which is carried by three massive vector bosons, called the W^\pm and the Z^0 .

The electromagnetic and weak interactions are combined into a single theory, that of the electroweak interactions [2-4];

- The **strong interaction**, which is present between particles carrying a color charge (the quarks) and is mediated by eight massless gluons. A description of it is provided by Quantum Chromodynamics (QCD).

The last three of these interactions have been incorporated into a single theory called the Standard Model. It is characterized by the gauge group

$$SU(3)_C \times SU(2)_L \times U(1)_Y \quad (1.1)$$

in which the first term corresponds to the strong interaction, while the remaining two terms form the electroweak theory.

1.2 The Higgs Mechanism

As presented above, the electroweak theory defines four force carriers, viz. the photon, the W^\pm and the Z^0 . Except for the photon these vector bosons are massive. This poses a problem to the theory for it is impossible to generate these masses by adding explicit mass terms to the Lagrangian as they would break the gauge invariance. So instead, the vector bosons must acquire their mass through the process known as spontaneous symmetry breaking.

The symmetry of a theory is spontaneously broken when its vacuum is not invariant under the total symmetry group of the Lagrangian, but only under one of its subgroups. Or in other words, the vacuum has spontaneously chosen an arbitrary direction in the space of symmetry transformations. As a classical example, consider the theory of ferromagnetism near the Curie temperature. Above it, all the dipoles in the ferromagnet are randomly oriented and the ground state of the system is rotationally invariant. Below the Curie temperature spontaneous magnetization aligns the dipoles in some arbitrary direction, thereby breaking the rotational symmetry.

The spontaneous breaking of a continuous symmetry like the one of the electroweak theory leads to the creation of zero-mass particles called Goldstone bosons [5]. However, because no known particles can be attributed to these bosons, they somehow have to be erased out of existence. The Higgs mechanism [6, 7] does just that by using them to generate masses for the vector bosons. According to the model, a Goldstone boson becomes the extra degree of freedom that a massive vector boson has over a massless one. Or in other words, a Goldstone boson is “eaten” by a massless vector boson to give it its mass. Mass is then nothing more than the result of an interaction with an omnipresent background Higgs field.

In the case of the Standard Model, the Higgs field has four degrees of freedom. When it is given an appropriate vacuum expectation value it breaks the electroweak symmetry group $SU(2) \times U(1)$ down to the $U(1)$ of electromagnetism. In the process, the vector bosons swallow three of the four degrees of freedom of the Higgs field and become massive, while the remaining fourth degree of freedom manifests itself as the physical Higgs boson.

1.3 Higgs Creation

Like any particle, the Higgs can be created in a collision of other particles. In the case of proton-proton collisions, the dominant channel for the production of a Higgs boson is gluon-gluon fusion (see figure 1.1). The cross sections of the other channels are in general 1 to 2 orders

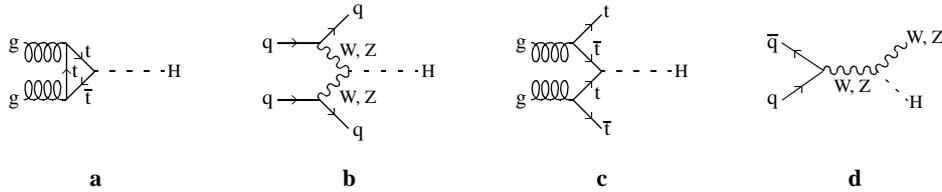


Figure 1.1 The four main Higgs production channels: gluon-gluon fusion (a), WW and ZZ fusion (b), tt fusion (c), and W and Z bremsstrahlung (d).

of magnitude smaller. As can be seen from figure 1.2 only above a Higgs mass of 900 GeV does the weak boson fusion process give a contribution comparable to that of gluon-gluon fusion. However, the advantage of the other three channels is that the Higgs is accompanied by two quarks or an intermediate vector boson. These give rise to specific signatures in the detector (two jets, a lepton pair or an isolated lepton) that can be used to suppress most of the background.

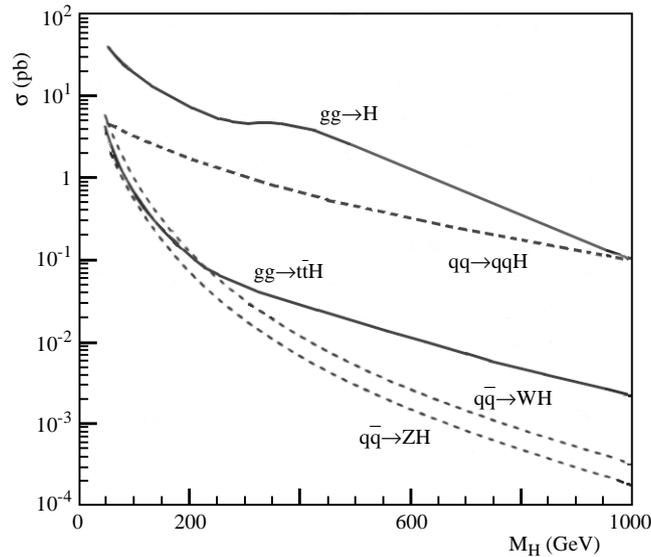


Figure 1.2 Production cross sections for the main Higgs creation channels in a proton-proton collider with a centre-of-mass energy of 14 TeV [8].

1.4 Higgs Decay Channels

Because of its nature, the coupling strength of the Higgs to other particles depends strongly on their mass. As a result, it decays preferentially into the heaviest particle available. But as its mass is not predicted by the Standard Model, which process that is, is not known.

Some limits on the Higgs mass can however be set. Because it has not yet been observed by any existing experiment, a lower bound of ~ 109 GeV can be assumed [9]. Furthermore, indirect predictions from the precision fits using the full set of electroweak data put the upper limit at a mass of 215 GeV (95% confidence level), with the preferred value at

$$m_H = 77^{+69}_{-39} \text{ GeV} \quad (1.2)$$

Based on this value, the possible decay channels for the Higgs are [10]:

$80 < m_H < 150$ GeV

If the Higgs mass is less than twice the W-boson mass, the decay mode $H \rightarrow b\bar{b}$ is the dominant channel with a branching ratio of 90% (see figure 1.3). Unfortunately however, in the case of direct Higgs production, this decay cannot be identified over the huge QCD two-jet background. Therefore, the associate-production channels with a W or Z boson, or a $t\bar{t}$ pair are the most promising processes: The isolated high- p_T lepton(s) or additional jets that they create, reduce the background to such a level that identification is possible.

The only other detectable decay channel of a light Higgs is the higher-order process

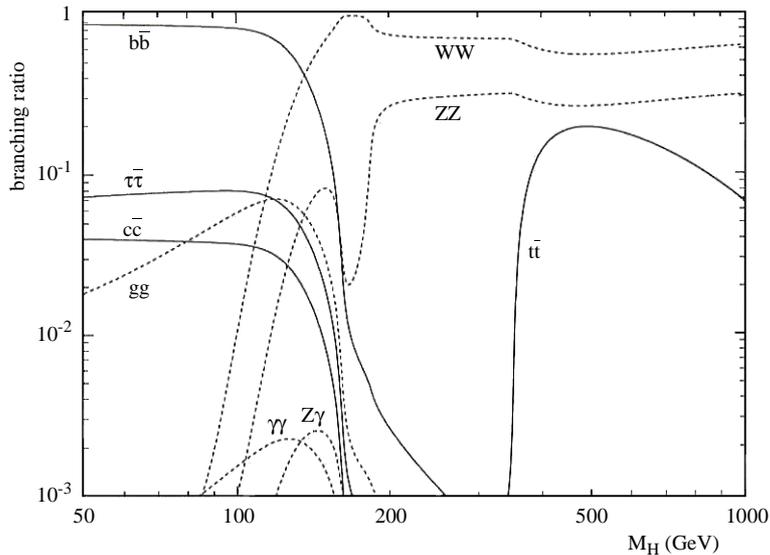


Figure 1.3 Higgs branching ratios.

$H \rightarrow \gamma\gamma$. It is only observable over a limited mass range ($100 < m_H < 150$ GeV), where the production cross section and the decay branching ratio are both relatively large.

$130 < m_H < 160$ GeV

In this mass range the decay $H \rightarrow ZZ^* \rightarrow 4$ leptons, with one of the Z bosons off its mass shell, is the best-observable channel. The branching ratio is larger than that of the two-photon channel and rises even further with increasing Higgs mass up to a value of around 150 GeV. Then a pronounced dip appears because of the WW-creation threshold.

$150 < m_H < 190$ GeV

When the Higgs is approximately twice as heavy as the W boson, the leptonic branching ratio of $H \rightarrow WW^{(*)} \rightarrow l\nu l\nu$ is about a hundred times larger than that of the ZZ^* mode. Its disadvantage however is that it is impossible to reconstruct a mass peak, because of the missing energy from the neutrinos. Instead, an excess of events must be used to identify the presence of a Higgs signal and to extract information about its mass.

$2m_Z < m_H < 700$ GeV

In this very large mass range, the Higgs can be easily identified from the decay $H \rightarrow ZZ \rightarrow 4l$, which is referred to as the “gold-plated” channel. Its signal is much higher than the expected background, which is dominated by the continuum production of Z boson pairs.

Because the main topic of this thesis is muon reconstruction, the decay channel of the Higgs into a final state consisting of four leptons is the most relevant one. We will come back to it in chapter 8.

┌

┐

└

┘

CHAPTER 2 | The ATLAS Experiment

Man shall never reach the moon, for such a quantity of gunpowder would be needed as to gravely injure the crew.

Children's Encyclopaedia, 1926

2.1 The LHC

Of the colliders currently in operation at CERN¹, the electron-positron collider LEP achieves the highest centre-of-mass energy, surpassing the 200 GeV. During its lifetime it has undergone many upgrades, but increasing its energy into the TeV range will not be possible. For a circular collider like LEP, the energy loss due to bremsstrahlung would become too high: a 500 GeV electron would come virtually to a halt before it would have been able to make one full turn through the accelerator. The only way to accelerate electrons to these kinds of energies is to use a linear collider.

However, because of the infrastructure available at CERN (see figure 2.1), a circular collider is a much more viable option. The only solution then is to use heavier particles. Because the amount of synchrotron radiation is inversely proportional to a particle's mass to the fourth power, a proton would generate only a fraction of the radiation lost by an electron. CERN has therefore decided to build the Large Hadron Collider or LHC, a proton-proton collider that will replace LEP and become operational in 2005 [11].

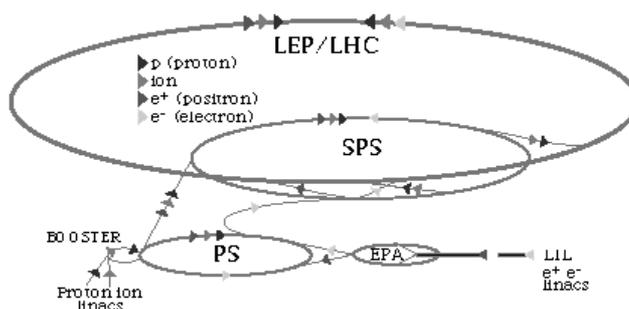


Figure 2.1 CERN accelerators.

1. The European Laboratory for Particle Physics, located near Geneva, Switzerland.

The disadvantage of using protons is that they are not elementary particles; they consist of three valence quarks and a sea of quark and anti-quark pairs, all immersed in a plethora of gluons. All of these constituents carry part of the proton's energy, and it is these particles that actually collide with each other. As a result, to create collisions with energies around 1 TeV, the protons have to be accelerated to much higher energies. This has led CERN to set the centre-of-mass energy of the LHC at 14 TeV.

Another unfortunate side effect of all these particles within the proton is that most of the collisions will have a soft hadronic nature, and will do nothing more than to obscure the interesting events. The interesting cross sections are consequently small, and in order to be able to maintain an effective physics programme, the interaction rate must be immense. The LHC will concentrate the protons into bunches, each containing around 100 billion particles. When the two bundles are then focused on each other, an average of 23 collisions will take place at each bunch crossing. The majority of these are of a soft hadronic nature and are called minimum bias events or "pile-up". Together with a 25 ns bunch spacing, this means that the LHC will operate at a luminosity of $10^{34} \text{ cm}^{-2} \text{ s}^{-1}$.

To study the collisions at the interaction points of the LHC, two general-purpose particle detectors are being built: ATLAS (A Toroidal LHC Apparatus [12, 13]) and CMS (Compact Muon Solenoid [14]).

2.2 The ATLAS Detector

ATLAS is a general-purpose detector, not only capable of finding the Higgs in the mass range from 80 GeV to 1 TeV, but also versatile enough to study B and top physics, supersymmetry, heavy vector bosons, and many other topics. As any typical colliding-beam detector it consists of an inner and outer tracker separated by electromagnetic and hadronic calorimeters (see figure 2.2).

Inner Detector

The inner detector utilizes three technologically different subdetectors to measure charged particles and to identify (secondary) vertices. Closest to the interaction point are layers of high-resolution pixel detectors, which provide 3-dimensional space points essential for the vertex reconstruction. On the outside of it lie the layers of the silicon strip detector (SCT), followed by the TRT, a straw tube transition radiation tracker. To reconstruct the particles' momenta, the whole inner tracker is surrounded by a superconducting solenoid generating a field with an average value of 2 Tesla.

Calorimetry

The ATLAS calorimetry uses different techniques in various regions of the detector as best suited to the specific requirements and the varying radiation environment. The electromagnetic calorimeter is a liquid-argon (LAr) detector with an accordion geometry. In the range $|\eta| < 1.8$

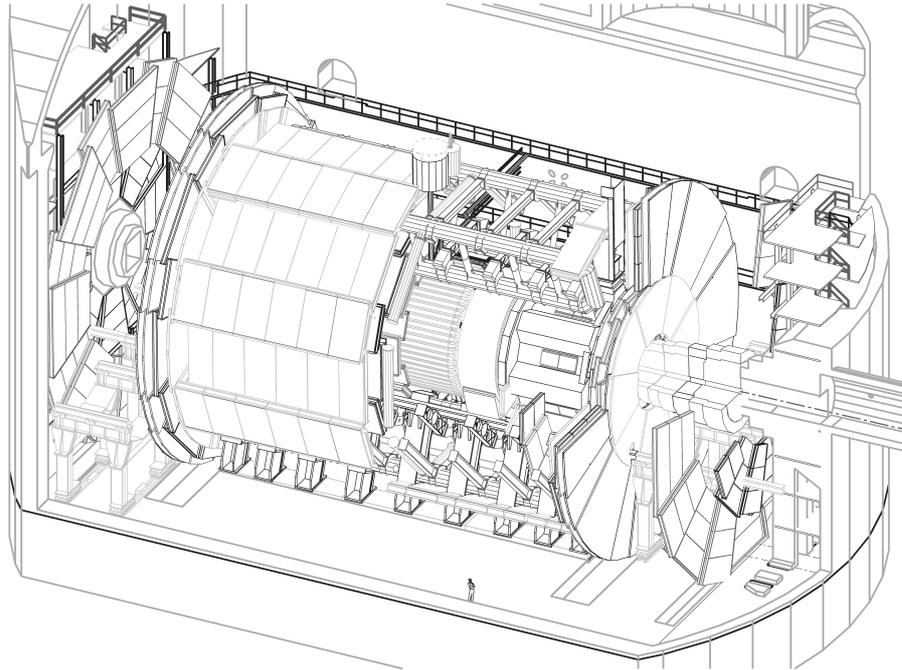


Figure 2.2 Three-dimensional view of the ATLAS detector with parts of the muon spectrometer removed to show the inner structure of the detector.

it is preceded by a presampler, whose finer granularity can be used to discriminate between photons and pions.

The hadronic calorimeters in the barrel are made of iron interspersed with scintillating tiles directed towards the interaction point. In the endcaps the tiles are not suited because of the high levels of radiation present there, and are replaced by liquid argon calorimeters. They extend the coverage up to a pseudorapidity of $|\eta| = 4.9$, which is needed to correctly identify events with missing energy.

Outer Tracker

The calorimeter is surrounded by the muon spectrometer, a tracker that is based on large superconducting air-core toroids (see the next section). It defines the overall dimensions of the ATLAS detector: The outer chambers of the barrel are at a radius of about 11 m, the half-length of the barrel toroid coils is 12.5 m, and the third and outer layer of the forward muon chambers, mounted on the cavern wall, is located some 23 meters from the interaction point.

2.3 The Muon Spectrometer

The muon spectrometer consists of widely interspersed stations of chambers, which are positioned in such a way that particles coming from the interaction point always traverse at least three of them. In the barrel region this is achieved by arranging them in cylindrical layers around the beam axis, while in each endcap wheels with detectors, concentric around the beam axis are used (see figures 2.3 and 2.4). In both these regions the detector is divided into 16 towers

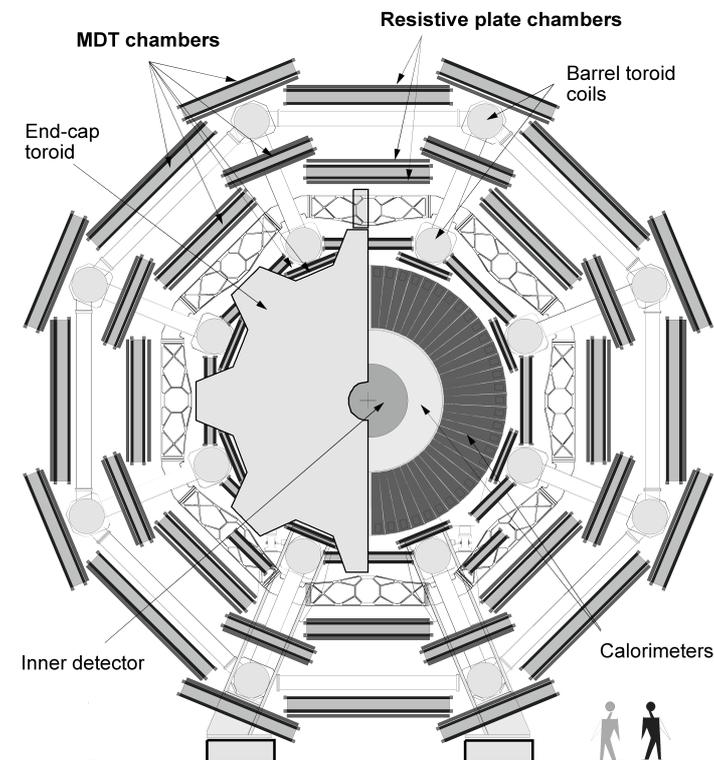


Figure 2.3 Transverse view of the barrel of the muon spectrometer [15].

alternately consisting of large and small stations.

Each station contains one chamber for the precision measurement of the particle's position. In most cases these are MDTs (Monitored Drift Tube chambers), and only in the inner forward region where the counting rate is extremely high are they replaced by Cathode Strip Chambers (CSCs). Some stations also contain separate trigger chambers: Resistive Plate Chambers (RPCs) in the barrel and Thin Gap Chambers (TGCs) in the endcaps. Together they contain a total number of readout channels exceeding the 1.2 million.

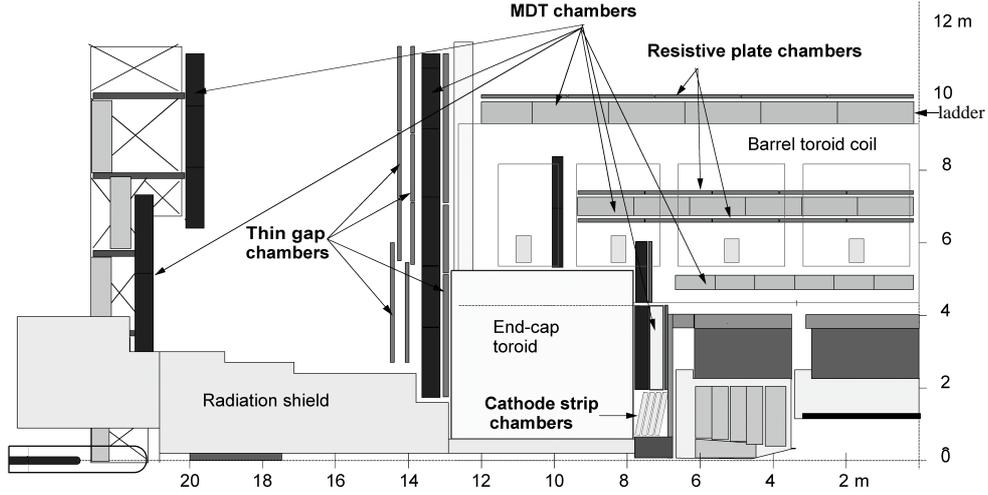


Figure 2.4 Side view (rz-projection) of one quadrant of the muon spectrometer.

Magnetic Field

The measurement of the particles' momenta is made possible by the presence of a toroidal magnetic field. This field is generated by 3 superconducting air-core magnets whose coils follow an eight-fold symmetry, with the endcap toroids rotated with respect to the barrel ones by 22.5° (see figure 2.3).

The open structure of the magnets minimizes the effects of multiple scattering and energy loss. In the barrel it allows for the reconstruction of a muon's momentum from a measurement of the sagitta in the three muon stations. In the endcaps the positions of the magnet cryostats do not allow for this arrangement. Instead, the muon momenta are obtained from a point-angle measurement with one point in front of and two points behind the magnetic field region.

The open air-core toroid generates a relatively modest magnetic field with an average value of 0.5 Tesla. A field that is also very inhomogeneous (see e.g. figure 2.5) so that particle trajectories can be very irregular, especially at low transverse momenta.

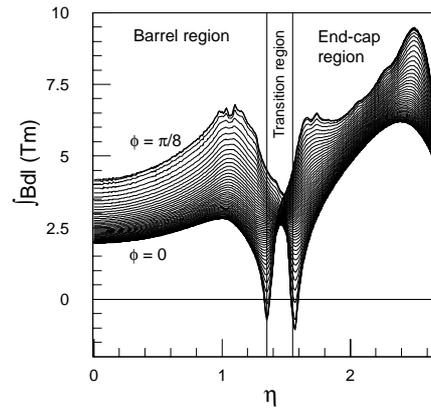


Figure 2.5 Field integral versus η for infinite momentum muons. Each curve corresponds to a fixed azimuthal angle [15].

Precision Chambers

Monitored drift tubes are used for the precision measurement over most of the area of the detector. Almost all MDT chambers consist of two multilayers made up of 3 or 4 monolayers of drift tubes. The barrel chambers are rectangular while the endcap chambers are of trapezoidal shape, but otherwise the design is similar.

The aluminium drift tubes have a diameter of 30 mm and are operated with a gas mixture Ar(91%)-N₂(4%)-CH₄(5%) at 3 bar absolute pressure. With the selected gas, the maximum drift time is around 500 ns and the average single tube resolution is 80 micron, except very close to the wire where it rises sharply.

Only in the inner station of the endcap region are the MDTs replaced by cathode strip chambers to provide a finer granularity, which is required to cope with the demanding rate and background conditions present there. CSCs are multiwire proportional chambers with cathode strip readout. They have a symmetric cell in which the anode-cathode distance is equal to the anode wire spacing, viz. 2.54 mm. This is considerably less than the MDT tube radius, thereby lowering the occupancy per wire as well as the electron drift times to a maximum of 30 ns. The precision coordinate is obtained from a measurement of the charge induced on the segmented cathode by the avalanche formed on the anode wire. The resulting resolution is around 60 μm .

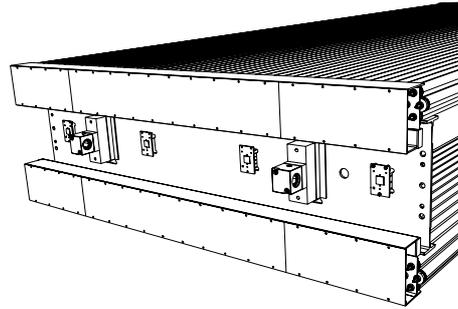


Figure 2.6 The readout side of a MDT chamber.

Trigger

The muon trigger chambers cover the pseudorapidity range of $|\eta| \leq 2.5$ and serve a threefold purpose:

- First and foremost as a trigger system with a well-defined p_T cut-off. This requires a granularity of the order of 1 cm, given the magnetic field generated by the toroids.
- For bunch crossing identification, requiring a time resolution better than the LHC bunch spacing of 25 ns.
- For the measurement of the second coordinate, i.e. the coordinate in the direction orthogonal to the one measured in the precision chambers, with a typical resolution of 5-10 mm.

In the barrel, the trigger chambers in the form of the RPCs are arranged in three layers. They are located on both sides of the middle MDT station and either directly above or below the outer MDT station (see figure 2.3). The RPCs are gaseous detectors providing a typical spatial resolution of 1 cm and a time resolution of 1 ns. The basic unit is a narrow gas gap formed by two parallel resistive plates with readout strips on both sides of the gap, one measuring the η - and the other the ϕ -coordinate. Each chamber then consists of two such units.

The TGCs form the trigger system in the endcaps. Three layers complement the middle MDT station while a fourth layer is present near the inner station, but that one is not used for the generation of the trigger signals. The TGCs are multiwire proportional chambers with the anode wires arranged parallel to the MDT wires. The two outermost chambers are doublets: They consist of two gas gaps, each equipped with readout strips that are orthogonal to the wires and that measure the second coordinate. The chambers on the inside of the middle MDT layer (TGC1 in figure 2.7) are triplets with three wire planes but only two strip planes. Finally, the TGC0 chambers, which are the ones closest to the interaction point, serve only to measure the second coordinate. They consist of two gas gaps without any strips.

The trigger system has been designed such that it can supply both low (6 GeV) and high (20 GeV) momentum trigger signals. The 6 GeV trigger in the barrel is based on the RPC1 and RPC2 layers. In both the η - and ϕ -projection, a coincidence in 3 out of the 4 strip planes is required. In the endcaps, the same trigger is realised by a 3 out of 4 coincidence in the TGC2 and TGC3 chambers, i.e. the two outermost layers.

Both triggers can be extended to become 20 GeV triggers by respectively requiring an additional hit in each projection in the outer RPC layer, or a 2/3 coincidence in the bending plane of the triplet of wire layers of the TGC1 chambers plus a 1 out of 2 coincidence in its azimuthal strip planes.

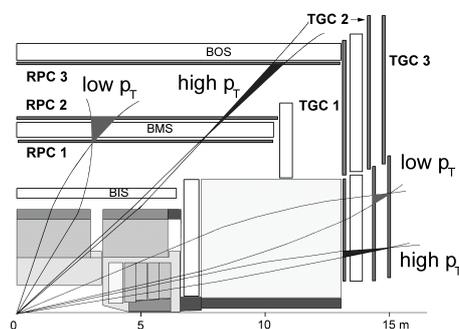


Figure 2.7 Level-1 muon trigger scheme.

2.4 ATLAS Computing

The ATLAS detector as described above is a complicated piece of equipment and as such requires a substantial effort to design, build and test it. This not only holds true for its hardware, but for its software as well. Most of the simulation, reconstruction and analysis software that is currently available is written in Fortran. However, due to the size and complexity of the code, this has led to severe maintenance problems. To solve these problems, ATLAS has after extensive studies [16-18] decided to adopt the object-oriented (OO) methodology, together with C++ as the implementation language [19-23].

2.4.1 Object Orientation

Object-oriented programming is about capturing the behaviour of the real world in a way that hides the implementation details. In other words, it allows the programmer to think in terms of the problem domain, as opposed to the world of the computer (language). It is also a data-centered view of programming in which data and behaviour are strongly linked: They are

combined into a single entity called a class. Instances of a class are called objects and each object has its own set of data, giving it a unique identity.

The three core features of OO are abstraction, encapsulation and polymorphism.

Encapsulation

Encapsulation, which is also referred to as data-hiding, is about hiding an object's implementation, i.e. its data, from its clients. Instead, they only see the object's interface, i.e. the behaviour it presents to the world. The advantage of this clear separation is that the object is free to implement its interface in any way it sees fit.

This feature can also be extended to entire software packages by limiting their entry points to a number of interface classes. Encapsulation can therefore lead to a significant reduction in the complexity of the software by increasing code modularity. It also enhances its flexibility and robustness, and promotes code reuse.

Polymorphism & Inheritance

Polymorphism allows different kinds of objects that share some common behaviour, i.e. have a common interface, to be used interchangeably. Or in other words, it is the ability of an operation to behave differently depending on the type of the object on which it is invoked. For example, a box and a tube are both geometrical objects that can be drawn on a screen, but the implementation of their draw method is completely different.

The way in which most object-oriented languages implement polymorphism is through inheritance. Inheritance can open the way to code reuse, but only when the class hierarchies are explicitly designed with that in mind.

Abstraction

Abstraction lies at the heart of OO: During the analysis phase, real world concepts are abstracted into classes. Later on, during the design, similarities among objects are expressed in terms of interfaces and base classes, using respectively polymorphism and inheritance.

When taking all this into account, an object-oriented application is simply a collection of collaborating objects: They interact and communicate with each other by sending and receiving messages. Whereas procedural designs rely heavily on a (few) main function(s) to manage the application, OO designs grant more equality of control to the objects within the application. The goal in OO design is to achieve a consistent set of objects whose behavioural characteristics (their interfaces) form the collaborations needed to fulfil the requirements.

2.4.2 Domain Decomposition

It was realised from the start that applying the object-oriented paradigm alone would not be sufficient to create high quality and maintainable software. A software process [24] was therefore developed, providing guidelines for the analysis, design and coding phases, and defining a review mechanism to check the quality of their outputs.

Part of the process has been the division of the software and its development into domains, the ones most important to the reconstruction being (see figure 2.8) [25]:

- The **control** domain provides the steering mechanisms for the application, i.e. the way domains communicate with each other, the method in which parameters from e.g. the user interface are made available, etc.;
- The **(graphical) user interface** provides access to the program and handles the user's input. The design must be able to deal with the simultaneous existence of multiple interfaces;
- The **event display** represents visually the objects that exist within the ATLAS software. To meet the needs of the community, many different displays are envisaged;

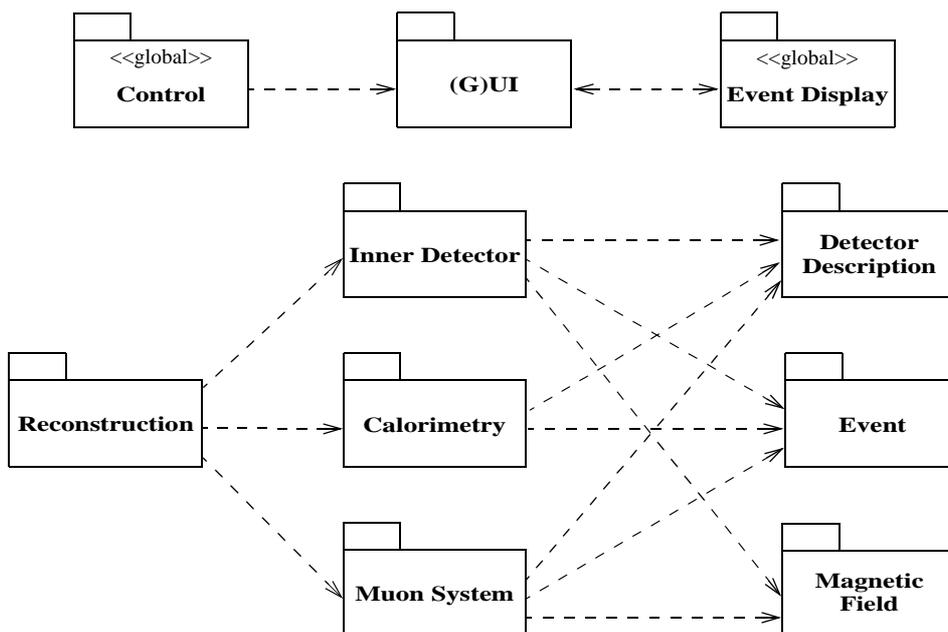


Figure 2.8 Domain decomposition of the ATLAS reconstruction software. The notation is explained in appendix A.1.1.

- The **reconstruction** domain coordinates the activities of the detector domains, and performs the combined reconstruction and possibly the particle identification;
- The **inner detector**, **calorimetry** and **muon system** domains are responsible for the stand-alone reconstruction in the respective subdetectors;
- The **detector description** stores the description of the ATLAS detector in a database and makes it available through various logical views as needed by e.g. the simulation, the reconstruction and the event display;
- An **event** is the container for all data associated with a physics event, i.e. the raw or simulated data, the reconstruction results and the analysis objects. The event domain is responsible for the storing of these events, and for providing fast, flexible and easy access to them;
- The **magnetic field** domain provides access to the magnetic field, and performs the propagation of particles through it.

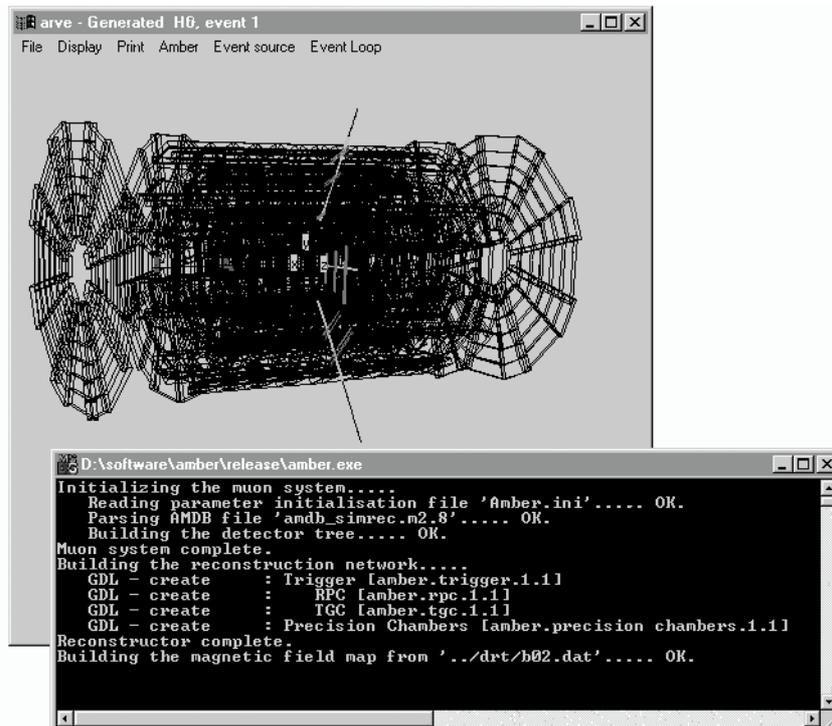


Figure 2.9 Arve display and console window.

2.4.3 Arve

As the baseline for further development of a full OO reconstruction program, Arve (the ATLAS reconstruction and visualization environment) has been adopted [26]. Arve is an object-oriented framework for reconstruction and physics analysis intended to facilitate fast and easy development. It defines the classes for building a detector hierarchy, and for simulating the traversal of particles through it. In addition, it defines a control and GUI structure with integrated graphics that are tailored towards the task of software creation.

This concludes the analysis of the problem domain. The next chapter continues with the design of the software.

┌

┐

└

┘

CHAPTER 3 | Software Design

Minds are like parachutes. They only function when they are open.

Sir James Dewar

3.1 Global Architecture

The goal of the software described in this thesis is to perform the stand-alone reconstruction of events in the ATLAS muon spectrometer. Hence, following the domain decomposition as presented in the previous chapter, it belongs to the muon system domain. However, from a global viewpoint it doesn't look any different than any other reconstruction program, independent of the language in which it is written. A detector description, an event structure,

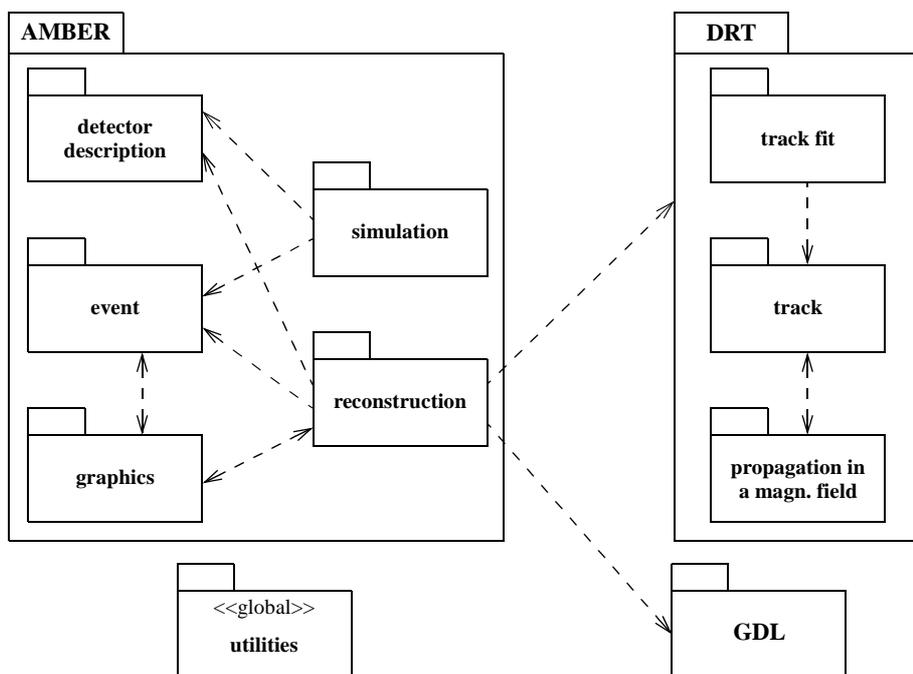


Figure 3.1 Global architecture of the muon reconstruction software (see also appendix B).

the implementation of a reconstruction algorithm, etc. are all easily visible. The difference with most other programs is that here they are strictly separated, and only see each other through a handful of interface classes. Furthermore, in adherence to good design practices [27], anything that is not specific to the ATLAS muon reconstruction has been split off.

All these observations are especially true of the way the reconstruction algorithm is implemented. To explore the full potential of OO and C++, it has been designed using everything they have to offer. Along the way, the main design goals have been flexibility, extensibility, reusability and robustness in the face of change. Or in other words, the design tries to adhere to the Open Closed Principle, which states that the software must be extensible without requiring change to the existing code [28].

In fact, the ATLAS muon reconstruction algorithm is nothing more than a blueprint, defining how to use the building blocks supplied by two independent, general-purpose packages, viz. the Detector Reconstruction Toolkit or DRT and the Generic Dataview Library or GDL. The DRT defines a diverse set of general reconstruction classes such as error points and cones, tracks and magnetic field implementations. In addition, it performs a number of related tasks such as track fitting and the propagation of tracks through a magnetic field.

The second package, the GDL, is a novel library that incorporates the dataflow principle into an object-oriented design, and provides a framework in which data-driven algorithms can be implemented in a straightforward and intuitive way. It is the core of the reconstruction; controlling it and defining its logic.

A third independent package holds the utilities [29, 30]. It is a library of general-purpose classes that provide support for such diverse things as commands and callbacks, smart pointers, named parameters and a basic Component Object Model (COM). The latter is explained in section 3.3.1, while some of the other classes are mentioned in the parts of this chapter to which they bare relevance.

Many of the paragraphs that follow go into considerable detail. For those of you not interested in this, the beginning of each section gives an overview of the functionality offered by the corresponding package. The subsequent subsections can then be skipped without losing the ability to comprehend the rest of this thesis. Also, some of the more important terms are explained in the glossary (see appendix C).

3.2 AMBER

The ATLAS Muon Barrel and Endcaps Reconstruction program or AMBER performs the stand-alone reconstruction of events in the muon spectrometer. But from the start it has been designed with flexibility in mind. This also makes it a framework, aimed at facilitating the development of reconstruction algorithms and the building of complete programs for the ATLAS muon spectrometer and beyond. Special care has been taken to shield it from the ATLAS specific definitions for the detector description and the event structure, and to decouple the different subpackages from each other as much as possible.

In this section an overview of these subpackages is given, loosely following the steps taken when processing an event, up to the moment when the actual reconstruction starts. That

will be the topic of the next chapter. It is impossible to go into all the details here, and in many cases therefore only the top-level classes are shown.

3.2.1 Integration into Arve

Following the ATLAS strategy at the time, AMBER is integrated into the Arve framework. It uses its control system to steer the processing of events, its geometry and detector description classes to represent the muon spectrometer and to simulate its behaviour, and its graphics and console windows to output the results. In AMBER's main function, the first two lines exist to create and initialize Arve, as well as its core package called Gismo (see listing 3.1), while the other commands deal with the initialization of the main AMBER classes, beginning with System.

```
Arve app(world_size);
new Gismo(world_size, 0, &Arve::instance()->display());

// System setup
amber::System().initialize();
amber::System().reconstructor(new amber::Reconstructor());

// Event sources
amber::EventSimulator* simulator = new amber::EventSimulator(
    new amber::SingleParticleGenerator("mu+", Point(0, 0, 0)));
amber::EventGenerator::instance()->add(simulator);
amber::EventGenerator::instance()->add(new amber::G3EventLoader());

app.run();
```

Listing 3.1 AMBER initialization (main function).

Within Arve, each subsystem is represented by a `Module` class whose function it is to construct the system and to control all outside access to it. In the case of the muon system domain, this task is performed by a combination of the `System` class and its nested class `Module` (see figure 3.2). The former is a monostate class, which means that all objects share the same state¹. As a consequence, `System`'s constructor can not be used to initialize its state, and the `initialize` method has to be called instead (line 4 in listing 3.1). Its foremost tasks are to create a new instance of its nested `Module` class, in order to incorporate AMBER into the Arve framework, to read one or more parameter initialization files, and to construct the detector description.

1. The monostate pattern [31] has been selected in favour of the singleton pattern (see section 3.2.2 for a description), because only the former can be used in conjunction with inheritance. This means that specialized system classes can inherit from `System`.

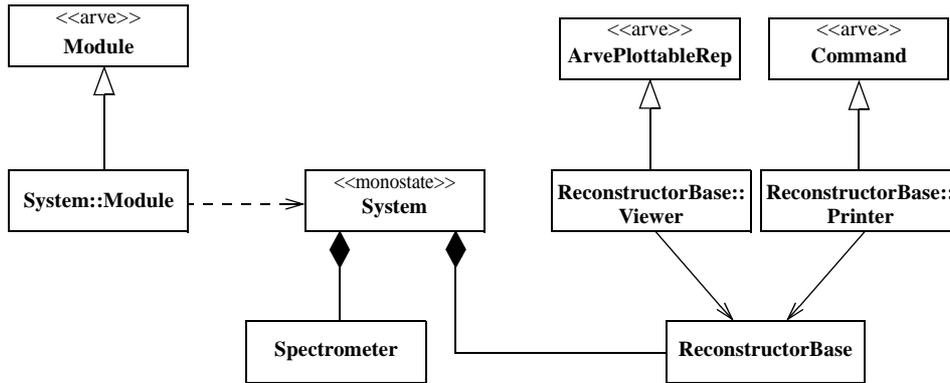


Figure 3.2 External interface of AMBER as seen from Arve. For an explanation of the syntax, see appendix A.1.2.

In addition to the detector tree, System also stores a link to the reconstructor. Any class inheriting from ReconstructorBase and implementing its clear and execute methods will do. This base class, through its nested Viewer and Printer classes, provides the link to Arve's GUI, ensuring that the results of the reconstruction are both displayed on the screen and printed on the console (see section 3.2.4).

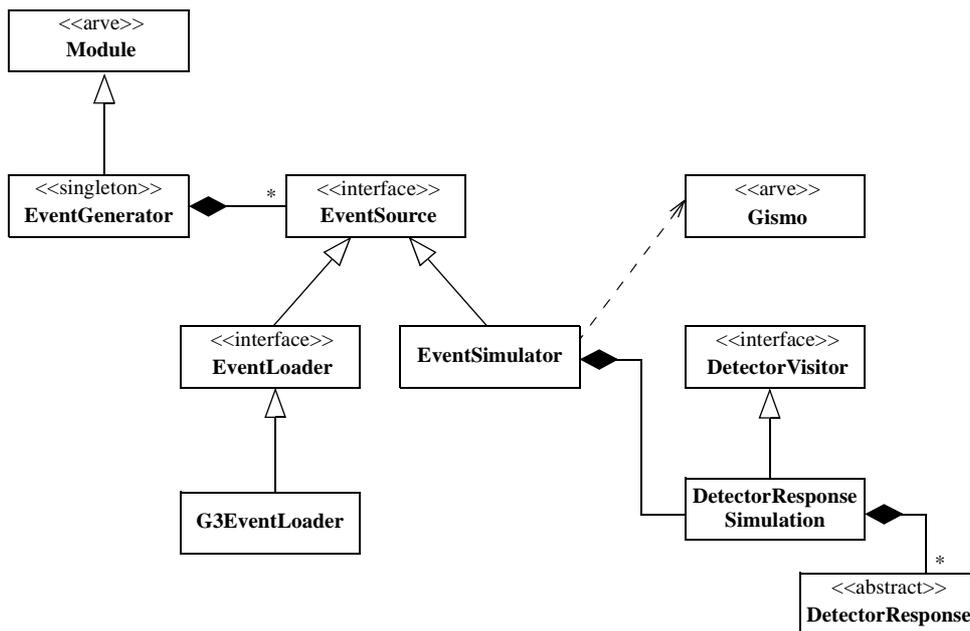


Figure 3.3 Event loading and simulation.

3.2.2 Event

After the creation of the System class, the various event sources are constructed (see figure 3.3). Like the System class, EventGenerator is a descendant of Arve's Module, and is therefore executed during each pass through Arve's event loop. The generator is implemented as a singleton [32], which limits the number of objects that can exist at run-time to exactly one. Its instance method grants access to that single object for, among others, the System class: EventGenerator is added to the list of modules that have to be executed first before System can run its reconstructor.

The event generator class stores an arbitrary number of EventSources, which are responsible for the actual generation of the events. Two such sources are implemented by AMBER, viz. the G3EventLoader, which reads in events generated by the ATLAS simulation software based on Geant 3, and EventSimulator, which uses Arve's internal simulation (the Gismo class).

The EventSimulator in turn contains a DetectorResponseSimulation object that visits the detector hierarchy and updates the results of the simulation. DetectorResponse descendants exist for simulating detector inefficiencies, adding noise and taking into account the finite resolution of the various detectors. The sequence of calls that are executed when the EventSimulator is called upon to generate an event is shown below.

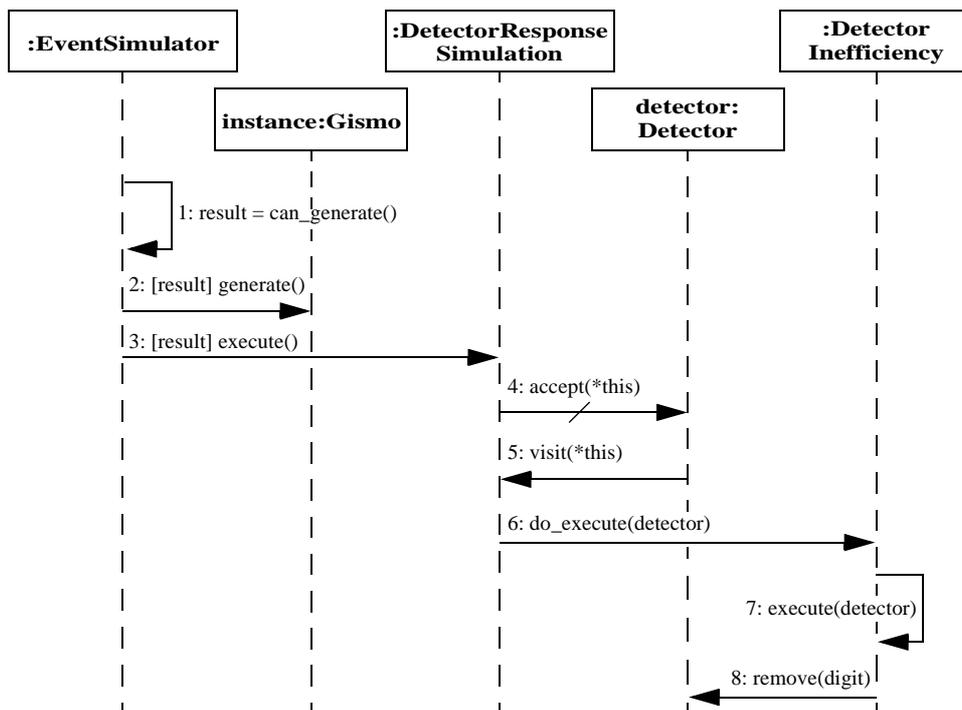


Figure 3.4 Sequence diagram showing some of the classes involved in simulating an event. The syntax is explained in appendix A.1.3.

1. First a test is performed to check whether the EventSimulator can generate a new event, i.e. to see whether it is enabled and the number of events has not reached the total amount requested by the user.
2. When a new event is to be created, the call is forwarded to the only instance of Arve's Gismo class. It generates an event and propagates it through the detector.
3. After the digits have been added to the detector, its response is modified to take into account effects like detector inefficiencies.
4. As DetectorResponseSimulation is a DetectorVisitor, it is passed on to the root of the detector tree (see the visitor pattern in [32]). During its traversal it passes through all detectors.
5. Upon receiving the visitor, each detector calls it back with itself as an argument.
6. From within the visit method, all DetectorResponse objects are executed with only the call to DetectorInefficiency shown here.
7. The do_execute method belongs to the DetectorResponse base class. It checks whether its specific type of response modification is enabled before passing execution on to its descendant.
8. Finally, DetectorInefficiency loops over all digits in the detector and for each one decides based on a random number whether to keep it or not. If a digit is to be deleted, the detector's remove method is called.

At the end of the event generation the digits are stored in the detector hierarchy, each one in the detector to which it belongs. During high-luminosity running, the number of these digits can become very large and they are therefore designed to be as lightweight as possible. In addition to the detector response information, e.g. the drift distance in the case of the MDT digits, they only store their element number (1...n) and a pointer to the detector to which they belong

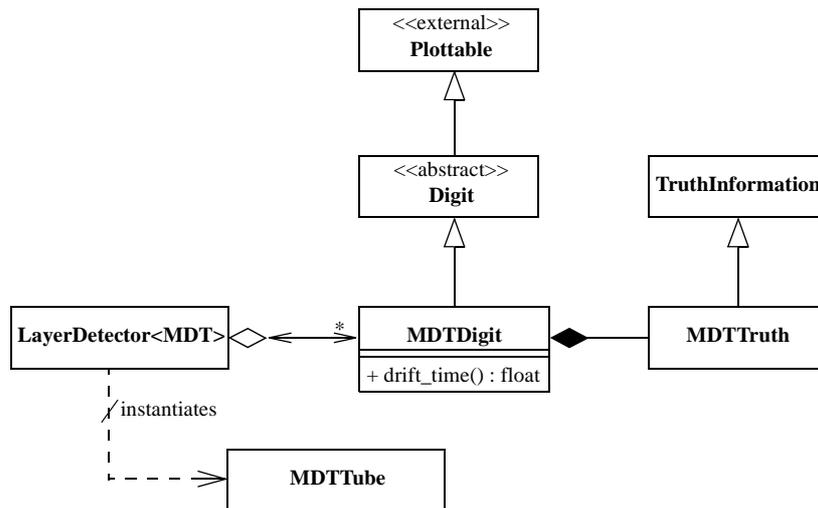


Figure 3.5 MDT digit structure.

(see figure 3.5). All remaining information like their position and dimension is calculated from their containment in that detector. These calculations are performed by the detector element classes, which for the MDTs is the `MDTTube`.

In the case of simulated data, the truth information is also stored in the `digit`². The `TruthInformation` base class stores general information such as a kine index, while `MDTTruth` keeps track of the MDT specific data like the real drift distance, the coordinate along the wire and the time-of-flight correction.

Also shown in figure 3.5 is the class `Plottable`. It is part of the ATLAS graphics design, and identifies `Digit` as being plottable on a graphics scene. This will be explained in section 3.2.4.

3.2.3 Detector Description

The term detector description is applicable to two different concepts. It can be used to describe the *data*, i.e. the actual geometrical parameters of the detector. And it can be related to the so-called *metadata*, i.e. a description of the logical structure of the detector. Within AMBER, the detector description is a combination of both: It is the structure within the program that is built based on the logical description of the detector, but also serves as the front-end to the geometrical properties, granting access to it and at the same time hiding its internal details. In addition, it is also the place in which the events are stored, providing the reconstruction with a uniform view of the data independent of their origin or type.

The full logical structure of the detector, which consists of the sensitive volumes, the dead material and their respective parents has to be present for Arve's internal simulation. Therefore, AMBER's detector description is built on top of the structure defined by Arve (see figure 3.6). Because the reconstruction attaches itself to the sensitive leaves of the detector (see section 3.2.3) and as a result only sees the parts of the whole detector hierarchy that it needs to see, there is no need to have two separate logical descriptions, one for the simulation and the other for the reconstruction.

Arve makes the distinction between media, volumes and detectors. Detectors are objects that know how to react to a particle crossing them, but they know nothing about their position or size. That is the task of the volume classes such as `Box` and `Tube`. Finally, the media classes complete this picture by combining the other two and by building a detector tree through the application of the composite pattern [32].

To interface to this design, AMBER defines two classes, viz. `Medium` and `Detector`³. In addition to their role of shielding the other AMBER classes from the details of Arve, they also store the official name of the medium, respectively the detector [15]. The generic `Medium` class is used for every part of the detector; the only exception to that rule being `Spectrometer`, which represents the root of the muon detector hierarchy. It is different, because it is responsible for

-
2. The pointer to the `MDTTruth` object can be removed with the help of a preprocessor directive, thereby eliminating any unnecessary overhead when running with real data.
 3. Because AMBER's source code resides within the `amber` namespace, there are no name clashes with the corresponding classes in Arve, or in any other part of the ATLAS software.

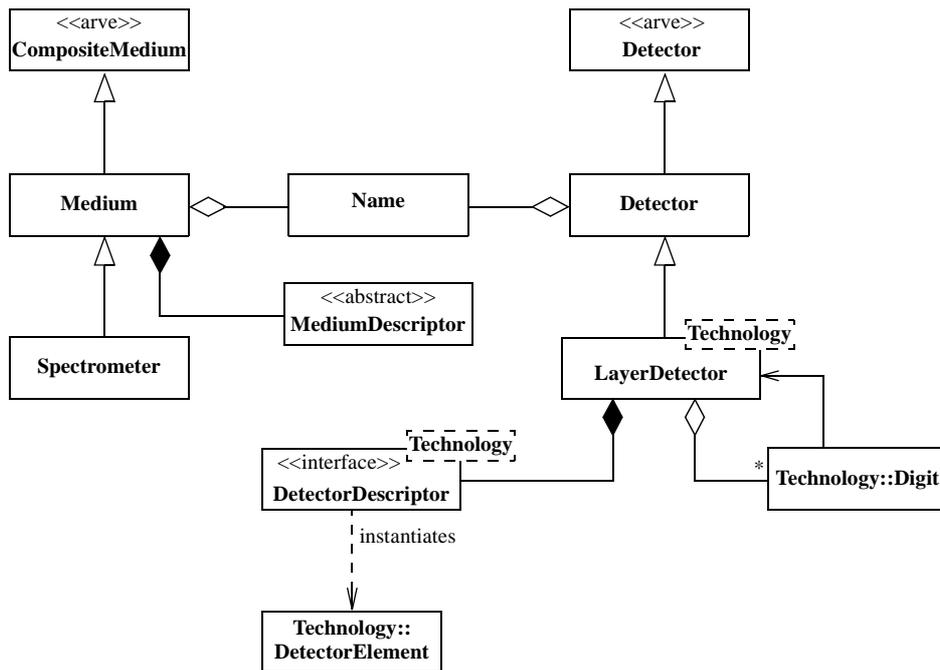


Figure 3.6 Interface view of AMBER's detector description domain.

the construction of the whole detector tree. As can be determined from figure 3.7, the construction sequence is as follows:

1. Based on the value of a named parameter⁴, Spectrometer creates a parser. The one shown here reads the ATLAS Muon Database or AMDB [33]. It then builds a GeometryDescription object containing the name, coordinate transformation, dimension and internal structure of each item in the detector hierarchy.
2. Spectrometer subsequently passes this description on to a DetectorBuilder, with itself as the parent to which the detector tree must be attached.

The division of this process into two separate steps (the parsing and the building) has been done to keep the impact of changes to the input format to a minimum. Furthermore, by defining a separate DetectorBuilder class instead of giving the medium and detector classes build methods, the design is more flexible because it minimizes the external dependencies of the detector description classes.

4. Named parameters are provided by the utilities package [30]. Based on the name of the parameter, they search for its value in a database.

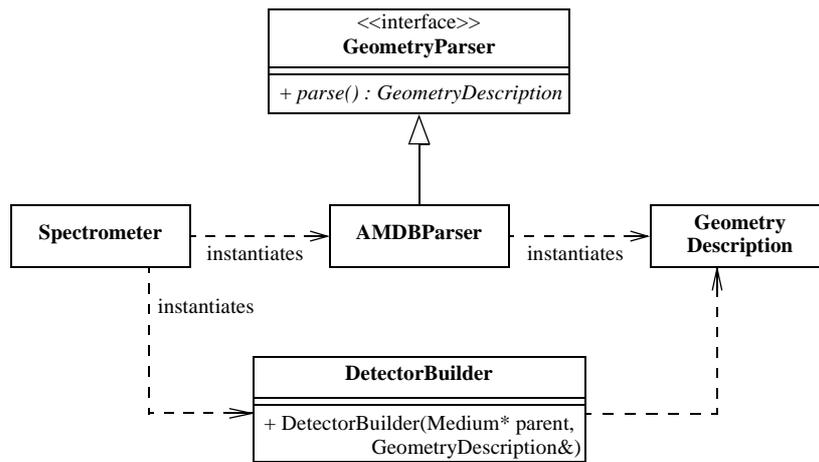


Figure 3.7 Classes involved in the creation of the detector description.

The final class in the detector description structure is `LayerDetector`. It represents one layer of detector elements, the type of which depends on the template parameter `Technology` (`MDT`, `RPC`, etc.). They store the digits and serve as the aforementioned entry points the reconstruction can hook onto.

This leaves the only classes in figure 3.6 that have not yet been mentioned, viz. the descriptors. These are interfaces, hiding the implementation details of the geometrical description of the detectors, and thereby making the detector description independent to changes made therein. In the case of `MediumDescriptor`, the functionality focuses on transformations between the local and global coordinate systems, while for the `DetectorDescriptor` class template the emphasis lies on the digitization process. The latter class also provides access to a descriptor for each detector element in the layer. The `MDTTube`, `RPCStrip`, `TGCWire`, `TGCStrip`, `CSCWire` and `CSCStrip` classes (their exact type is part of the `Technology` template argument) provide information about the position and dimension of the particular element they represent. They are generated on the fly, and are not stored in the detector.

3.2.4 Graphics

We end this tour of AMBER with a short look at how its objects are displayed. The official ATLAS graphics design contains four main interfaces (see figure 3.8). A `Plottable` is an object that can be displayed on an `AbstractScene`. For each combination of plottable and scene there exists a `PlottableRep` class that knows how to display the former on the latter. And lastly, the `PlottableModel` class is used to glue everything together by creating the correct representation for each plottable/scene pair.

To interface the scenes with the display capabilities of Arve, the `ArveGraphicsScene` and `ArveConsoleScene` classes have been written. The latter has two base classes separating it from

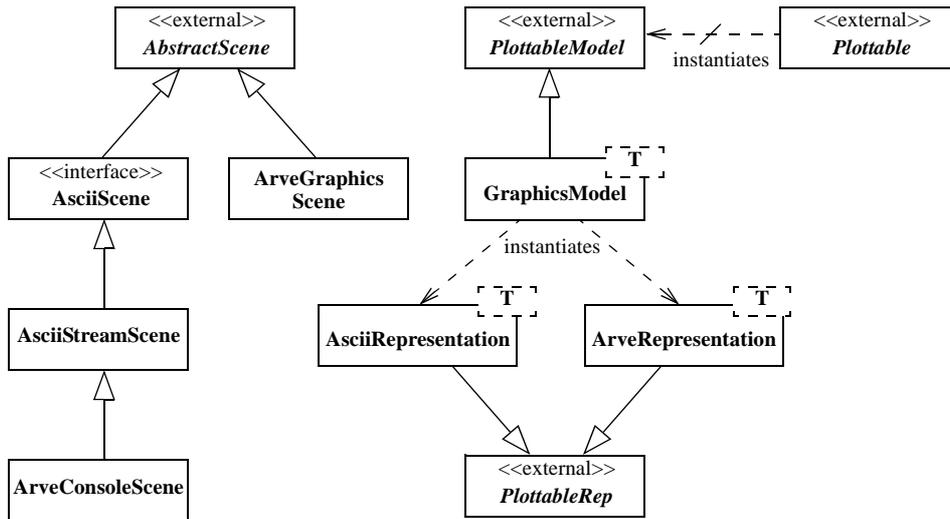


Figure 3.8 Implementation of the ATLAS graphics scheme within AMBER [34].

AbstractScene: `AsciiScene` is an interface defining the normal `std::ostream` operators [19], while `AsciiStreamScene` stores a pointer to an output stream to which it forwards all messages. `ArveConsoleScene` then only serves to hide from the user the details of obtaining the output stream corresponding to Arve's console.

To implement the `PlottableModel` interface, AMBER defines the `GraphicsModel` class template. It instantiates either an `AsciiRepresentation` or `ArveRepresentation` object, with both being specialized for every class that is plottable.

For example, in the detail of Arve's event display shown in figure 3.9 a part of a barrel station is depicted. In it, the Arve representations of the plottables `RPCDigit` and `MDTDigit` as well various reconstruction results are drawn. The dashed lines form the boundaries of the region of activity based on the RPC digits (see also section 4.1), and the line inside of it is the reconstructed track. The MDT hits, i.e. the digits that were found to be part of the track, are plottables as well, so that they are displayed in a different color than the unused digits.

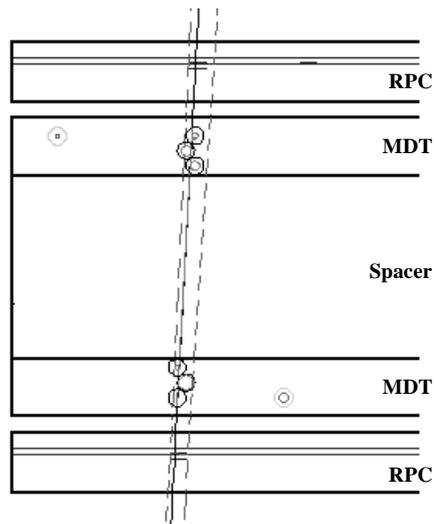


Figure 3.9 Detail of Arve's event display.

3.3 Detector Reconstruction Toolkit

The **Detector Reconstruction Toolkit** or DRT is, as the name suggests, a toolkit of classes that are useful in the reconstruction of physics events. They were thought general enough to be separated off from AMBER. Along with a few classes that deal with geometrical entities such as error points and cones, the bulk of the DRT is related to tracks. Its three major subpackages deal with the track classes themselves, the track fitters (see chapter 4) and the propagation of tracks through a magnetic field.

3.3.1 The Track Package

The track is the most central concept in the reconstruction of a high-energy physics experiment. Consequently, its applications are diverse, and so are the properties and functionalities assigned to it by different programs, or even by different sections within one program. Trying to come up with a single closed design to fit all these cases is doomed from the start. Hence, the first requirement of any track package must be formulated as:

1. Algorithm independence

The track classes must be general enough to be used by all reconstruction packages.

From this requirement alone, it follows that the track package can not consist of an explicit implementation, but instead can only define a framework; an extensible structure on top of which each program can implement its own classes. The following functional requirements only serve to make this framework complete, flexible and internally coherent [35].

2. Querying a track

A track must supply the following information:

- The track fit parameters, including their errors.
- The elements associated to a track such as its hits and vertices. A user must be able to supplement this list with additional types of his own.
- The quality of the track (fit).
- A user-definable type or status, e.g. to record whether the track came from a reconstruction of the hits in the muon or inner detector, which algorithm was used to find/fit the track, which magnetic field was used in the fit, etc.
- The truth information for Monte Carlo generated tracks.

3. Updating a track

A user must be able to update any of the fields listed in requirement 2.

4. Comparing tracks

It must be possible to determine whether one track is better than another, with

the user being able to define what “better” means.

5. Track selection and ordering

The track package must supply the architecture for selecting and ordering tracks based on their query and compare methods (see requirements 2 and 4).

6. Combining tracks

It must be possible to combine any two tracks, possibly of different type, with the framework making this as convenient as possible.

Requirements 2 to 5 are addressed in the next two paragraphs in which a first level design is presented. However, because that design does not fulfil requirements 1 and 6, it will be further enhanced in subsequent paragraphs.

Basic Design

The basic track package is one that can only be used by a single program. All the required functionality is there, but the contents of all classes and their interdependencies are explicitly defined, and there is very little flexibility.

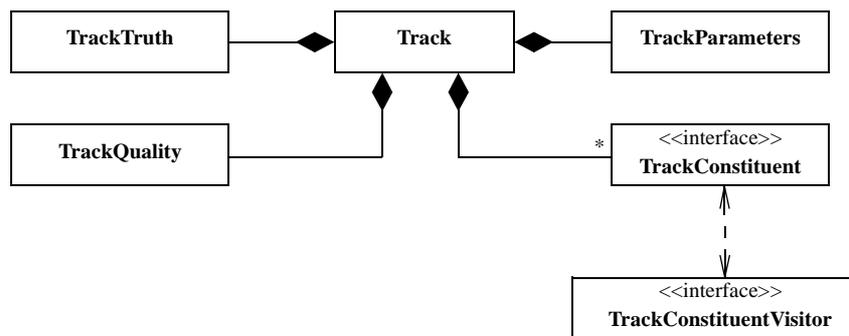


Figure 3.10 Basic design of the track classes.

The Track class itself is basically a container, storing information without providing any real functionality. This is necessary even in a design where all the functionality required of a track is known, because the number of methods of the Track class would otherwise proliferate. To this end, different classes are defined to hold the track parameters, its quality and the truth information. The fourth class, TrackConstituent, is the abstract base class of everything that can be associated with a track. Examples of this are hits, inert material (multiple scattering points) and vertices. As all these constituents are known by the program, the visitor pattern [32]

in the form of the `TrackConstituentVisitor` interface is an appropriate way to add functionality to them without cluttering up their interfaces.

Helper Classes

To allow for the diversity of operations that can and will be done to and with tracks, such as their building, extrapolation and fitting, they are separated off into an unlimited number of independent helper classes. These must of course be defined by the user, but for the selection and ordering of tracks (requirement 5) a template framework comparable to the D0 cuts package [36] can be defined.

Traits Design

The problem with the basic design presented above is that it does not satisfy the requirement of algorithm independence. Different programs have to define their own track package, even if they share most of the design. To solve this unnecessary duplication of code, the track package is made user-modifiable by introducing the “traits” [37]. The `Traits` class is nothing more than a collection of four type definitions, viz.

- `constituent_type` : The base class of the track constituents.
- `identifier_type` : The type by which all track constituents can be uniquely identified.
- `parameter_type` : The parameter set used by the track.
- `quality_type` : The quality (base) class of the track.

The type of the truth information has not been added to this, since it is based on the general Geant 3/4 format. However, if it should be needed in a later stage, it would be a trivial matter to add.

The `Track` and `TrackConstituent` classes are now parameterized with the `Traits` class, resulting in the design presented in class diagram 3.11. Because the `Track` class has become a template, it is beneficial to have a common base class e.g. as an interface to the outside world. This task is fulfilled by the `Trajectory` class. Its two methods, which are depicted in diagram 3.11 are related to the propagation of the trajectory through a magnetic field and are discussed in section 3.3.2.

The `TrackParameters`, `TrackConstituent`, and `TrackQuality` classes still exist, but they have now become interfaces from which the user can derive his own, or he can choose to use completely different classes. This second option would not have been possible without the traits. The first one of course already existed in the basic design, but the problem would then have been the absence of direct access to the user-defined descendants outside of the visitor pattern⁵.

5. Except by using dynamic casts.

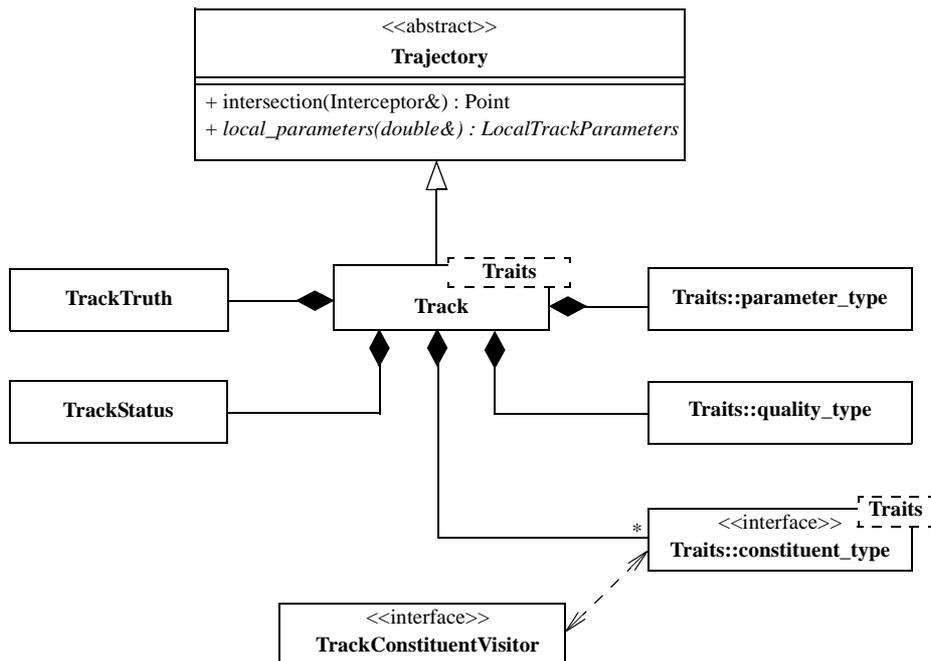


Figure 3.11 Traits design of the track classes.

One other addition is the `TrackStatus` class. It serves as the base class of an entire hierarchy of user-definable status classes. These classes do not have to have any state, as their functionality can be compared with that of the members of an enumerated type, with the difference that an enumeration can not be extended once it has been defined.

The `TrackConstituentVisitor` class also still remains, but is now an implementation of the acyclic visitor pattern [38]. Instead of defining just one visitor base class, it defines such a class for each constituent type. This not only makes a non-templated visitor class possible, but more importantly, it eliminates the otherwise necessary dependencies between the different constituent types.

COM Design

The Traits make it possible for the Track package to become a general toolkit, to be used by multiple programs. However, a specific `Track<Traits>` implementation in most cases still only makes sense for one application. If one wants to combine the results of two or more programs, say e.g. the tracks found in the muon spectrometer with those in the inner detector, then one is forced to add conversion constructors or operators in one or both of the classes, or to introduce wrapper classes. This would be a very inflexible approach, making one program dependent on possibly many others.

One way to solve the problem of how to transparently access the information supplied by any number of (unknown) sources is to use a Component Object Model (COM) [39]⁶. It is based on the principle that an object can make its functionality available through a number of interfaces. The calling party then only sees the interface he is interested in, and not the implementation behind it. And what is most important, interfaces can be added and removed without breaking the code that does not use them, and without even having to recompile it.

The central base class of these COM interfaces, as well as of the COM-enabled classes themselves, is the `IUnknown` class (see figure 3.12). It defines a number of `query_interface` methods, which return the requested interface when either the object itself, or the object that is hidden behind the interface supports it. The simplest way to turn a class into a COM-enabled one is to inherit it from the `COMObject<Object>` base class, with `Object` the type of the class that is to become COM-enabled. It manages the list of interfaces that its descendant `Object` implements. The second main class in diagram 3.12 is the `COMImplementation` template, which takes care of the administrative tasks required of an implementation of a COM interface.

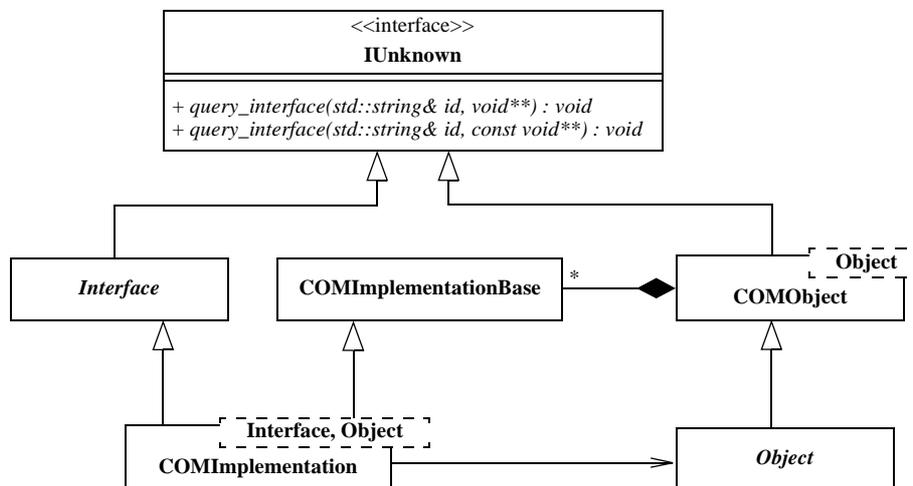


Figure 3.12 Implementation of the COM model as it is used by the track package. Note that a class name in italics doesn't embody a real class, but a type of class instead. In the DRT for example, `Object` can represent `Track` or `TrackConstituent`.

To add the Component Object Model to the track package, both `Track` and `TrackConstituent` are turned into COM-enabled classes by inheriting from `COMObject`. In the case of the `Track` class this is done to meet requirement 6 (combining tracks). The `TrackConstituent` class has been changed to support COM in order to have another way in

6. COM has been developed by Microsoft, and just as other similar solutions like CORBA, it is in its full form far too bulky for such a simple thing as a track package. Hence, the COM model used here is a simplified version, implemented by the utilities package [39].

addition to the (acyclic) visitor pattern to add functionality to its descendants. For an example of its use, see the track fits in sections 4.2.3 and 4.4.

3.3.2 Track Propagation in a Magnetic Field

Being able to define tracks is one thing, but they are pretty useless without their accompanying helper classes. One very important helper package is the propagation of a track through a magnetic field. The requirements on such a package are fairly straightforward [40]:

1. **Field value**

The package shall describe the magnetic field anywhere in the detector.

2. **Field gradient**

The package shall provide the gradient of the magnetic field at all places where it is able to provide a field value.

3. **Tracking by step**

The package shall be able to extrapolate a track including its error along a given distance taking into account the effect of the field, while ignoring physics effects such as multiple scattering, energy loss and particle decay.

4. **Tracking to a surface or volume**

The package shall be able to extrapolate a track to its intersection point with a surface or volume.

The first two requirements are easily fulfilled by the class hierarchy topped by the `MagneticField` interface (see figure 3.13). It declares query methods for the value and gradient of the magnetic field at any point in space. Two descendants, one for a constant field and the other for an ASCII-based field map, have currently been implemented by the DRT.

The actual propagation of a track is built around the `MagneticFieldTracker` class. It is a static class, i.e. no instances can be created and all clients see the same static state. It performs no real work, but is merely an engine executing the appropriate, user-definable classes around it. It does this in a three-step process:

1. First, it selects the step size based on the maximum allowed error per step and the gradient of the magnetic field.
2. Then it approximates the track and its errors by a local helix as defined by the `LocalTrackParameters` class.
3. Finally, it extrapolates the helix over a distance equal to the chosen step size (`SimpleStepTracking`, a descendant of `TrackingAlgorithm`), or using the Runge-Kutta algorithm. This `RungeKuttaTracking` class is used by default. In addition, the transport matrix of the local helix errors is calculated for the current step, and is added to a running aggregate maintained by the tracker.

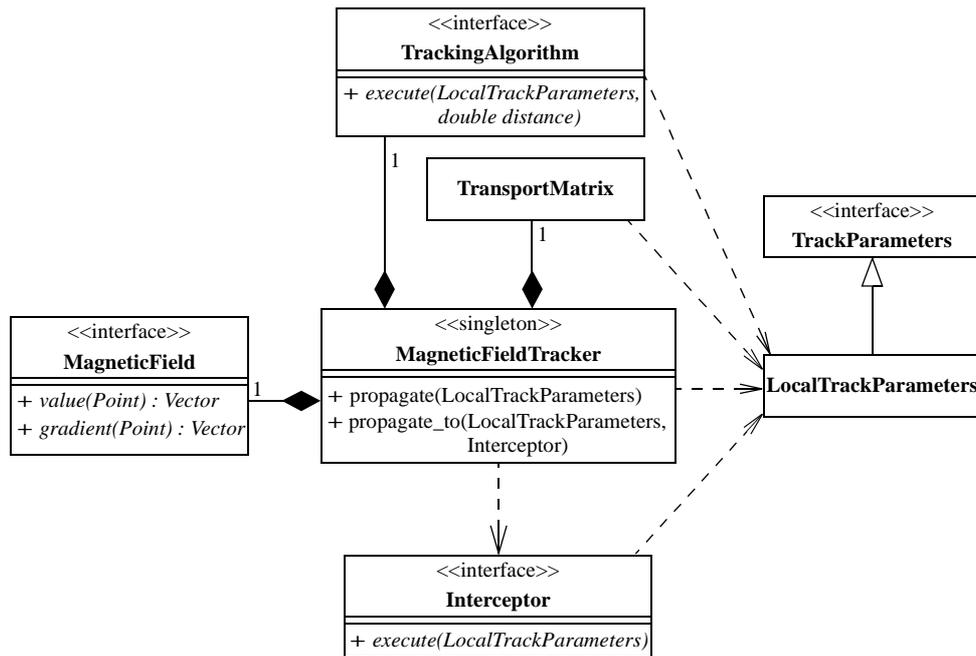


Figure 3.13 Interface view of the track propagation package.

These steps are repeated until the end of the propagation as defined by requirement 3 or 4 is reached, at which time `MagneticFieldTracker` updates the original parameters and their errors.

The last track-propagation requirement is satisfied by the `Interceptor` classes, which calculate the intersection of a track with the surface or volume defined by that interceptor. An interceptor template following the Template Method pattern [32], as well as implementations to work with the `Surface` and `Volume` classes of Arve have been defined. An example use of an interceptor is described in sequence diagram 3.14:

1. When a user wants to propagate a track to the surface of a cylinder, he creates a `SurfaceInterceptor` with the cylinder as an argument.
2. Then the interceptor is executed with the track as an argument.
3. The interceptor queries the cylinder to determine the position of the track relative to the location of the cylinder.
4. As long as the track has not intersected with the cylinder (the sign of its relative position has not changed), the track is propagated by the default step size.
5. When the track enters or leaves the cylinder, the propagation is reversed with a step size equal to half the distance to the surface as returned by the latter's `how_near` method. This process is continued until a certain accuracy has been

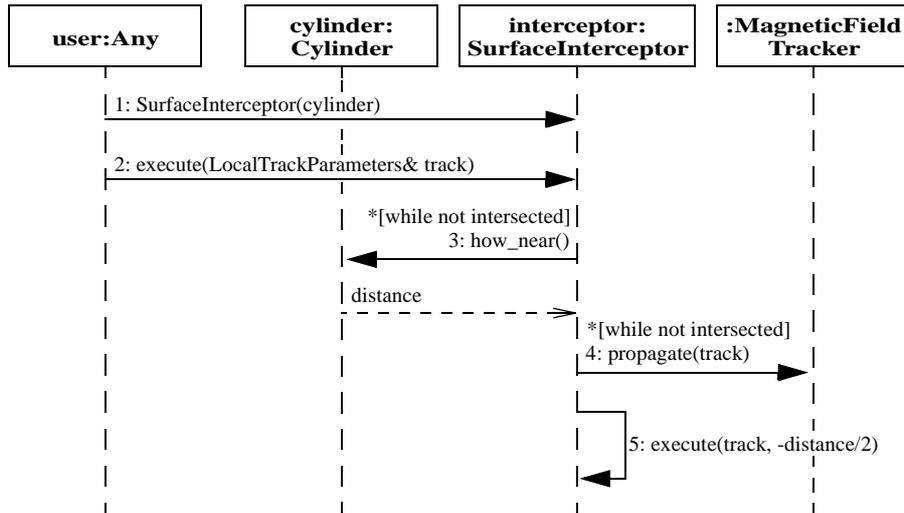


Figure 3.14 Propagation of a track to the surface of a cylinder.

achieved. Finally, track is updated to reflect the intersection point.

Instead of calling the execute method of the interceptor directly, this calculation can also be started by invoking the MagneticFieldTracker's propagate_to function. This duplication merely exists to complete the tracker's interface.

3.4 Generic Dataview Library

Reconstruction programs like most other software that is algorithm based, are to a large extent dataflow oriented: Starting with a certain set of data, a number of successive operations are performed to reach the sought-after results. The way these problems are generally solved is to build lists of objects, and then to write the functions that operate on them and create new lists. This decoupling of containers that store the data, and algorithms that work on it, is also present in the Standard Template Library (STL) [41] and is called generic programming⁷.

The containers and algorithms work together through so-called iterators. An iterator is an object that refers to a specific value within a container, and each container must supply two such iterators, one to its first value and the other to its end. Iterators come in five different flavours (input, output, forward, bidirectional and random-access [41]), each one with its own

7. The STL is part of the C++ standard library and it implements generic programming through the use of templates.

well-defined functionality. The algorithms base themselves on this functionality, and have therefore no need to know anything about the underlying container.

The Generic Dataview Library or GDL uses these iterator specifications to define its dataviews, which are basically iterators that adapt other iterators. Consider for example the simple algorithm shown in figure 3.15. It depicts the creation of track segments out of two hits, one from each of the detectors. These detectors are the real containers. They store the hits and supply the required iterators. The `Combinatorials` dataview that follows them is an iterator adaptor. Internally it stores two iterators, one to the current value of each detector. And its corresponding value is the pair created out of these current hits. Similarly, `Segment Builder` is an adaptor with one input. It transforms that input value, i.e. the pair of hits, and creates a track segment out of it. So in effect, a dataflow network as shown in figure 3.15 is nothing more than a chain of iterator adaptors linked together. Each of the adaptors represents another view on the data, hence the name dataview.

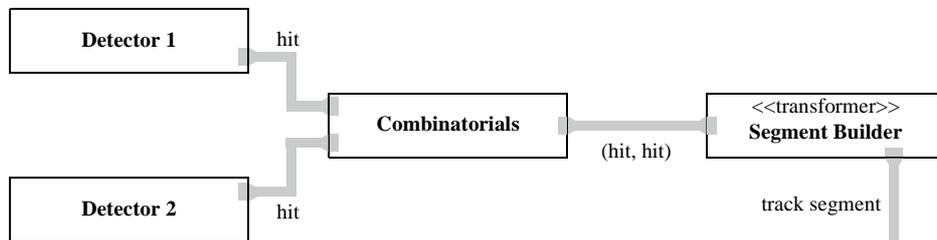


Figure 3.15 Example of a small GDL network. For an explanation, see appendix A.2.

A dataview is in many respects identical to a component, in that it completely decouples its interface from its implementation. The former is a combination of the iterator type it belongs to and the type of its output. The latter is the whole upstream network, i.e. its inputs. It can consist of only a single or more than a thousand dataviews, but the behaviour of the dataview remains the same. This means that a dataview completely encapsulates its upstream network.

An important feature of the dataviews is that they are of the data-pulling type. This means that it is only on the request of the user that something happens. For example, calling `operator++` on the `Segment Builder` of figure 3.15 results in a call to `Combinatorials` to look for the next pair of hits. It does this by advancing its internal iterator to `Detector2` by one, or when that iterator is at the end of the detector, to reset it to the first value and to advance the iterator to `Detector1` by one. This is contrary to the data-pushing approach in which every time a value changes, a number of registered functions are called. This would mean that the data and not the user is in control. In this scenario, whenever a hit is added to one of the detectors, `Combinatorials` would be called automatically. As can already be seen from this simple example, this would lead to a much more complicated programming logic.

Another consequence of this feature is that the dataview network is based on lazy evaluation. In the example above, the track segments are only built on request. When the querying of `Segment Builder` stops after the first segment, the others are never calculated. This is a major advantage over the list-filling approach in which an operation is applied to a whole list of values before the next operation is performed.

In a dataview network, the copying of data is reduced to a minimum. All dataview values are passed on through the network as references. And when a new object is to be created, it is stored in a reference-counted pointer. This has the added benefit that the object is automatically deleted when it is no longer needed.

3.4.1 Core Implementation

The standard way to define an iterator adaptor is to parameterize it with the type of the iterator it connects to. Although this would work fine in small programs, it doesn't scale very well. The reason for this is that almost every dataview would be a separate instantiation of the adaptor template, with the template argument containing the whole upstream network. As a result, the compile time and program size would increase with every new dataview that is used. So instead, the dataviews form a class hierarchy with common base classes at the top (see figure 3.16). The

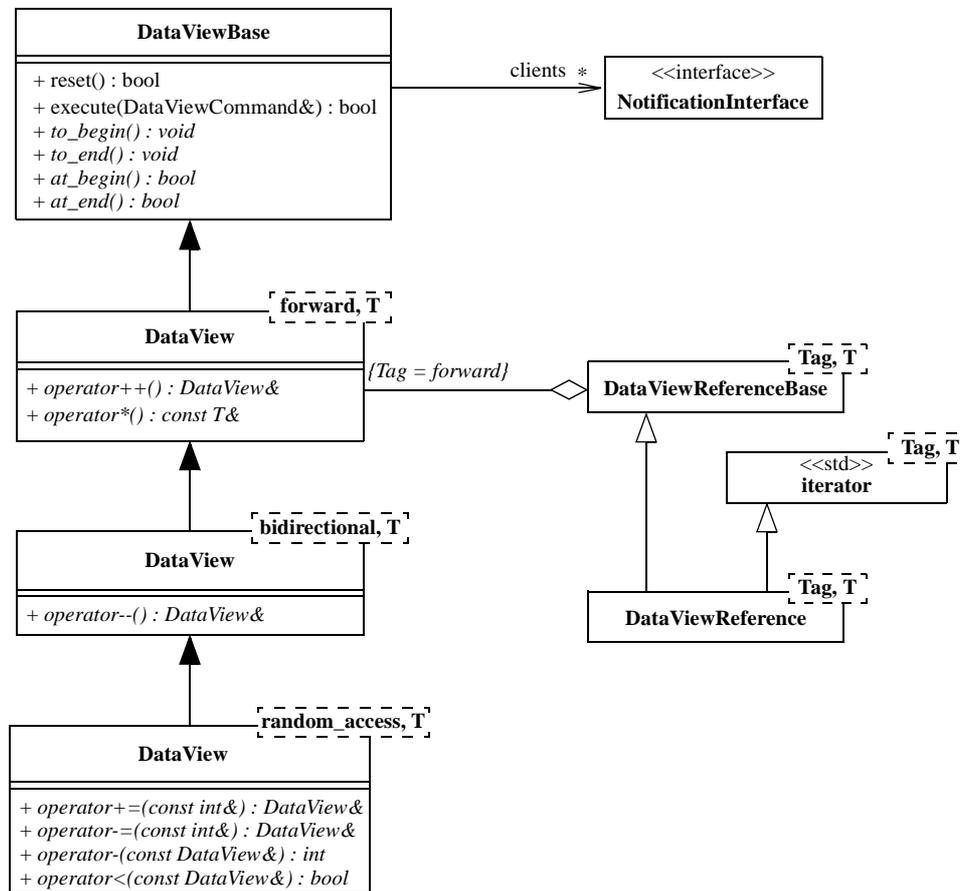


Figure 3.16 Dataview classes.

dataviews can then refer to these base classes, and don't have to know the exact type of the connections.

The type-independent behaviour of the dataviews is defined by the `DataViewBase` class. It stores a name and type, both of which can be removed from the program by setting a preprocessor directive, as well as a list of clients. These `NotificationInterface` descendants are notified when the dataview becomes invalid or is deleted. A large part of this list is formed by the downstream dataviews that adapt it, i.e. the dataviews that are connected to its output.

The most important methods of `DataViewBase` are listed in figure 3.16. Through the `reset` method it resets itself and the upstream dataviews to an empty state. For example, in the case of a container the method deletes its contents. The `execute` method can be used to execute a user-defined command: `DataViewCommand` is a typedef for the `CommandFunction1R` class template from the utilities package, taking a `DataViewBase` as its argument and returning a boolean value. The command is passed up through the network until a dataview is found that can handle it, or until the end of the chain is reached.

The remaining methods of `DataViewBase` serve to set the dataview to its begin or end state, or to test whether it is in one of those two states. In a normal STL application, the begin and end iterators are created by the corresponding methods of a container. But in the GDL the containers are hidden behind an unknown number of dataviews, and hence the dataviews must define this functionality themselves.

The second base class of the dataview hierarchy is the `DataView` class itself. It is parameterized with an iterator tag (forward, bidirectional or random-access functionality) and a value type, and is specialized on the former. As can be deduced from figure 3.16, the three dataview specializations inherit from each other so that e.g. a random-access dataview can be interpreted as a bidirectional one. The inheritance relationship is moreover inclusive to increase the flexibility when implementing specific dataviews (see also the next section).

All operators of the `DataView` class are abstract as they are to be filled in by the specific implementations. This causes some performance degradation but as explained above, this can not be avoided. Also, only the pre-increment and decrement operators are supported. The post-increment and decrement operators require the creation of a copy of the dataview, and hence of the whole upstream network, and that is an operation that could be very costly.

Another consequence of using dataview base classes is that internally pointers are used everywhere. To shield the user from this, dataviews can be wrapped inside `DataViewReference` objects. Like the `DataView` class it is specialized for the different tags, and the common behaviour has been factored out into a base class (`DataViewReferenceBase` in this case) to prevent code duplication.

As a dataview is in most cases an iterator adaptor, it must have one or more connections to other dataviews; its upstream network. It acquires the functionality for storing those connections by inheriting from a connection-type specific base class (see figure 3.17). Inheritance has been chosen in favour of aggregation, because it requires the least amount of effort on the part of the writers of the dataview implementations: They do not have to define any methods to access the connections, and don't have to be concerned with maintaining their state.

All connection base classes inherit virtually from `DataViewBase`, thereby granting them access to the state of the dataview. They are also all parameterized with a generic argument for

Containers

Containers are used to store items either permanently (`PersistentContainer`) or temporarily (`Container`). In fact they are not real containers but instead dataviews, i.e. iterators to the values of the containers. The actual container is hidden behind a `ContainerStubInterface` pointer (see figure 3.18). This allows `PersistentContainer` to use different container implementations. For example a `LayerDetectorStub`, a descendant of `ContainerStubInterface`, exists to interface to AMBER's detectors (see also sections 3.2.3, 4.1 and 4.2).

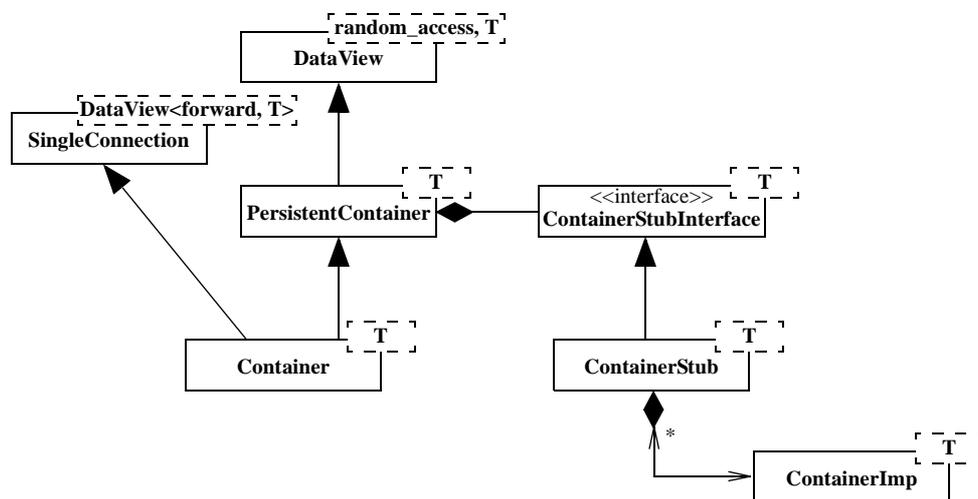


Figure 3.18 Container classes.

The second advantage of this separation between dataview and container is that it permits multiple `PersistentContainer` dataviews to share the same underlying container, thereby preventing the unnecessary duplication of its contents. The default implementation is formed by the `ContainerImp` class in conjunction with its `ContainerStub`. As can be determined from figure 3.18, `ContainerImp` knows about multiple `ContainerStubs`, which it all notifies when one of them changes the contents of the container. However, the `ContainerStubs` are the ones that own the `ContainerImp` object, and not vice versa, and when the last stub is deleted, it takes the container with it.

To come back to the dataviews, `PersistentContainer` is a random-access dataview without any connections, and whose contents is not affected by the reset method. Instead it defines `push_back`, `erase` and `clear` methods to manually alter the data it contains. Next, the `Container` dataview combines the functionality of `PersistentContainer` with a connection to another dataview. When queried for the first time, it loops over all the values of that connection and stores them in the container. This is useful when one wants to save intermediate results that are too expensive to be recalculated.

Transformers

The `Transformer<Tag, From, To>` dataview transforms the values of a connected dataview into new values of the type `To`, and has a `Tag` that is identical to the one of the connection. Like most other dataviews the real work is done by a derived implementation class called `TransformerImp`. It is parameterized with the type of the transformation to perform, so that any function class defining the appropriate function operator

```
To operator() (const From& arg) const
```

can be used. This separation leads to a friendlier interface for the user.

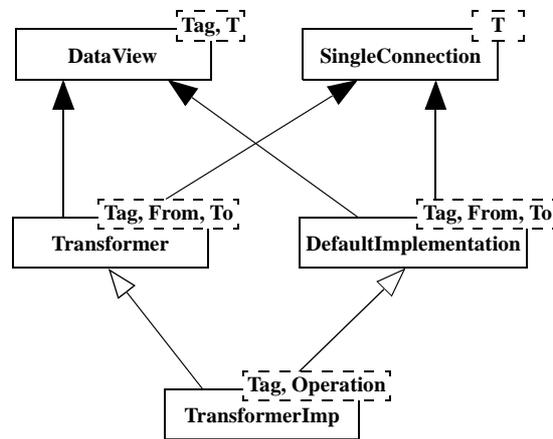


Figure 3.19 Transformer classes.

In this particular case, the increment and decrement operators are implemented by `DefaultImplementation`, which simply increments, respectively decrements the connected dataview as stored in its `SingleConnection` base class.

To complete this picture, the function

```
template <class Source, class Operation>
TransformerImp<typename Source::tag, Operation>*
transform(const std::string& name, Source* source, const Operation& op)
```

is provided to easily create a transformer dataview.

In addition to `Transformer`, the GDL also defines a `BinaryTransformer` class. It has two connections, the second of which is passive. This means that its state is not changed by the binary transformer, and only its current value is used as a second argument to the transformation operation. For the remaining part, `Transformer` and `BinaryTransformer` are identical.

Filters

The `Filter<Tag, T>` dataview filters the values of a connected dataview and lets through only those for which a user-defined predicate evaluates to true. The iterator tag of the filter is equal to the minimum of the tag of the source dataview and `bidirectional`, because it is not possible to implement the step operators, `operator+=(n)` and `operator-=(n)`, more efficiently than by calling `operator++`, respectively `operator--` n times.

`SortedFilter` is similar to `Filter`, but it only works on a random-access input whose values are sorted. By supplying two predicates, which define the lower and upper bound of the valid range of input values, the connection can be binary searched, increasing the speed of the program. This range is determined the first time the dataview is queried, after which the dataview has a random-access functionality.

Finally, `BinaryFilter` is to `Filter` what `BinaryTransformer` is to `Transformer`: It has a second, passive connection whose value it also passes on to the filter predicate.

ContainerModifiers

A `ContainerModifier` dataview can only connect to a `Container`, and is capable of updating the latter's contents as a whole, and not just one value at a time like e.g. the `Transformer` does. Its descendant `ContainerModifierImp` is parameterized with a unary function that must take a `Container` as its argument. When the dataview is queried for the first time, this function is applied to the container. One such operation, viz. `Sorter`, is supplied by the GDL and it sorts the container's contents.

When a `ContainerModifier` is connected to a dataview that is neither a `Container` nor another `ContainerModifier`, an intermediate `Container` dataview is created on the fly.

Wrapper

The `Wrapper` dataview creates for every value of its connection a new object that wraps that value. The `Wrapper` is automatically followed by a `Container` to store the wrapper objects as their state would otherwise be lost when the program continues with the next value. One wrapper object supplied by the GDL is `Used`, which adds a "used"-flag to the original object.

To complement the `Wrapper`, the `unwrap` function is provided, which creates a `Transformer` that returns the original, wrapped object.

Combinatorials

Two dataviews exist to create the combinatorials of the values of two connections. `Combinatorials` builds and returns all possible combinations of the values, while its counterpart `SortedCombinatorials` employs a selection criterion. It requires that the second connection is of the random-access type and that it is sorted for each value of the first connection. Its descendant `SortedCombinatorialsImp<Tag, Low, High>` is then able to perform a binary search with the help of the `Low` and `High` predicates (cf. `SortedFilter`).

The output of the combinatorials dataviews is a pair of reference-counted pointers to the current values of the two inputs.

Merger

The Merger adapts multiple dataviews, and dynamically merges their values into a single stream. The tag and value type of the first connected dataview determine the type of the Merger, and all subsequent connections must provide at least the same functionality as that first one. Connections can moreover be added and removed on the fly.

When using some of the dataviews presented above, the example network shown in figure 3.15 can be coded as follows⁹:

```
gd1::DataView<gd1::random_access, TrackSegment>* algorithm;
algorithm = gd1::transform("Builder",
                          gd1::combinatorials("Combine", Detector1,
                                              Detector2),
                          build());
```

Listing 3.2 Program to build the example network of figure 3.15.

The only function that is to be supplied by the user is `build`, which must define the algorithm to turn two hits into a track segment. When the hits in `Detector2` are sorted, a `SortedCombinatorials` dataview can be used instead of the `Combinatorials`, which would speed things up considerably when the number of hits in the second detector is large.

3.5 Conclusion

The pursuit of the Open Closed Principle has led to an ensemble of software packages that are far more general than the original task for which they were developed, i.e. that of muon reconstruction in the ATLAS detector. This makes it possible for e.g. the classes of the Detector Reconstruction Toolkit to be adopted by the rest of the ATLAS software community. Especially the track package has been found to be flexible enough for most people to be comfortable with it. In addition, the classes responsible for the propagation of tracks in a magnetic field are being evaluated by ATLAS. They are somewhat slower than the highly optimized Fortran version, but improvements are still possible. Independent of this, the track-propagation package has also been successfully ported to the software of the D0 experiment, requiring only minimal changes that have to do with their different `Point` and `Vector` classes.

9. All GDL classes and functions reside in the `gd1` namespace.

The dataflow networks of the Generic Dataview Library are so general that they go beyond the realm of physics. In fact, similar principles are found in commercial packages like Open Inventor, but these are in most cases not object oriented but instead of a procedural nature. As more and more people in the ATLAS collaboration become better acquainted with C++, it is hoped that more complicated looking software like the GDL will be more widely used. Because its principles correspond so well with the nature of reconstruction algorithms, it presents a really intuitive way of programming them.

Of course all packages are already incorporated into the ATLAS software as part of the AMBER program, which is the main component of the muon system domain. Because the official architecture is in a constant state of flux, AMBER itself will require continuous updating. For example, the Arve framework will need to be abandoned in favour of Paso [42], and with it the detector description, event structure and graphics will have to be changed. This will be a non-trivial task, but fortunately AMBER is layered such that these changes will impact only small sections of the program. Also, it is hoped that by making this step AMBER will be able to directly access the GEANT-simulated events [43, 44], thereby making a direct comparison with other programs possible.

As a final note, all of this flexibility must of course come at a price. As was already mentioned for the track-propagation package, this price is a decrease in program speed, caused by the requisite abstract (virtual) functions present in the various interface classes. A cost, which is small and which in our opinion is far outweighed by its benefits.

CHAPTER 4 Reconstruction Algorithm

*I hear and I forget.
I see and I remember.
I do and I understand.*

Confucius

The reconstruction of events in the ATLAS muon spectrometer is built on top of the AMBER framework (see section 3.2), utilizing the classes provided by the DRT (see section 3.3), with the actual reconstruction algorithm implemented in terms of the GDL (see section 3.4). From a global perspective, the reconstruction looks like the component diagram shown in figure 4.1.

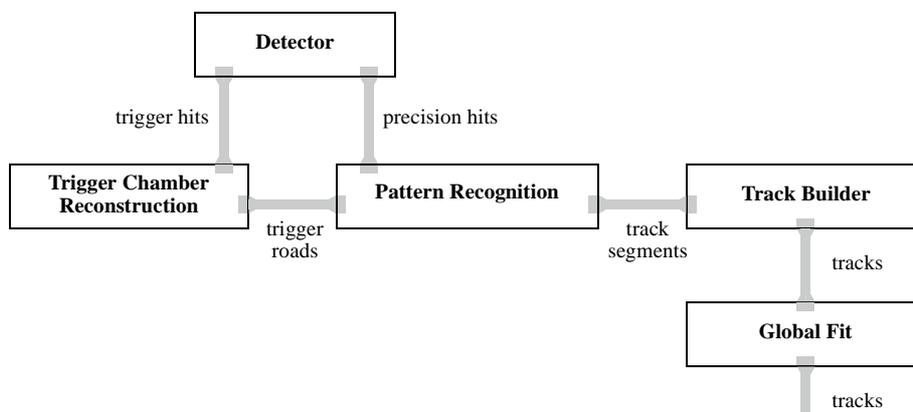


Figure 4.1 Global view of the muon reconstruction algorithm.

The hits in the trigger chambers (RPCs and TGCS) are retrieved from the detector layers with the help of a `ContainerStubInterface` descendant (cf. figure 3.18). These hits are used to build the regions of activity called trigger roads to which the subsequent reconstruction of the precision chambers is confined¹. The pattern recognition creates track segments out of the precision hits, which are when possible combined into tracks. And finally, a global fit of the track segments and the trigger hits is performed to determine the exact track parameters.

1. The precision-chamber reconstruction is not directly coupled to the `TriggerRoad` class, and in fact any descendant of the DRT class `RegionOfActivity` will do.

4.1 Trigger Chamber Reconstruction

The first step in the reconstruction of tracks in the muon spectrometer is the creation of trigger roads, i.e. regions of activity based on the hit information in the trigger chambers. They are needed to guide the reconstruction of the precision chambers because of the following reasons (see also section 2.3):

- The high background environment in the precision chambers requires the presence of a selection criterion with a high capability of rejecting the background hits if the execution time of the algorithm is to be kept in check;
- The large drift times of the precision chambers relative to the bunch spacing of the LHC make an efficient tagging of the bunch crossing to which a given track belongs by the chambers themselves impossible;
- The MDT chambers do not measure the azimuthal coordinate along the wire, which is needed to calculate the real drift time of a hit;

Because of their fast read-out and very low occupancy, the trigger chambers are very well suited for these tasks.

The algorithm for finding the trigger roads mimics part of the work that is performed by the level-2 trigger [45]. However, instead of trying to determine the momentum of the passing muon as is the task of the trigger, the goal of the algorithm here is to define a road that contains all the muon hits and a minimum of background hits. This is essential because all subsequent processing is limited to hits that lie inside the road.

The main advantage of the trigger algorithm as it is implemented here is that it is fast because it uses only the geometrical properties of the trigger chambers, and does not require any knowledge of the hits in the precision chambers, nor of the magnetic field. It is described in sections 4.1.3 and 4.1.4 for the low- and high- p_T trigger respectively, but first the reconstruction of the individual chambers is explained.

4.1.1 The RPC Chambers

The RPC chambers provide the trigger information in the barrel of the muon spectrometer. They consist of two layers, each one based on a gas gap around which two strip planes provide respectively the ϕ - and η -coordinate of a track (see figure 4.2). Each such plane is represented by a detector in AMBER's detector description, and its digits are the individual strips that are hit by a particle.

When such a particle crosses a strip close to its edge, neighbouring strips can also fire, resulting in multiple digits being generated by a single track. Therefore, the first step in the reconstruction is to cluster adjoining digits (see figure 4.3). A cluster is based on the `ErrorPoint` class provided by the DRT. It stores a position corresponding to the centre of gravity of the digits

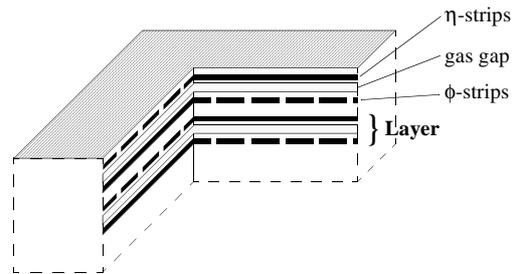


Figure 4.2 Schematic view of an RPC chamber (the internal representation is not shown to scale).

that make up the cluster, and a 3-dimensional error based on their extent. When the goal of the trigger reconstruction would have been to calculate the position of the track, a scale factor of $\sqrt{12}$ or higher² could have been applied to this extent. But as we are searching for the boundaries within which the particle has traversed the detector, the whole extent must be taken into account.



Figure 4.3 Reconstruction of a plane of detector elements (see also the “Containers” paragraph in section 3.4.2).

Based on the ATLAS trigger definition as described in chapter 2, one cluster in each projection is the minimum requirement for a trigger signal to be generated by an RPC chamber. This means that from this point on there are two possible ways to proceed. The first is to combine the clusters of the η - and ϕ -planes that make up a layer. This approach fails however when a particle generates a hit in only one of the two planes. When such a hit is combined with one of the uncorrelated clusters in the other plane, its size is incorrectly restricted in the dimension that is measured by the second cluster. Therefore, the only solution is to keep all original clusters, but this not only increases the number of combinatorials in the reconstruction that follows, but also requires an extra step at its end in which duplicate clusters have to be removed.

The other strategy, and the one that has been adopted, is to first combine the clusters in the planes that have the same orientation, i.e. either the ϕ - or the η -planes. In this case only the

2. When a cluster contains two digits, the position of the track can be inferred to have been close to the boundary between the two strips, except of course when one of the digits was caused by a δ -ray or any other source of background.

original clusters that have not been used need to be saved for the next step. The algorithm for the reconstruction of such a doublet of detector planes is shown in figure 4.4.

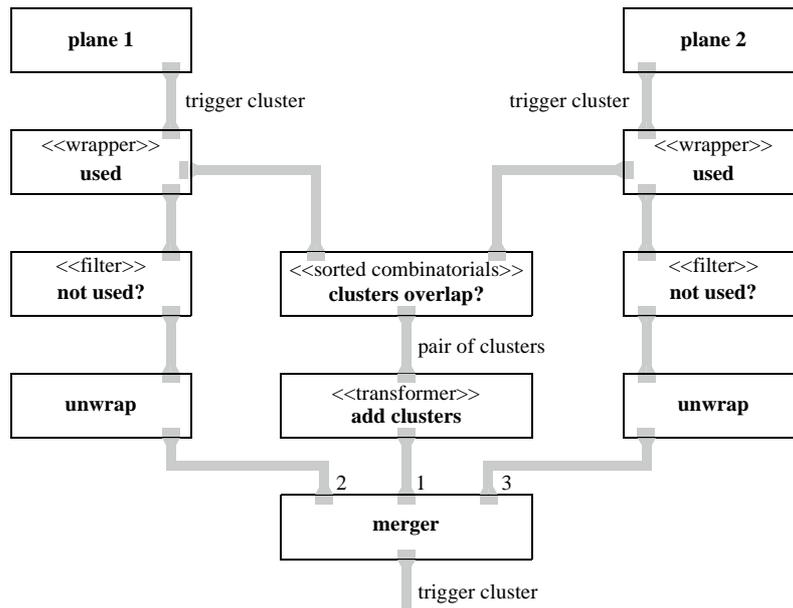


Figure 4.4 Reconstruction of a doublet of detector element planes.

When two clusters, one from each plane, overlap or are close enough as defined by the user, they are added together into a single cluster. Because the clusters, like the digits they are based on are sorted, a SortedCombinatorials dataview³ can be used for this process, in conjunction with a Transformer. The latter takes as its input the pairs of clusters coming out of the SortedCombinatorials and for each pair calculates their total extent and sets the origin equal to their centre. As a last step, a Merger concatenates the list of these clusters with the original ones that were not used in the combinatorials.

The final step in the reconstruction of an RPC chamber is to take the combinatorials of the η - and ϕ -clusters. Because according to the ATLAS trigger logic at least one hit is required in each projection, and because there is no way to determine which clusters belong together, all combinations of the clusters of the two doublets must be taken (see figure 4.5). Such a combination is formed by calculating the weighted sum of the two clusters, which results in a cluster the size of their overlap region.

3. The SortedCombinatorials dataview both constructs the combinatorials of the values of its two inputs and applies a filter on the created pairs. Because of its knowledge about the ordering of the input values it can use a binary search algorithm, which makes it (much) faster than when these two operations were applied separately (see also section 3.4.2).

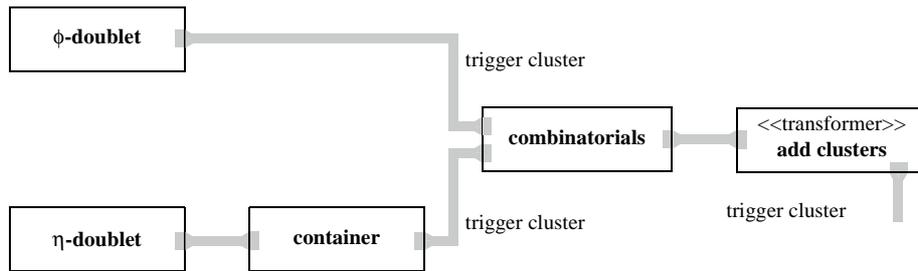


Figure 4.5 Reconstruction of a trigger chamber.

The RPC layers

The RPC chambers are arranged in three cylindrical layers, consisting both of large chambers in the odd ϕ -sectors, and of small chambers in the even sectors (see figure 2.3). As a particle coming from the interaction point can cross such a layer only once, it makes sense to combine the reconstructed clusters from the chambers that make up a layer into a single stream (see figure 4.6). The clusters are sorted in ϕ and not in η , because the roads are much narrower in the ϕ -projection where there is hardly any magnetic field that can cause the tracks to bend. As a result, a filtered combinatorials on ϕ later on in the reconstruction will give the most reduction in the number of combinations.

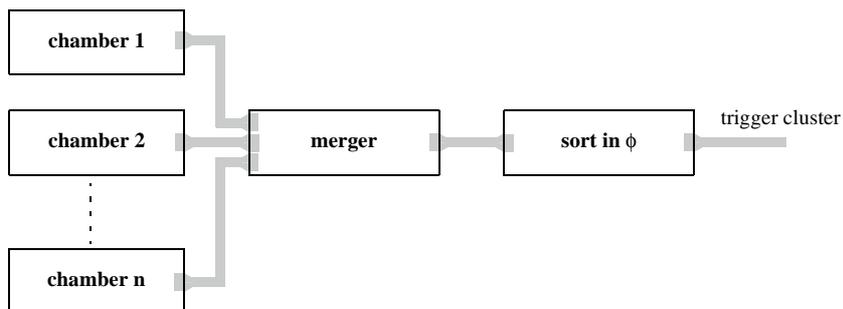


Figure 4.6 Reconstruction of a trigger layer.

4.1.2 The TGC Chambers

In the endcaps the trigger information is provided by the TGC chambers. These are multiwire proportional chambers of which three different types are used, depending on their position within ATLAS (cf. figure 2.7). In the innermost TGC0 layer, the chambers consist of only two wire planes. These wires, which measure the azimuthal coordinate are grouped together, 4 to 20 at a time. Such a wire-group behaves just like a strip from the perspective of the

reconstruction, and hence the algorithm described above for the RPCs can be reused here. Only, the building of the combinatorials out of the two different projections as shown in figure 4.5 must of course be skipped. For the second type of TGC chambers, which consist of two wire planes in conjunction with two strip planes (see figure 4.7b) even this small deviation from the RPC algorithm is not needed.

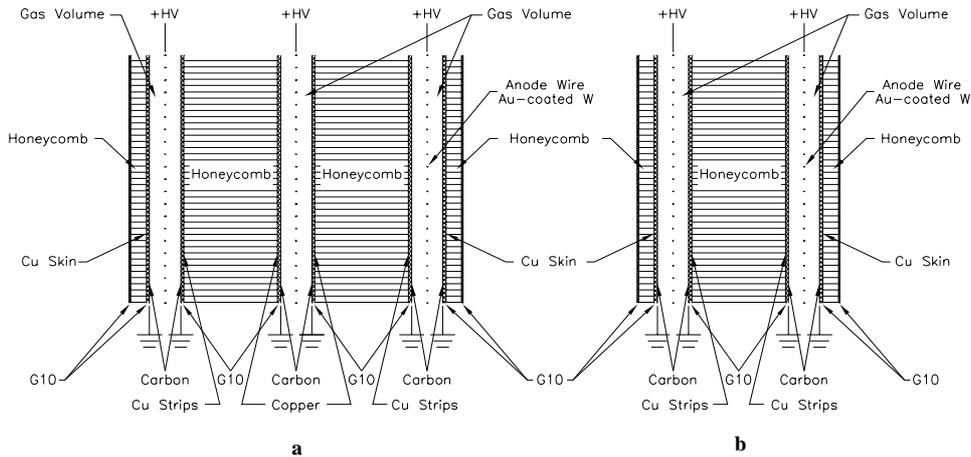


Figure 4.7 Schematic view of a triplet (a) and of a doublet (b) of TGCs (the gas gap is not shown to scale).

That leaves the triplets, chambers consisting of two strip and three wire planes. For the strips, the standard algorithm can be used, but for the wires a specialized version must be developed. The ATLAS trigger logic states that a 2 out of 3 coincidence is required (cf. section 2.3), and so all three the combinatorials of 2 layers each are taken (see figure 4.8). Of course, this procedure overestimates the number of real clusters, as a track can create hits in all three layers. Therefore, as a last step, the clusters that are compatible with each other are combined into a single one.

The result of the reconstruction so far is the creation of 11 trigger layers (three in the barrel and four in each endcap), which can be used to create the trigger roads. The low- p_T roads are constructed first, after which an attempt is made to extend them into the high-momentum regime. These algorithms are in no way dependent on the RPC or TGC background of the trigger layers, and use only the positions and sizes of the generic trigger clusters.

4.1.3 Low- p_T Trigger

The low- p_T trigger is a 6 GeV trigger based on a 3 out of 4 coincidence in each projection in the two middle RPC or in the two outer TGC layers (see figure 2.7). Because the individual chambers were reconstructed based on a 1 out of 2 coincidence per projection, by combining

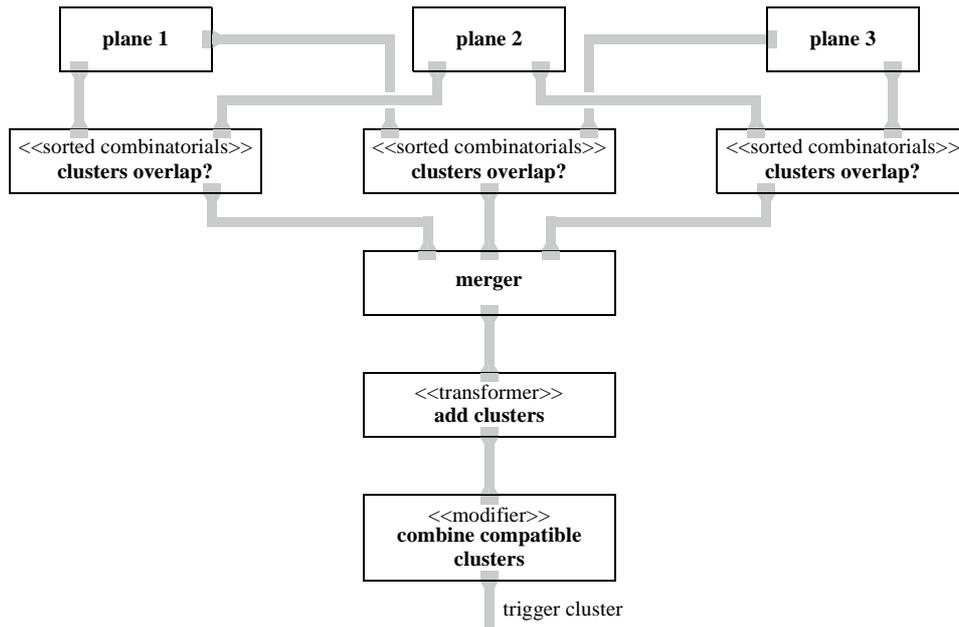


Figure 4.8 Reconstruction of the three wire planes of a TGC triplet.

the clusters of the two trigger layers, the reconstruction algorithm is capable of finding all tracks that pass the ATLAS trigger, and in addition the ones that leave only one hit per projection in each layer.

The algorithm itself is quite straightforward. When two trigger clusters are close enough in ϕ and η , they are combined into a trigger road (see figure 4.9). As the clusters are sorted in ϕ , the combinations are created with the help of a SortedCombinatorials dataview. The filtering

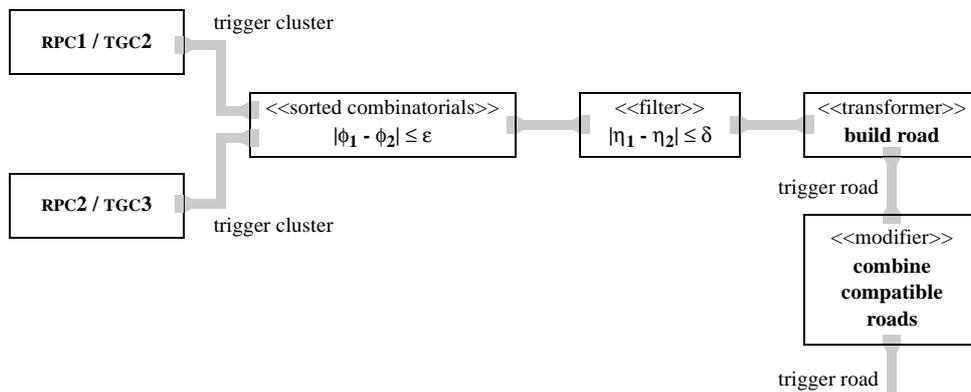


Figure 4.9 Low- p_T trigger reconstruction.

in η is subsequently performed by a regular `Filter` class, after which a `Transformer` is responsible for the creation of the roads in the form of `TriggerRoad` objects.

The `TriggerRoad` class is derived from DRT's `ErrorCone`, which represents a three-dimensional cone with variable ϕ - and η -shapes (see figure 4.10). For the low- p_T trigger, the ϕ -shape has the form of an hourglass with a width and opening angle based on the size of the two clusters. In the η - or bending plane a chalice shape is used. Both of its sides are helices aimed away from the axis of the cone. For ATLAS these are set to 6 GeV trajectories in a 0.5 Tesla field. An actual estimate of the momentum of the track from the two clusters that form the road is not possible because their separation too small compared to their size [46].

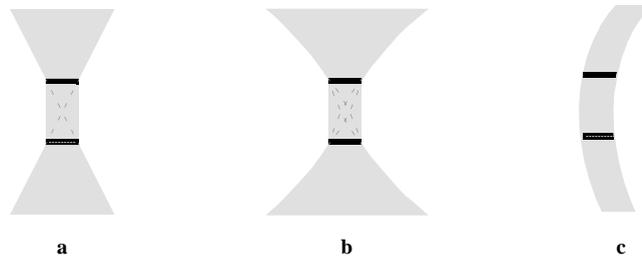


Figure 4.10 The different trigger road shapes used by the reconstruction, viz. hourglass (a), chalice (b) and helix (c). All of them can be used in both the ϕ - and η -projections, but as the toroidal field of the ATLAS muon spectrometer only bends tracks in η , (a) is used as the ϕ -shape, while either (b) or (c) form the η -shape.

As a final step in the low- p_T reconstruction, compatible roads that can be found in the overlap regions of the small and large chambers are combined.

4.1.4 High- p_T Trigger

The results from the RPC and TGC low- p_T triggers are combined into a single list, and an extension into the high- p_T regime is attempted. To that end, the clusters in the RPC3 and TGC1 layers are merged together and subsequently sorted in ϕ (the `high-pT clusters` dataview in figure 4.11). To match the clusters to the trigger roads, a `SortedCombinatorials` on ϕ and a `Filter` on η are used. When both are successful, the cluster is added to the road. As a result the shape in the ϕ -projection is narrowed, and the η -shape is changed into a helix form when the sign of the track's charge could be determined. Otherwise, it remains a chalice-shape, all be it a reduced one.

The list of newly created trigger roads is augmented by the original low- p_T ones that could not be extended into the high- p_T regime. As a last step, a network identical to the one shown in figure 4.11 is used to try to refine these roads with the clusters found in the TGC0 layer, i.e. the innermost TGC chambers that only measure the azimuthal coordinate.

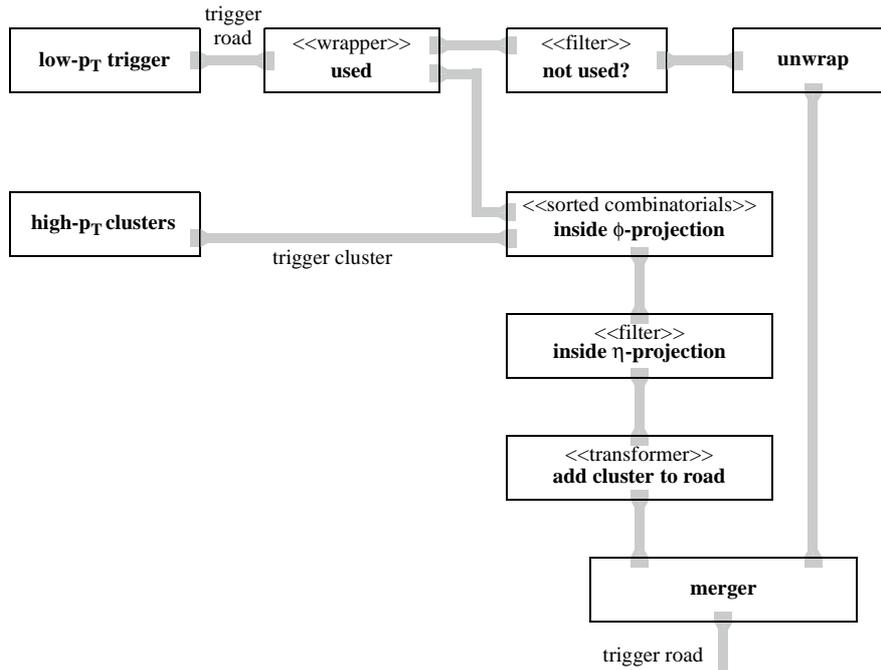


Figure 4.11 High- p_T trigger reconstruction.

4.2 MDT Pattern Recognition

The trigger reconstruction is followed by the pattern recognition in the precision chambers. The ATLAS detector contains two types of these chambers, viz. the MDTs and the CSCs (cf. figure 2.4). The latter, which are only used in the inner forward regions, are not implemented by the ATLAS simulation program and have been replaced with MDTs instead. The reconstruction as implemented by AMBER will therefore do the same.

4.2.1 Local MDT Reconstruction

All MDT chambers consist of either one or two multilayers, containing three to four tube layers each (cf. figure 2.6). In the barrel these chambers are arranged in so-called ladders, i.e. rows of chambers adjacent in z (i.e. the beam axis), that belong to the same detector layer (i.e. cylinder), ϕ -sector and side of the muon spectrometer (cf. figure 2.4). The corresponding entity in the endcaps is a sector of a MDT wheel, but for the remainder of this chapter, it too will be referred to as a ladder.

Seen from the interaction point, a ladder is a surface that a track can pass only once. Furthermore, the chambers are so close together in z that a particle can easily cross two

neighbouring chambers. To exploit this first feature, and to make the reconstruction independent of the second effect, the digits from identical tube layers of all the chambers in a ladder are grouped together (see figure 4.12). These layers are attached to the merger in such a way that the digits coming out of it are sorted in z for the barrel and in r for the endcaps.

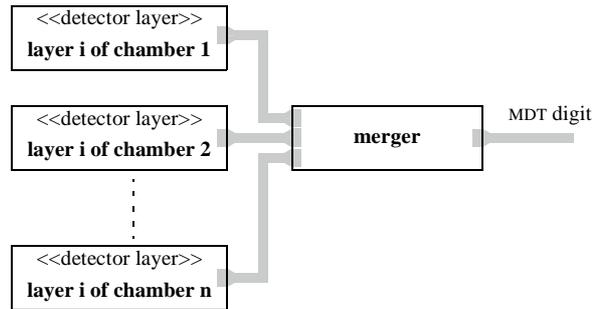


Figure 4.12 Definition of the reconstruction algorithm for the i -th tube layer in a ladder.

A chamber can have anywhere between three and eight of these layers, which means that there are just as many lists of digits in the reconstruction of a ladder. To simplify the pattern recognition that is to follow, these lists are combined into a single one (see figure 4.13). A downside of this approach is that topology requirements on the hits can only be checked by querying the digits for their identifiers⁴.

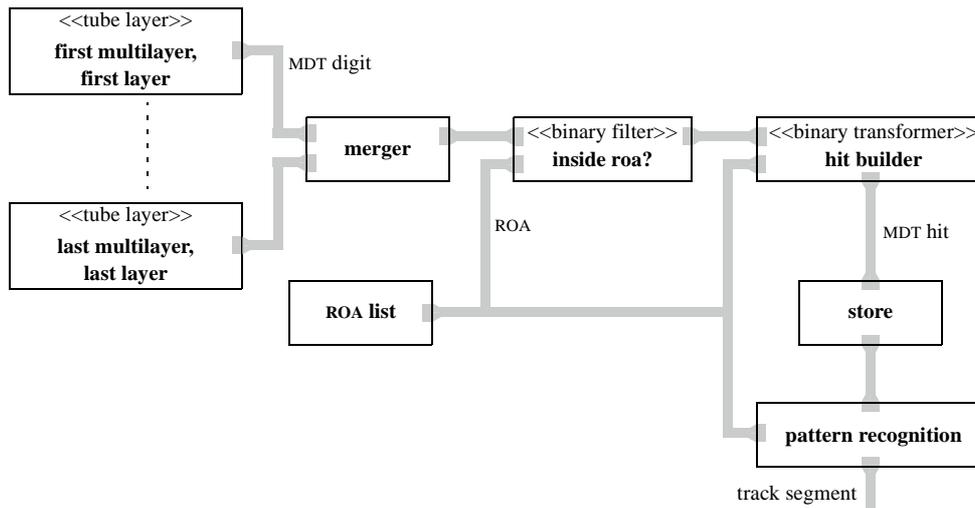


Figure 4.13 Reconstruction of an MDT ladder.

4. An example is the requirement that in a track segment at least one hit should come from each multilayer.

The current region of activity is then used to discard all digits that do not lie (partly) within it. In normal operating mode these ROAs are the trigger roads calculated by the trigger reconstruction (see the previous section), but any other source will do just as well. The digits that pass this selection are transformed into hits. It is not possible to convert all digits at the beginning of the reconstruction, because a ROA is needed for the determination of the second-coordinate position. This is the same ROA as was used by the filter. In fact, a single region of activity is used throughout the whole local MDT reconstruction and pattern recognition. Only when all tracks that can be created have been built, is the next region retrieved (see also section 4.3).

The creation of a MDT hit starts by subtracting from the digit's drift time the time it takes the signal to propagate along the MDT wire to the front-end electronics. To determine this time, the centre of the overlap region of the wire with the ROA is used, together with a user-definable signal speed. Subsequently, the time is corrected for the time-of-flight of the particle from the interaction point to the MDT tube. Here a straight line approximation of the track is used, which introduces an error well below the resolution of the detector. The resulting drift time is then converted to a distance by the detector to which the digit belongs, and a correction for the Lorentz angle is applied.

The error on the drift distance is determined based on the error in the r-t relation and the length of the section of the wire that falls inside the ROA. The latter has an effect on both the signal propagation time and on the time-of-flight correction.

4.2.2 Pattern Recognition

From the list of hits that lie inside a region of activity, all possible track segments are created by considering every combination of two hits. First a check is performed to determine whether the pair is valid, i.e.:

1. It is not part of any previously created track segment.
2. The line connecting the wire positions of the two hits points within a certain error to the interaction point. Because the hits lie so far from the origin, there is no need to take the drift circles into account in this step.

When a hit pair passes these tests, the four possible combinations of their left/right ambiguities are examined. For each, the following tasks are performed:

3. An initial track segment is created given by the formula (see figure 4.15 for an explanation of the variables used):

$$\alpha = -\operatorname{atan}\left(\frac{x_2 - x_1}{y_2 - y_1}\right) + n \cdot \operatorname{asin}\left(\frac{r_1 + m \cdot r_2}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}\right) - n \cdot \frac{\pi}{2} \quad (4.1)$$

$$b = y_1 - x_1 \tan(\alpha)$$

with $m = \pm 1$ for the left/right side of the first hit, and $n = \mp 1$ for the left/right side of hit number 2. Furthermore, y_2 must be larger than y_1 .

4. The hits that lie within a certain user-definable distance from the segment are added to it. All hits within the list are tested for their compliance to this rule, which should not be a problem as in most cases the number of hits inside a ROA is small.

When the track segment still has only two hits, it is discarded and the next one is tried.

5. A straight line is fitted through the hits as described in the next section.

From the four created track segments out of each original pair of hits only the best one is kept, where “best” is determined based on the quality of the fit and the topology of the hits. As a last step, the direction orthogonal to the drift plane is added to the segments by copying it from the current region of activity. They are then passed on to the global reconstruction of the precision chambers as described in section 4.3.

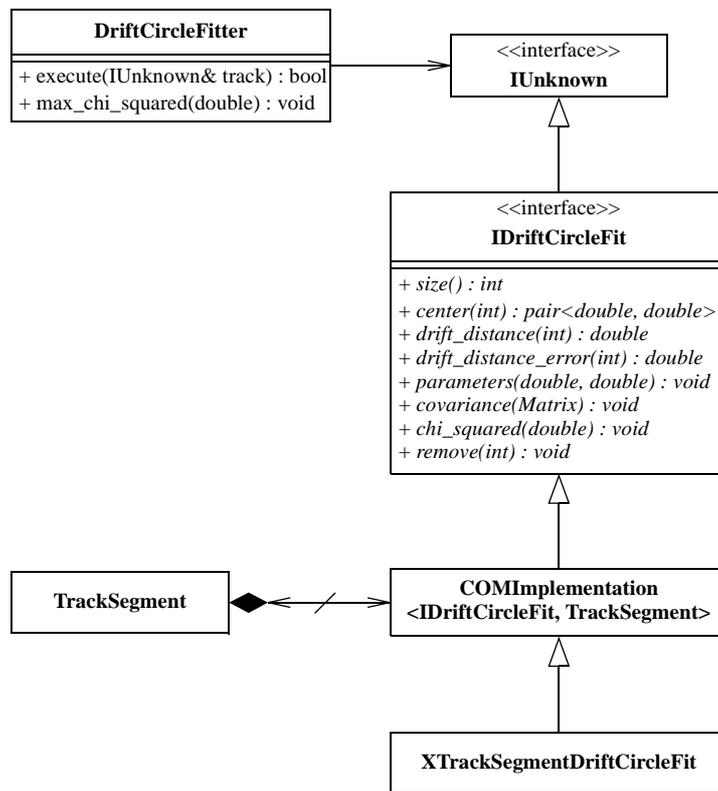


Figure 4.14 Classes involved in the straight-line fit through a number of drift-circle hits (cf. figure 3.12).

4.2.3 Drift-Circle Fit

A straight-line fit to the drift-circle hits belonging to a track segment is implemented with the help of the COM mechanism as described in section 3.3.1. The actual fit is performed by DRT's `DriftCircleFitter` class through its `execute` method (see figure 4.14). It takes an `IUnknown` object (e.g. a track or a COM interface to a track) as its argument, which is queried for its `IDriftCircleFit` interface. When it does not exist, the fit terminates with an error. The fitter uses the `IDriftCircleFit` interface to retrieve the hit information from the track on the one hand, and to store the results of the fit on the other hand. For AMBER's `TrackSegment` an `XTrackSegmentDriftCircleFit` implementation exists to provide the required functionality.

As the drift-circle hits are to all intents and purposes two-dimensional, so is their fit. The straight-line track segment is therefore given by

$$y = \tan(\alpha) \cdot x + b \quad (4.2)$$

with α and b the free parameters. The `XTrackSegmentDriftCircleFit` defines the x -axis as the pitch direction of the chamber, i.e. the global z -axis in the barrel and the radial direction in the endcaps. The y -axis is defined along the chamber's height, i.e. r in the barrel and z in the endcaps.

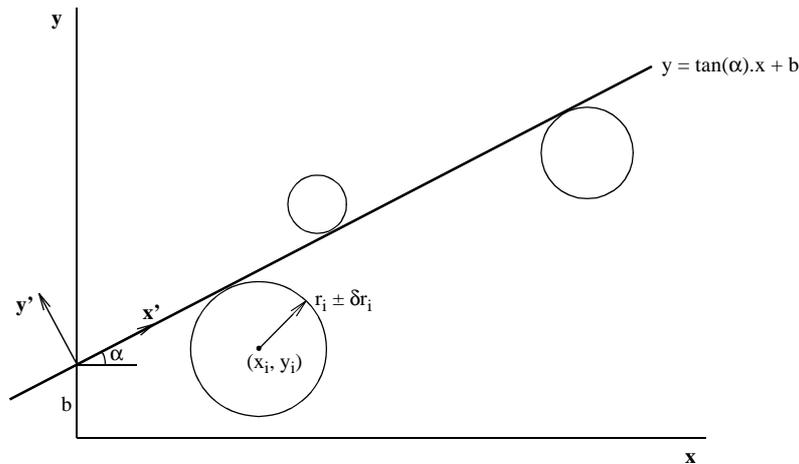


Figure 4.15 Definitions of the variables used by the drift-circle fit.

To determine the chi-squared of the track, a new coordinate system is chosen that lies alongside it. In this system, the distance of closest approach between a MDT wire and the track is given by the y'_i coordinate of that wire. Therefore,

$$\chi^2 = \sum_{i=1}^n \left(\frac{y_i' - r_i}{\delta r_i} \right)^2 = \sum_{i=1}^n \left(\frac{-x_i \sin \alpha + (y_i - b) \cos \alpha - r_i}{\delta r_i} \right)^2 \quad (4.3)$$

with the drift distance being a signed quantity. It is positive for the right side (in x) and negative for the left side. It is not possible to change the side of the hit during the fit, because that would compromise the stability of the algorithm: The chi-squared would have multiple minima at the various combinations of $\pm r_i$ ($i = 1 \dots n$), and especially in the case of small drift distances the fit would start to oscillate between them.

The derivative of the chi-squared with respect to b leads to the first of the two equations that are used to solve α and b , viz.

$$(b\Delta - S_y) \cos \alpha + S_x \sin \alpha = -S_r \quad (4.4)$$

with

$$S_\mu \equiv \sum_{i=1}^n \frac{\mu_i}{\delta r_i^2} \quad S_{\mu\nu} \equiv \sum_{i=1}^n \frac{\mu_i \nu_i}{\delta r_i^2} \quad (\mu, \nu = x, y, r) \quad (4.5)$$

$$\Delta \equiv \sum_{i=1}^n \frac{1}{\delta r_i^2}$$

The other equation follows from the derivative of the chi-squared with respect to α , which after substituting the expression for b from equation 4.4 becomes

$$\bar{S}_{xy} \cos(2\alpha) + \frac{1}{2}(\bar{S}_{yy} - \bar{S}_{xx}) \sin(2\alpha) = \bar{S}_{xr} \cos(\alpha) + \bar{S}_{yr} \sin(\alpha) \quad (4.6)$$

with the constant factors defined as

$$\bar{S}_{\mu\nu} \equiv S_{\mu\nu} \Delta - S_\mu S_\nu \quad (\mu, \nu = x, y, r) \quad (4.7)$$

This equation cannot be solved analytically, and hence it must be done iteratively:

1. The value of α is guessed based on the centres of the first and last hits.
2. This value is substituted in the right-hand side of equation 4.6, which can be solved to give a new value for α , viz.

$$\alpha = -\frac{1}{2}\beta + \frac{1}{2} \operatorname{asin} \left[\frac{\Lambda}{\sqrt{\bar{S}_{xy}^2 + \frac{1}{4}(\bar{S}_{yy} - \bar{S}_{xx})^2}} \right] \quad (4.8)$$

with Λ the result of the right-hand side of equation 4.6, and β defined as

$$\beta \equiv \text{atan} \left[\frac{2\bar{S}_{xy}}{\bar{S}_{yy} - \bar{S}_{xx}} \right] \quad (4.9)$$

3. Step 2 is repeated until α converges or until a user-defined maximum number of iterations has been reached. In most cases this method is found to converge within 3 to 4 steps.

The errors in the track parameters can most easily be calculated when the (x', y') coordinate system is used because in it, equations 4.4 and the one leading to 4.6 can be linearized in α and b without loss of accuracy. Rewriting them in a matrix formalism shows that the inverse of the covariance matrix is given by

$$C(\alpha', b')^{-1} = \begin{bmatrix} S_{x'x'} - S_{y'y'} + S_{y'r'} & S_{x'} \\ S_{x'} & \Delta \end{bmatrix} \quad (4.10)$$

To determine the errors at the centre of gravity of the hits, a shift is applied to the x_i' values such that $S_{x'}$ becomes zero. Inverting the covariance matrix is then a trivial matter, and results in the following errors^{5, 6}:

$$\begin{aligned} \sigma_\alpha &= \sqrt{1 / (S_{x'x'} - S_{y'y'} + S_{y'r'})} \\ \sigma_b &= \sqrt{1 / \Delta} \end{aligned} \quad (4.11)$$

The covariances are of course zero.

As an additional feature, the `DriftCircleFitter` is capable of removing hits from a track when their chi-squared exceeds a certain threshold. The `max_chi_squared` method can be used to set this value, and when the chi-squared of the worst hit is higher, it is removed and the S_{UV} factors are updated. The advantage of the algorithm described here is that these factors do not have to be recalculated from scratch, but instead follow from the original ones by merely subtracting the contribution from the bad hit. After that, the iterative process described above can restart.

The `DriftCircleFitter` class continues to remove hits as long as one of them has too large a chi-squared, and the number of hits that will be left is at least equal to two. When multiple hits are removed in this fashion, there is no guarantee that the final track created by the fitter is the best one based on the original set of hits. Instead, this responsibility has been delegated to the pattern recognition. The `Pattern Recognition` dataview stores a history of all created

5. The errors in α and α' are the same as the two angles differ only by a constant factor.

6. The stated error in b is actually the offset error perpendicular to the track. In the rotated (x', y') frame these two are identical, but not so in the original coordinate system. However, it is this standard error that is used throughout the remainder of this thesis.

tracks; tracks that do not contain the hits that were removed during the fit. Hence, tracks based on those hits will be created in the subsequent processing steps of the pattern recognition. This means that in the end all possible track segments have been built by it, after which a filter can be applied to select only the best one(s).

4.3 Global Reconstruction

After having reconstructed the regions of activity from the trigger hits, and having used them to find the track segments in the individual precision chambers, the final step in the reconstruction is to match these segments together and to build the global tracks. To this end, the pattern recognition is followed by a filter to select only those segments that pass certain cuts. The default criteria applied by this filter are defined as follows:

- If a segment crosses both multilayers, its number of hits must be higher than or equal to the number of layers in the chamber minus 1;
- If it crosses only one multilayer, it must have at least the same amount of hits as the number of layers in that multilayer.

The segments created by the MDT ladders that belong to the same detector layer⁷ are grouped together, and these layers form the inputs to the global reconstruction algorithm (see figure 4.16). The track builder is responsible for matching the various track segments and adding them to a track skeleton. This process is started with the segments in the outermost layers, because they have the lowest occupancy. The track builder is capable of applying certain

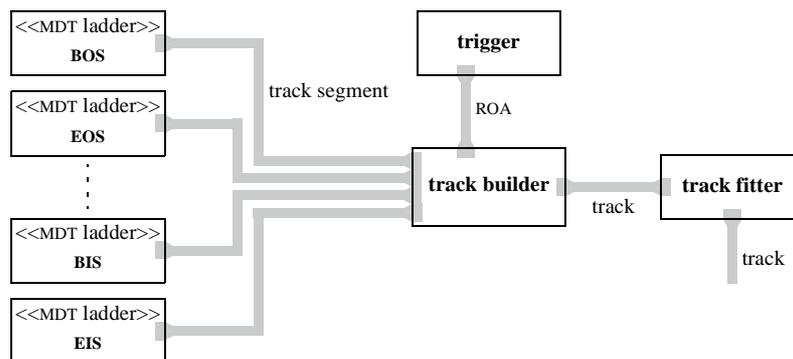


Figure 4.16 Global reconstruction algorithm.

7. A cylinder (BIS, BIL, etc.) in the barrel and a wheel (EIS, EIL, etc.) in the endcaps (see the glossary in appendix C).

matching criteria, but this is not necessary for the reconstruction based on the actual trigger roads as for them the segments fit together by design.

Next, the trigger clusters as stored in the ROA are added to the track and a fit is performed. This procedure is repeated for every track inside the current region of activity, and for all ROAs reconstructed by the trigger.

4.4 The Global Fit

The global fit of the precision and trigger hits is based on an iterative algorithm in which least-squared corrections are applied to the track parameters via the first derivatives of the residuals of each track constituent⁸ [57]. For m independent measurements and n track parameters p , the track residuals vector r and derivative matrix D are defined as

$$r \equiv \begin{bmatrix} r_1 \\ \vdots \\ r_m \end{bmatrix}, \quad D \equiv \begin{bmatrix} \frac{\partial r_1}{\partial p_1} & \cdots & \frac{\partial r_1}{\partial p_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial p_1} & \cdots & \frac{\partial r_m}{\partial p_n} \end{bmatrix} \quad (4.12)$$

The change to the track parameters is then given by

$$\delta p = (D^T \cdot D)^{-1} \cdot (D^T \cdot r) \quad (4.13)$$

with the covariance matrix equal to

$$C = (D^T \cdot D)^{-1} \quad (4.14)$$

The five independent parameters for the reconstruction of tracks in the ATLAS muon spectrometer are the $R\phi$ and z positions, the ϕ and θ angles, and the inverse of the transverse momentum $1/p_T$, all at some fixed radius R .

Since the inhomogeneity of the magnetic field prevents the analytic calculation of the residual and derivative matrices, the tracks must be propagated through the magnetic field to the position of each individual track constituent. They are therefore sorted in increasing distance along the track. From the position of closest approach of the track to a constituent, the residual can be calculated directly, while the derivatives can be retrieved from the transport matrix as it was created by the magnetic field propagation (cf. section 3.3.2).

The calculation of these positions of closest approach is performed by a list of so-called fit modules created out of the constituents of a track (see figure 4.17). In the case of the MDT hits, the tracks are propagated to the wire plane of the layer to which the hit belongs. Then a

8. In addition to a hit, a constituent can also be a vertex, a multiple scattering point, etc. (cf. section 3.3.1).

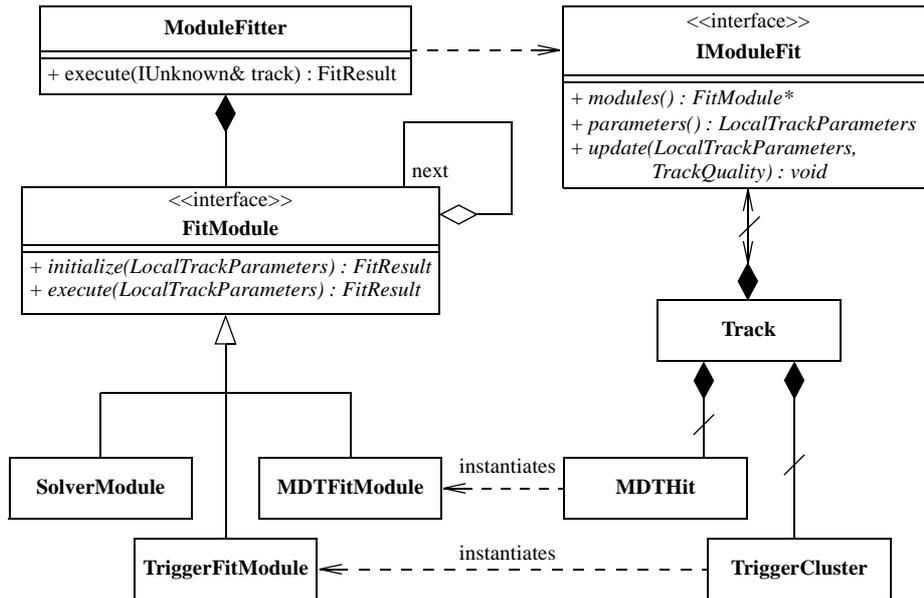


Figure 4.17 Classes involved in the iterative least-squares fit to a set of track constituents as defined by their fit modules.

straight-line approximation is used to determine the point of closest approach, after which the residual is determined based on the drift time of the hit and the track's coordinate along the wire. For the trigger clusters, the point of closest approach is determined in the plane defined by the two directions in which it has the largest extent, i.e. the directions of its member strips and/or wire groups. And as a cluster measures a track's position in two independent directions, it adds two rows to the residual and derivative matrices, one for each of these directions.

The resulting list of fit modules is controlled by the `ModuleFitter` class. It operates on a track through the `IModuleFit` COM interface (cf. section 3.3.1) and repeatedly executes the modules until the fit either converges or until a certain number of iterations has been performed. This convergence is determined by a `SolverModule` object, which is automatically added to the end of the module list. In its `execute` method it calculates the chi-squared, solves equation 4.13 and updates the track parameters and their covariance matrix.

As a starting point for the fit, the track's position and direction are copied from the track segment that was added last, which as a result of the definition of the `Track Builder` (see figure 4.16) is the innermost segment. Except for possible misalignments of the chambers, and the bending of very low energy tracks, these values accurately define the first four parameters of the global track.

The fifth parameter, i.e. the magnitude of the particle's momentum, can not be determined from that one track segment. Instead it is estimated based on the relative position and orientation of all segments, assuming a helical trajectory in a perfect and constant toroidal field. In the

barrel a magnetic field value equal to the average of the values at the centres of the track segments is taken. In the endcaps where the toroids lie between the stations, a constant field of 1 Tesla between the inner and middle stations is assumed. A different method would be to use a lookup table indexed on the track's η and ϕ coordinates, and using its sagitta. In any case, it turns out that the track fit is to a large extent independent of the accuracy in the initial guess of the momentum, and in most cases converges after 3 to 7 steps.

*Not everything that can be counted counts,
and not everything that counts can be counted.*

Albert Einstein

To test the pattern recognition algorithm, events are simulated in two stand-alone chambers, viz. a standard ATLAS inner layer chamber consisting of 2 multilayers with 4 tube layers each, and a typical middle or outer layer chamber with only 2×3 layers.

5.1 Simulation Environment

The simulation is executed within AMBER using the functionality provided by Arve: Muons with an energy of 100 GeV are generated and propagated through a 0.5 Tesla magnetic field with a direction that is parallel to the MDT wires. The origin and direction of the muons are varied so as to cover the whole chamber under angles ranging from -60° to $+60^\circ$ ¹.

The material description of the chambers includes:

- The cross-plates and long-beams taken from a typical chamber.
- The walls of the tubes (400 μm of aluminium).
- The gas, approximated as 100% argon.

Moreover, the electronics are simulated to have only a single-hit capability.

Then, in the conversion from the simulated drift distance to a drift time, the following operations are performed:

1. The drift distance is smeared according to a Gaussian distribution with a sigma that decreases linearly from 130 μm at the wire to 80 μm at a radius of 5 mm after which it remains constant. This means that the intrinsic single-tube resolution is equal to 90 μm ².

1. This covers the incident angles of all but the very low-energy tracks with the chambers in the ATLAS detector.

2. The average resolution is given by $\sqrt{\int \sigma^2(r) dr / 15}$, with $\sigma(r)$ the local resolution.

2. The distance is converted to a drift time with the help of a linear r-t relation based on a drift velocity of $30 \mu\text{m/ns}$.
3. A constant Lorentz angle of 0.2 radians is taken to increase the drift time. Furthermore an uncertainty in the magnetic field value of 5 mT is assumed, leading to an additional error in the drift time with a mean value of 0.25 ns [47].
4. The propagation time of the signal along the wire at a velocity of 0.7 times the speed of light is added to the drift time.

Effects that are ignored in the simulation are the time-of-flight correction and the possible misalignment of the wires. The former is impossible to correct for in the reconstruction as the origin of the simulated particle varies event by event. Besides, the error it introduces is negligible compared to the tube resolution. This is also true for the second effect, the wire misalignment, which introduces errors of $20 \mu\text{m}$ r.m.s. [15]. Moreover, as the resolution of a drift chamber is generally obtained from reconstructing tracks based on many different tubes, the wire displacements have already been folded into the single-tube resolution.

On top of this default behaviour of the chambers, two independent phenomena can be simulated. The first is the introduction of detector inefficiencies. From testbeam results, the single-tube efficiency has been determined to be over 99% [48]. However, during the long period of running of the ATLAS detector, tubes can cease to function. Therefore inefficiencies of up to 10%, which can be interpreted as 1 out of 10 randomly distributed tubes having gone dead, are investigated.

The second effect that can be simulated is the presence of background-induced hits. The nominal levels inside the ATLAS detector give rise to chamber occupancies varying between 0.4 and 2.2% for the inner chambers, and between 0.6 and 1% for the middle and outer ones [15]. This includes random hits that have a drift distance uniformly distributed between zero and the inner radius of the tube at the time the simulated track crosses the chamber, punch-through from the calorimeter and background-induced soft charged particles that intersect with

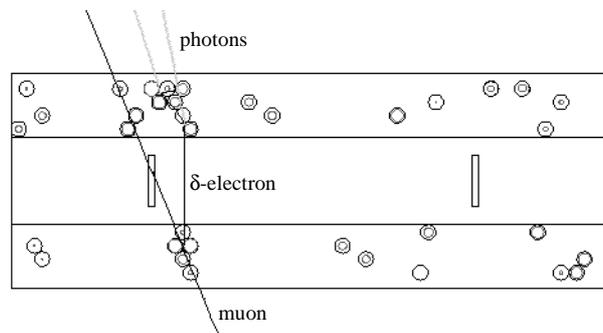


Figure 5.1 Simulated event in a 90%-efficient inner-layer (2×4) chamber with a background level that is 5 times nominal.

a few tubes³. In the studies presented in this chapter, the largest of these levels are taken as the nominal values. Because of the large uncertainties in the presented numbers, the pattern recognition is tested to rates of 5 times nominal.

5.2 Reconstruction Algorithm

To guide the reconstruction, a region of activity with a width of 3 by 3 cm around the simulated track is created based on the Monte Carlo information. This corresponds approximately to a trigger road based on two single-strip trigger clusters and serves a twofold purpose. First and foremost, it is used as an indication of the second coordinate. In the ATLAS detector, this information is retrieved from the trigger chambers, but in this simple simulation they are not included. Secondly, it greatly diminishes the false creation of fake tracks, as there is no interaction point that can be used as a reference point to which a track must point.

Another side effect of simulating only one chamber is that it is impossible to reconstruct the momentum of the tracks. For the reconstruction of the full ATLAS muon spectrometer, the drift-circle fit as described in the previous chapter only serves to select the best pattern of hits and to give an estimate of the track parameters; not to accurately derive them. This is in fact impossible for a straight-line fit as the tracks are curved in the magnetic field: Over the height of a chamber sagittas of around 50 μm for 100 GeV tracks, and close to a millimetre for 5 GeV ones are the result.

The only way that this problem can be solved is to have an estimate of the momentum, which can then be used to rotate the hits in the top multilayer relative to the bottom one. This estimate can be obtained from a global fit to all chambers, or from performing the drift-circle fit for different momenta and selecting the best one. However, in order to determine the intrinsic capabilities of the chambers and of the chamber reconstruction, the momentum estimate is set equal to the simulated value of 100 GeV.

5.3 Reconstruction Efficiency

The performance of the reconstruction in terms of efficiency and fake-track rate depends a priori on the internal parameters and cuts used by the program, as well as on the definition of what constitutes a good track, and what not. To start with the former, the requirements on the pattern recognition are simple: At least 7 hits per track for the 2 \times 4 chambers and a minimum of 5 hits for the 2 \times 3 chambers are needed. In addition, a track must have at least one hit in each multilayer. All tracks that pass these cuts are collected. If in a certain event there are no such tracks, the best one is kept anyway. In those cases the quality of a track is determined based on the chi-squared of the fit and the hit topology, which is a combination of the number of hits and the number of holes on the track.

3. This does not include the δ -rays which are automatically generated by Arve.

Then in the analysis, reconstructed tracks are classified based on the Monte Carlo information. A track is deemed “good” when the difference between its reconstructed and true angle is less than 1 mrad, the difference in offset is less than 100 μm , and the hit quality is larger than 75%. This latter quantity is defined as the fraction of hits on the track that correspond to the hits generated by the muon, including the correct assignment of the hit’s side. When the track fails any of these cuts, it is designated as a fake.

The resulting reconstruction efficiency for the 2×4 (BIL) chambers is plotted in figure 5.2. In the ideal scenario of no background and no detector inefficiencies, the reconstruction efficiency is found to be 99.8%. The remaining 0.2% is lost partly because of the misassignment of the side of the hits with very small drift distances, and partly because of the creation of δ -rays. The single-tube resolution at the levels used here has no significant effect on the efficiency, and neither does the Lorentz effect.

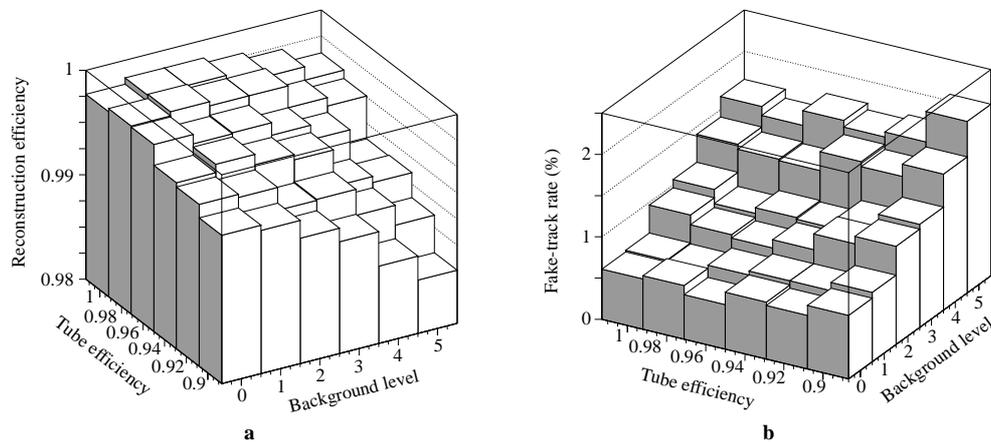


Figure 5.2 Reconstruction efficiency (a) and fake-track rate (b) of a 2×4 chamber as a function of the single-tube efficiency and the background level. The latter is quoted as a multiplicative factor applied to the nominal occupancy rate.

In virtually all these cases the real muon track was reconstructed but failed the cuts listed above. When not just the best track but all tracks above a certain quality threshold are kept, the efficiency can be boosted to a virtual 100% (1 out of the 10000 simulated tracks was not found). However, the number of fake tracks also increases dramatically, so this approach can only be applied when some additional criteria exist, as e.g. the existence of another track to which the segment must match.

As can be seen from figure 5.2, the reconstruction is very robust under deteriorations in either the single-tube efficiency or the background level. Only when these two factors conspire does the efficiency start to drop. At the same time the fake-track rate increases from 0.6% to 2.1%, but these numbers can be reduced when out of multiple tracks that share some common hits, only the best one is kept. Under nominal conditions this leads to a reduction of about 0.4%, while a 0.3% decrease is achieved in the worst-case scenario. Also, this procedure has no adverse effect on the reconstruction efficiency.

The behaviour of the 2×3 chambers, which are found in the middle and outer layers of the muon spectrometer is not much different (see figure 5.3). It is not surprising that because of the reduction in the number of tube layers, the overall reconstruction efficiency is lower than that of the 2×4 chambers, and that the fake-track rate is higher. The apparent vulnerability of the reconstruction to tube inefficiencies can also be attributed to it. In contrast, the effect of the (much lower) background is almost negligible. This behaviour is completely opposite to the situation in the inner-layer chambers.

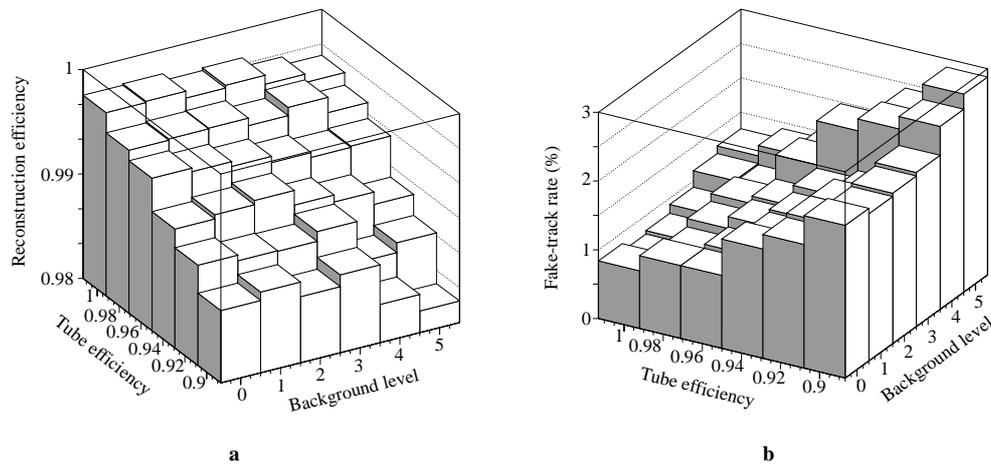


Figure 5.3 Reconstruction efficiency (a) and fake-track rate (b) of a 2×3 chamber as a function of the single-tube efficiency and the background level. The latter is quoted as a multiplicative factor applied to the nominal occupancy rate.

5.4 Fit Accuracy

The accuracy of the reconstruction is evaluated based on the fitted error in the two independent parameters of the straight track segment, viz. the angle and the offset. It turns out that these errors are fairly independent of the detector efficiency and background levels. Their slight increase is mostly related to the decrease in average number of hits per track as the external conditions deteriorate. In virtually all cases, fake hits are successfully removed from the track before the final track parameters are determined.

In the case of the 2×4 inner layer chambers, the mean error in the reconstructed angle rises from 0.21 mrad under nominal conditions (see figure 5.4) to 0.23 mrad in the worst case scenario. Similarly, the error in the offset rises from 30.4 to 33.0 μm . And in all cases the pulls of the distributions are in perfect agreement with unity.

In the case of the 2×3 chambers, the fewer number of hits per track lead to larger errors in the offset parameter. Depending on the efficiency and background levels, they vary between 35.2 and 37.5 μm . On the other hand, the smaller distance between the innermost and outermost hits compared to the inner layer chambers results in a smaller error in the track angle: It lies between 0.17 to 0.19 mrad.

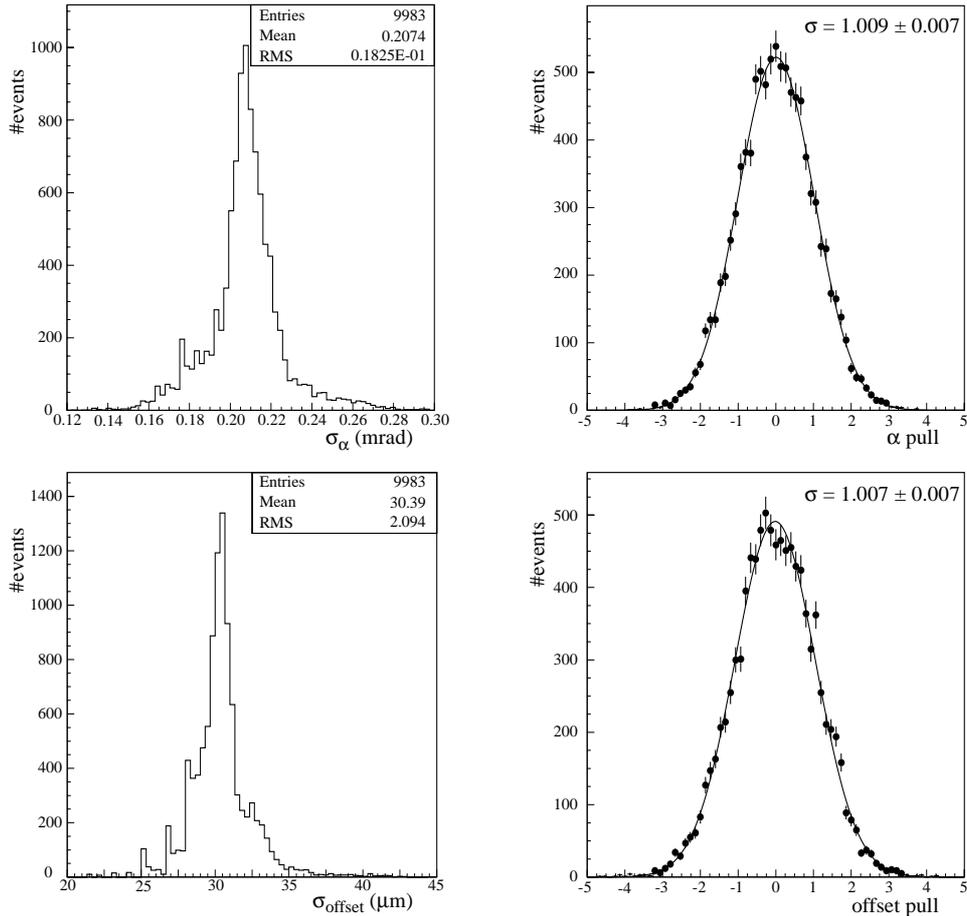


Figure 5.4 Accuracy in the angle and offset parameters of the fitted tracks in a 2x4 inner layer chamber under nominal conditions.

5.5 Single-Tube Resolution

The residuals of the reconstructed hits in both types of chambers under nominal conditions are shown in figure 5.5. These of course underestimate the single-tube resolution as a result of the bias introduced by the track fit. The simplest procedure to convert the residuals into a real resolution is to rescale them on a per track basis by a factor of $\sqrt{N/(N-2)}$ with N the number of hits on the track. For the two types of chambers this leads to the same resolution estimate of $88 \mu\text{m}$. Compared with the input resolution of $90 \mu\text{m}$, which is a combination of the intrinsic resolution of the tubes and the uncertainty in the Lorentz shift, this is a discrepancy of almost 2.3%.

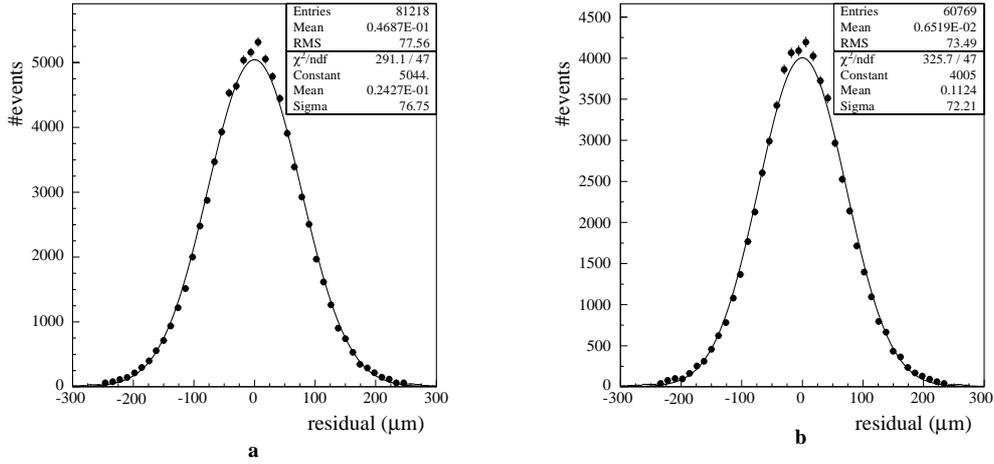


Figure 5.5 Hit residuals of the reconstructed tracks in a 2×4 (a) and a 2×3 (b) chamber under nominal conditions.

A method that is superior to the one above is to remove one hit at a time from the track, repeat the fit and use the residual of the removed hit as an estimator of the real resolution. The resulting distribution is shown in figure 5.6a. Its value of $102 \mu\text{m}$ in turn overestimates the resolution because of the finite precision of the track fit. If the fit's error at the position of the removed hit as shown in figure 5.6b is taken into account, the resolution is found to be $90 \mu\text{m}$, identical to the input resolution.

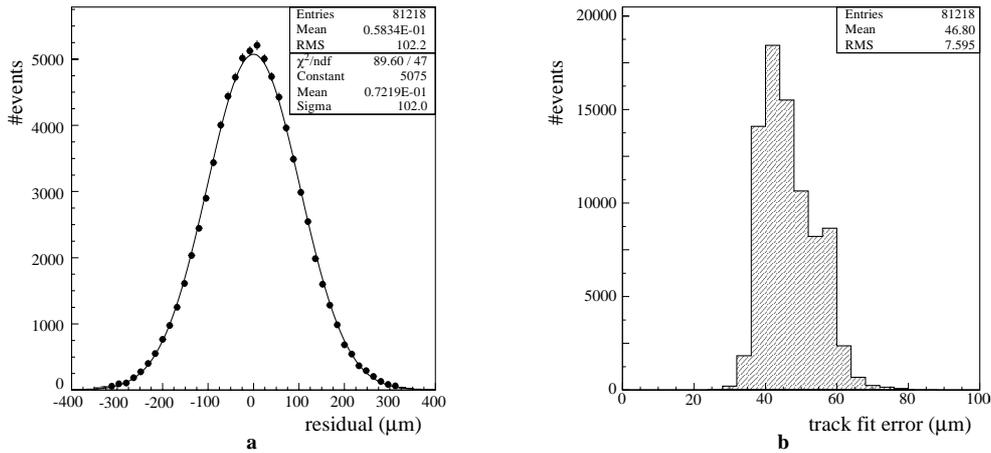


Figure 5.6 Residuals of the hits that are removed from a track (a) and the contribution from the finite fit precision to this quantity (b) for a 2×4 chamber.

┌

┐

└

┘

CHAPTER 6 | DATCHA

Never let reality get in the way of a good idea.

Chris Wallace

6.1 Introduction

DATCHA or Demonstration of ATLAS Chamber Alignment is an experimental setup designed to test the track detection and alignment capabilities of the ATLAS muon spectrometer [49]. It consists of three MDT chambers corresponding to a full-size barrel tower, augmented by three layers of RPCs (see figure 6.1).

The MDT chambers used in DATCHA consist of 2x3 layers in the case of the BIL and BML, and of 2x4 layers for the BOL, with the tubes extending in the x-direction (see figure 6.2). They are similar to the chambers that eventually will be used in the ATLAS detector, but due to their prototype nature they do have some imperfections. To begin with, the gas leak rate is two (BIL and BML) to three (BOL) orders of magnitude higher than the design single-tube rate of 10^{-8} bar.l.s⁻¹. This is in part caused by leaks in the gas distribution manifold. In addition, the BML

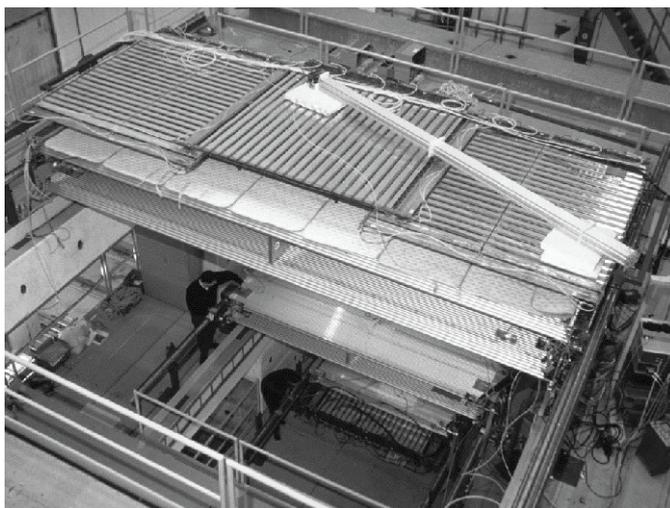


Figure 6.1 Photo of the DATCHA setup at CERN.

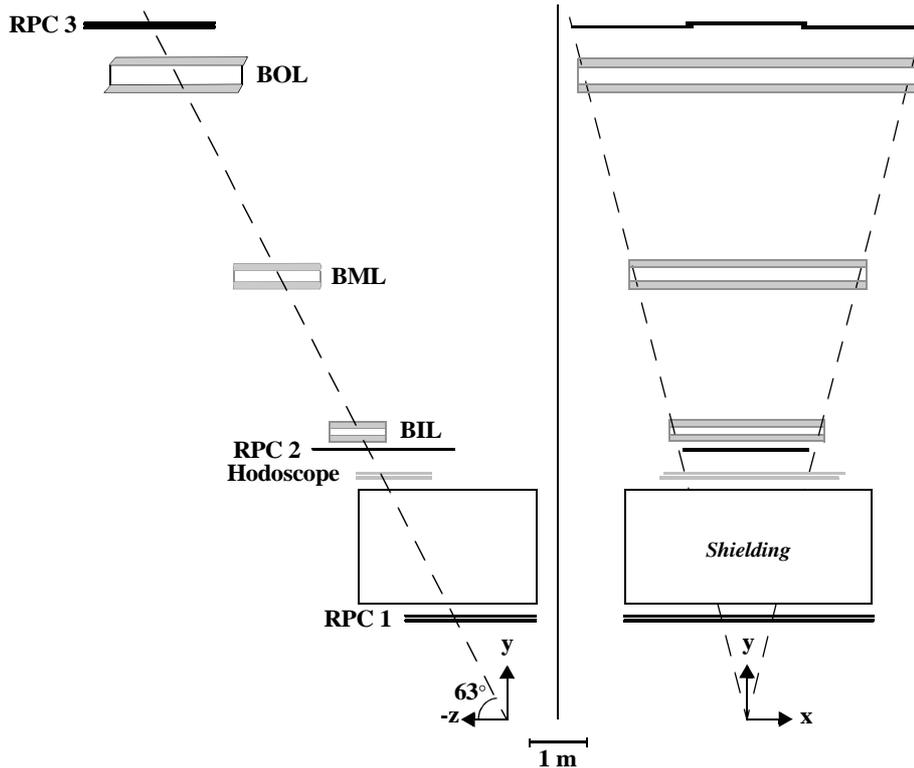


Figure 6.2 Schematic overview of the DATCHA chambers.

and BOL chambers also suffer from leaks in the gas connectors and cracks in the potting used to electrically insulate the passive components in the endplugs. As a consequence, the standard ATLAS gas $\text{Ar}/\text{N}_2/\text{CH}_4 - 91:4:5$ had to be replaced by a mixture of Argon and CO_2 in the ratio 80/20. Operated at 2 bar absolute pressure with a high voltage of 3150 V, its 1300 ns maximum drift time is much higher than the 500 ns of the standard ATLAS gas. Also, this choice of gas mixture leads to a highly non-linear r - t relation (see section 6.2.3).

A second shortcoming is that a significant fraction of the tubes in the BML and BOL chambers exhibit small discharges. However, this has for the most part been solved by adding a small amount of water (about 2500 ppm) to the gas.

The front-end electronics of a MDT chamber consist of a hedgehog preamplifier board with 32 channels serving eight tubes in each of a maximum of four tube layers. In addition they include a thick copper-clad ground plate to minimise electromagnetic interference, as well as a discriminator/multiplexer board with five outputs. Four of these are TDC outputs, one for each tube layer. This means that a TDC provides the logical OR of eight adjacent tubes within a layer. In addition, it is only capable of time stamping a maximum of eight leading and trailing edges. A typical TDC spectrum is shown in figure 6.3.

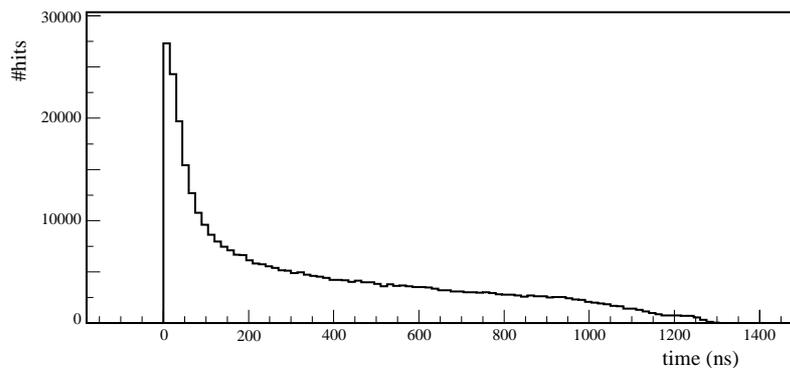


Figure 6.3 TDC spectrum of the BIL chamber after t_0 calibration (run 2015; see section 6.1.1).

The fifth output provides the correspondence between the TDC hits and the channel numbers of the tubes in which the hits were generated, but it can only keep track of a maximum of four addresses per TDC output, i.e. per group of eight tubes.

On the other end of the tubes the high voltage is provided by Cockcroft-Walton generators, one for each multilayer. They exhibit a long term stability of about 2 V at the chamber's end. The observed leak currents in the BIL chamber are about $1 \mu\text{A}$, whereas in the other two chambers several groups of eight tubes had to be disconnected to keep the leakage below $25 \mu\text{A}$ per multilayer (see also section 6.3.1).

The MDT alignment information is provided by several RASNIK alignment systems, which operate by creating an image of a coded checkerboard mask using an infrared LED, and projecting that image onto a CCD sensor with the help of a lens [50]. In this way they are capable of measuring relative displacements perpendicular to the optical axis with an accuracy of $1 \mu\text{m}$. Each MDT chamber is equipped with an in-plane system for monitoring possible chamber deformations. In addition, the corners of the three chambers are interconnected by projective alignment systems that record relative chamber displacements and rotations. About every ten minutes all 3×4 in-plane and four projective RASNIKS are read out, their images analysed and the results stored for offline analysis.

The DATCHA RPC chambers are also similar to the trigger chambers that will be used in the ATLAS spectrometer, all be it with a much simpler layout. The two uppermost chambers, RPCs 2 and 3, contain only one layer of strips that measure the second coordinate¹. Only RPC1 measures both coordinates as it has one layer of strips for each projection. Together with the scintillator hodoscope they are responsible for triggering on the cosmic muons. The hodoscope, which is positioned just below the inner MDT chamber, creates the primary signal with an overall timing resolution of 1 ns. A hit in the topmost RPC chamber is then needed to increase the chances that it was an actual muon that generated the hit in the hodoscope and that it has

1. The second coordinate of a hit is its coordinate along the MDT wire.

traversed the whole DATCHA setup. In addition, a hit in RPC1, which lies underneath the shielding enforces a lower limit of around 3 GeV on the energy of the cosmic rays. All in all, this leads to a trigger rate of about 5 Hz.

6.1.1 Data Runs

The results presented in this chapter are based on a number of data runs, all taken in December 1997. For our purposes here, run 2015 serves as the reference run, while the others were taken after the BML chamber had been shifted in the y- or z-direction. Listed in table 6.1 are the number of events in each run, as well as the number of “good” muon events. This classification is based on the reconstruction of the trigger hits using a simplified version of the GDL network described in chapter 4. In addition to the trigger definition given above, the following three requirements must be met:

1. The second and third RPC chambers must each contain only one cluster of hits, which are then used to build a trigger road. Hits in the RPC1 chamber are not used in this construction, because a cosmic-ray track can deviate significantly from a straight line due to the multiple scattering in the shielding. As a result of this requirement between 30 and 35% of the events are discarded.
2. The RPC trigger road must match to a cluster in the hodoscope, which also reduces the original sample by 30 to 35%.
3. No other clusters are allowed in the hodoscope to ensure an unambiguous determination of the trigger time. This results in a 60% rejection of the original events.

| Run | # events | # muon events | Comment |
|------|----------|---------------|--------------------------------------|
| 2011 | 299,866 | 118,346 | $\Delta Y_{\text{BML}} \sim -1.0$ mm |
| 2014 | 299,891 | 117,049 | $\Delta Y_{\text{BML}} \sim -2.0$ mm |
| 2015 | 299,879 | 115,408 | Reference run |
| 2016 | 299,875 | 114,937 | $\Delta Z_{\text{BML}} \sim 2.0$ mm |
| 2018 | 299,865 | 120,650 | $\Delta Z_{\text{BML}} \sim -2.0$ mm |

Table 6.1 Summary of the DATCHA runs, which are used in this chapter.

From the quoted numbers alone it follows that these requirements can not be independent. And in fact, there is an almost 90% correlation between requirements 1 and 2. Furthermore, it turns out that the sample remaining after requirement 3 is nearly a complete subset of the requirement 2 sample. Hence the overall trigger efficiency of 38 to 40% is only slightly lower than that of requirement 3.

6.1.2 Event Simulation

To complement the real data, simulation runs are performed using the internal simulation facility of Arve. The simulation of the MDTs is the same as described in the previous chapter, with the signal propagation velocity, r-t relations and residuals taken from the analysis of run 2015 (see section 6.2). For both the RPC chambers and the hodoscope² an approximation of the support structure designed to produce the appropriate amount of multiple scattering has been implemented. Furthermore, the iron (1.6 m) and concrete (0.8 m) shielding are added to the detector description to correctly form the trigger decisions.

As a particle source, a cosmic-ray generator is used. It creates muons and anti-muons in a ratio of 4 to 5 and with their origin uniformly distributed in a plane above the detector, while their direction has an angular distribution of $\cos^2(\theta)$, with θ the angle between the muon and the vertical y-axis. They are given a momentum in the range of 3 to 100 GeV/c with an underlying p^{-2} distribution [51]. The minimum of this range is based on the shielding present in the detector, the maximum on the electromagnetic interaction tables available for the materials.

6.2 Calibration

Precise knowledge of the detector's behaviour is needed to correctly interpret the TDC times that come out of the data acquisition system. Various corrections must be applied before these times can be converted into drift distances, which can then be used to reconstruct the tracks. These calibration aspects include

- The time-of-flight of the muon and the response of the hodoscope;
- The relative timing between the MDT channels in the form of the t_0 (leading edge) and t_{\max} (trailing edge) values of the TDCs;
- The r-t relation of the MDT gas mixture;
- The velocity with which the signal propagates along the wire;
- The relative positions of the wires;
- The chamber deformations and displacements, as well as the gravitational sag of the wires.

The first four of these corrections are explored in detail in the paragraphs that follow. The fifth effect, that of the individual wire offsets, is ignored and the design values are used in the reconstruction. To compensate for this, in the determination of the resolution of the drift tubes, an uncertainty of 20 μm r.m.s. is assumed [15].

This then leaves the last item. The chamber displacements have no effect on the MDT track segments, but only on their global matching. In contrast, the chamber deformations and

2. The hodoscope is defined as a RPC chamber with strips the size of the scintillator tubes.

gravitational sag of the wires do effect both parts of the reconstruction. However, as they vary only moderately and continuously over the extent of the chamber, in the small region in which a track crosses that chamber they can be assumed to be constant. As a result they have no real effect on the pattern recognition, but only influence the final track segment parameters, and thereby the global matching. All these global alignment effects are folded into the sagitta measurement as described in section 6.3.4, and a more comprehensive study of the alignment systems in DATCHA can be found elsewhere [54-56].

6.2.1 Time-of-flight Correction

The first effect that has to be considered is the time it takes the cosmic muon to fly from the MDT tube in which it generates a hit to the hodoscope, which determines the trigger time. Since the muon travels at a speed close to the speed of light³ along a nearly straight path, this time-of-flight correction is only dependent on the vertical position of the hit and the direction of the particle. In principle, an initial segment fit is needed to determine these parameters. However, as it is favourable to being able to correct each individual hit before the segment reconstruction is started than to have to perform an iterative procedure, the parameters are estimated instead.

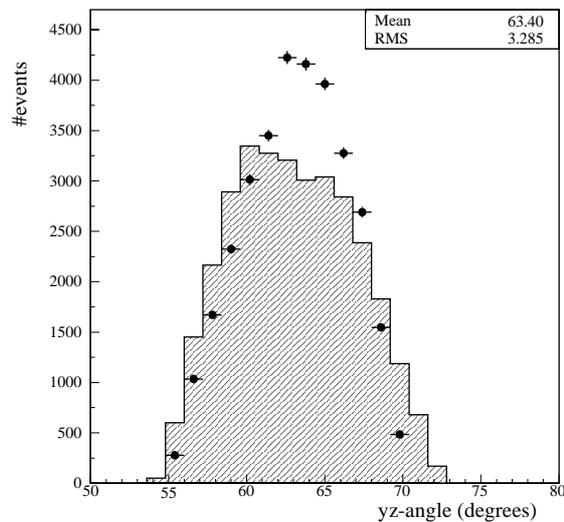


Figure 6.4 Reconstructed track angle in the yz-projection. The points represent the data from run 2015, while the shaded histogram is the result of a Monte-Carlo simulation. The reason that the Monte Carlo data has a wider distribution than the real data lies in the fact that the wall of the pit in which the detector resides is not included in the simulation.

3. At their lowest triggering energy of 3 GeV, a muon travels already at a velocity of 99.9% of the speed of light. Hence even for the BOL, the maximum effect on the time-of-flight is 0.03 ns, which can safely be ignored.

For the hit's vertical position, the location of the wire can be taken. The particle's direction is however slightly more difficult. In the xy -plane (cf. figure 6.2), it can be derived from the road created from the RPC hits. In the yz -projection no such information is available, but here the geometry of the DATCHA tower helps out: All tracks that pass the trigger requirements must have had an angle of $63 \pm 3^\circ$ (see figure 6.4). For the time-of-flight correction, this spread is ignored and the average value is taken for all hits. The error that this approximation introduces can be calculated from the yz -length s of a track with an angle α (i.e. the real path of the muon), compared to that of a track with the average DATCHA angle of 63° (i.e. the path assumed in the time-of-flight correction):

$$s = \frac{s_0 \cdot \tan \alpha}{\sin(\Delta\alpha) + \cos(\Delta\alpha) \cdot \tan \alpha} \quad (6.1)$$

with s_0 the track length at 63° , and $\Delta\alpha \equiv \alpha - 63^\circ \geq 0$ the difference in angle between the two tracks. With a maximum angle of 70° (cf. figure 6.4) the upper limits of the time-of-flight errors are equal to 0.1, 0.5 and 0.9 ns for the BIL, BML and BOL chamber respectively⁴. For angles that are smaller than 63° equation 6.1 must be inverted, leading to lower limits of respectively -0.2, -1.1 and -2.3 ns, which correspond to a minimum track angle of 55° .

This difference between the lower and upper limits explains the shapes of figure 6.5 in which the deviation between the reconstructed and real time-of-flight correction for 10,000 simulated events is plotted. It also shows that the errors are dominated by the approximation of the track angle in the yz -projection. Including all effects, the errors in the time-of-flight estimate are equal to 0.1, 0.4 and 0.9 ns for respectively the BIL, BML and BOL chamber.

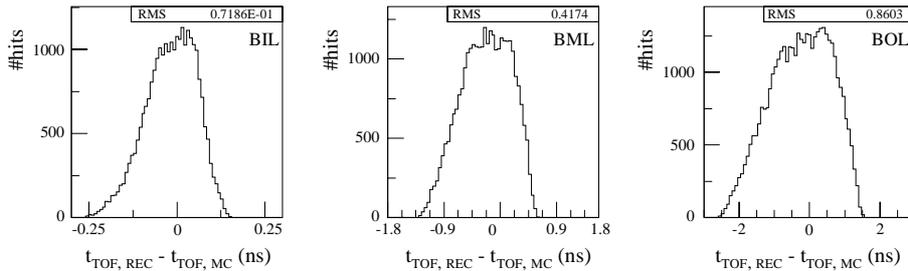


Figure 6.5 Difference between the reconstructed and Monte Carlo time-of-flight corrections.

6.2.2 Leading and Trailing Edges

The determination of the leading and trailing edges of the TDC spectrum of each individual tube is needed to factor out the behaviour of the front-end electronics, as well as the differences in length of the cables connecting the tubes to the TDCs. As its input, the procedure requires the

4. The fact that the maximum angle in the Monte Carlo data extends to 73° has no significant effect on the magnitude of the time-of-flight errors.

drift times corrected for the time-of-flight of the muon and for the propagation of the signal along the wire. However, the propagation velocity can only be determined after the r-t relations, and hence the leading edges, are known (see section 6.2.4). One solution would be to use only those hits that lie close to the front-end electronics, but that would lead to very poor statistics. So instead a propagation velocity equal to the speed of light is assumed, which introduces a systematic shift in the t_0 and t_{\max} values that is approximately identical to

$$\Delta t \approx \frac{L}{2} \left(\frac{1}{v_s} - \frac{1}{c} \right) \quad (6.2)$$

i.e. the difference in signal propagation time of a hit halfway down the tube as a result of the difference between the real propagation velocity v_s and the assumed velocity of the speed of light. This is however only valid if this shift is small enough so that it does not influence the outcome of the pattern recognition, and thereby the r-t calibration procedure⁵. These r-t relations can then be used to determine the real signal propagation velocity, after which the leading and trailing edges can be corrected for it.

The actual procedure of determining the t_0 values is quite straightforward: The leading edge of each individual TDC spectrum is parameterized by

$$L(t) = \alpha_1 + \frac{\alpha_2}{1 + \exp\left(\frac{\alpha_3 - t}{\alpha_4}\right)} \cdot \frac{\alpha_5 + \left[1 + \exp\left(\frac{t - \alpha_3}{\alpha_6}\right)\right]^{-1}}{1 + \alpha_5} \quad (6.3)$$

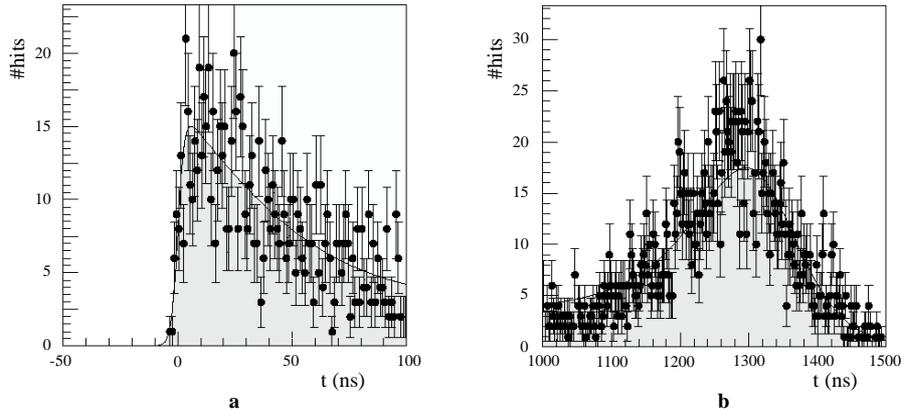


Figure 6.6 Example of a t_0 (a) and a t_{\max} (b) fit [52, 53]. The data is represented by the dots and the fit by the solid line.

5. For the BOL, which is the largest chamber, an actual signal velocity of 75% of the speed of light corresponds to an error of 3 ns. This is much smaller than the resolution of the individual tubes and therefore causes no problems for the pattern recognition.

in the interval $t \in -(50, 100)$ ns (see figure 6.6a), where only entries for hits that could be associated to a track segment are shown. In the first iteration all parameters are left free, while in the second iteration the slope parameters α_4 and α_6 as well as the plateau parameter α_5 are fixed to their average value for the layer to which the tube belongs. The value of t_0 is then defined as the time at which $L(t)$ reaches half of its maximum value. Its error depends therefore only on the slope parameter α_4 and on the event statistics, and is listed in table 6.2.

| Chamber | Δt_0 (ns) | Δt_{\max} (ns) |
|---------|-------------------|------------------------|
| BIL | 0.5 | 2.4 |
| BML | 0.6 | 3.1 |
| BOL | 0.7 | 4.6 |

Table 6.2 Errors in the leading and trailing edges.

A similar approach is used to determine the t_{\max} of each channel, i.e. the time of the last hit relative to the channel's t_0 (see figure 6.6b). These values are needed to rescale the drift times of a group of tubes to a single t_{\max} value so that a common r-t relation can be used. This procedure can only be used when the gas mixture and operating conditions of those tubes are similar, and when no out-of-centre positioned wires are present. However, these are also the requirements for a single r-t relation to be valid, so the success of the r-t calibration in the next section shows that the rescaling of the drift-time spectra is a legitimate procedure.

6.2.3 R-T Calibration

The determination of the r-t relations is performed with the help of an auto-calibration procedure in which an initial set of relations is used to reconstruct the events. Then, based on the reconstructed track segments, the relations are recalculated by taking the fitted drift distances instead of the computed ones. The mean values of Gaussian fits to the drift times in each drift-distance bin are plotted as a function of this distance as shown in figure 6.8. The binning in the drift distance is preferred, because it leads to similar statistics in each bin. The reason for this is that because of their cosmic-ray nature the muons illuminate the tubes uniformly in radius (see also section 6.3), while the non-linearity of the r-t relations causes this uniformity to be lost in the drift times. This procedure is then repeated several times until the r-t relations are stable.

To reduce the effects of random noise and δ -ray hits, only “good” track segments are selected as defined by:

- A segment must have at least 5 hits. In spite of the fact that the BOL consists of two more layers than the other two chambers, its many disconnected tubes prevent the application of a stricter cut;
- Of these hits, at least two must come from each multilayer;

- The chi-squared per degree of freedom of the segment fit must be equal to or less than 5.

These requirements alone are however not sufficient to always reconstruct the true track as can be seen from figure 6.7: When the hits all lie on the same side of the wires, the fit will shift the reconstructed track from the true one by any systematic error present in the r - t relations. Since the tracks used in the calibration are a mixture of type A and type B as defined in figure 6.7, this means that the calculated r - t relations lie somewhere in between the original and the true ones. Hence a larger number of iterations is needed in order for the calibration procedure to converge.

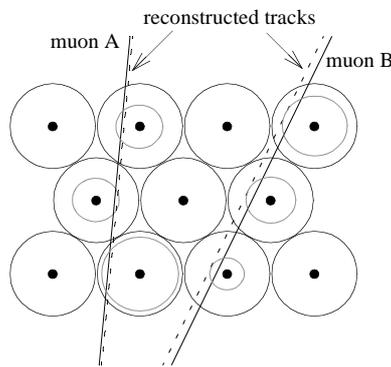


Figure 6.7 Sketch of a MDT multilayer in which the r - t relations overestimate the drift distance of the hits. Muon track A is reconstructed almost correctly despite the systematic error, whereas the reconstructed track B shows a large deviation from the true track.

To select only those tracks that cross tubes on both sides of the wires, the following criteria must also be met for each separate multilayer:

- The number of hits with the track passing the wire on the left side, and the number of right hits must both be larger than zero;
- The difference between these two types of hits must be either zero or one.

In the reconstruction of the track segments again a signal propagation velocity equal to the speed of light is assumed. This means that only for a hit in the centre of the MDT wire does the shift in the t_0 value cancel out against the drift time reconstruction error caused by the incorrect knowledge of this velocity. Using all hits independent of their second coordinate results in a spread in the drift times, but that does not effect the mean value in each drift-distance bin.

This leaves us with one unanswered question, viz. how many r - t relations are needed. From a theoretical standpoint it would be preferable to have a separate relation for each individual tube. However, this is neither practical nor precise due to the lack of statistics. So instead r - t relations are derived for each tube layer. Analysis has shown that the variations in

the r-t relations between different regions along the tubes' length are much smaller than the differences between tube layers or even between individual tubes [52].

A typical r-t relation is shown in figure 6.8, and the shapes of the 19 other relations are all very similar to it. Compared to the average of each chamber they differ by less than 5 ns in the case of the BIL, and by less than 10 ns for the other two chambers (see figure 6.9). To estimate the error in the r-t relations, three different comparisons have been performed:

- The difference in r-t relations between two iterations of the calibration procedure is about 0.5 ns independent of the chamber. This is however only a measure of the stability of the algorithm, and not of the correctness of the relations for each individual tube;
- The difference between a Monte Carlo input r-t relation and the reconstructed one is found to be in the order of 2 ns for the BIL chamber and 1 ns for the other two chambers. The reason for the larger error in the inner chamber is most probably its 2x3 tube layer layout in conjunction with its small multilayer separation. These numbers are an estimate of the correctness of the r-t calibration procedure, and do not include any tube-to-tube variations;
- From the determination of the r-t relation of each individual tube, it can be deduced that the tube-to-tube variations have an r.m.s. value of 1 ns in the case of the BIL chamber, and 2 ns for the other two chambers.

In total this means that an error of 2.3 ns in the r-t relations for all chambers can be assumed.

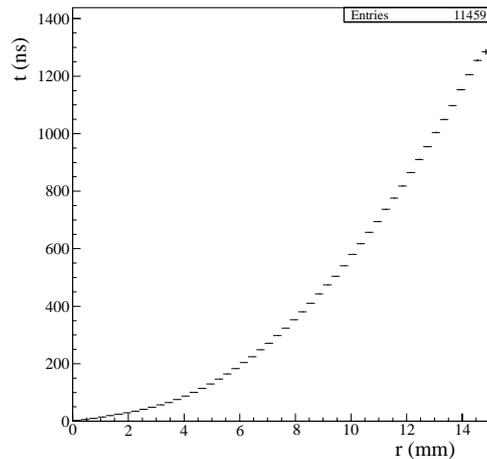


Figure 6.8 The r-t relation of the first layer, first multilayer of the BIL chamber (run 2015).

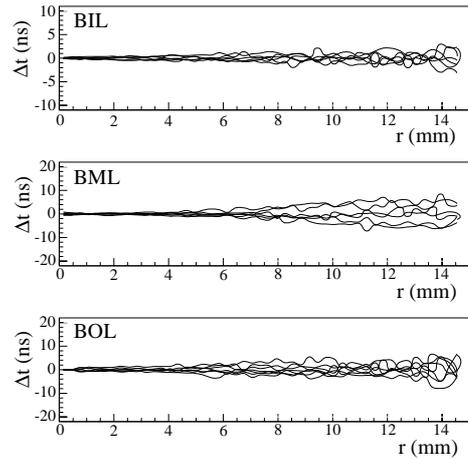


Figure 6.9 Variation in the r-t relations of the three chambers relative to their average value.

6.2.4 Signal Propagation Velocity

The time it takes the signal to propagate along the wire from the point the muon crosses the tube to the front-end electronics must be subtracted from the measured time to arrive at the actual drift time. Any deviation from the real propagation velocity in applying this correction shows up as a systematic increase or decrease in the radius of all drift circles; a shift that moreover depends linearly on the hit's second coordinate. This phenomenon can only be detected for track segments that have hits on both sides of the wires (cf. figure 6.7), which means that the same hit criteria as used in the determination of the r-t relations must be used.

The residuals of these hits, converted to drift times, can be plotted against their second coordinate as determined by the trigger roads. This is done in figure 6.10 for run 2015 with the propagation velocity set to the speed of light. Based on the slope of the fitted lines, the real velocity v_s can be determined according to

$$\frac{\Delta t_{\text{res}}}{\Delta x} = \frac{1}{v_s} - \frac{1}{c} \quad (6.4)$$

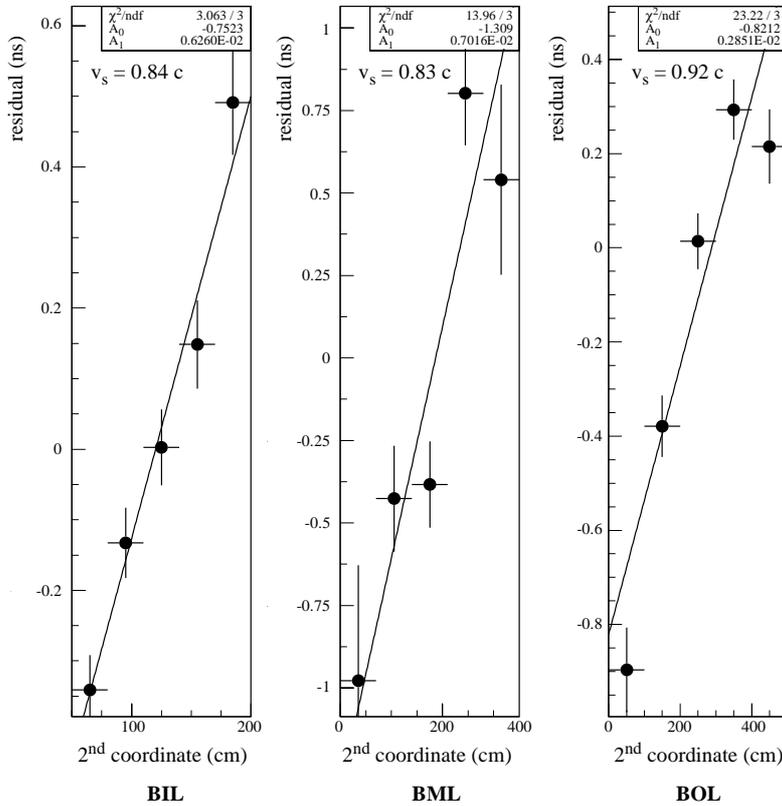


Figure 6.10 Signal propagation velocity in the MDT chambers. The slope parameter A_1 can be converted to a velocity using equation 6.4.

which results in velocities of respectively 0.84, 0.83 and 0.92 times the speed of light. Their errors can not be determined solely based on the errors of the fits, because the slopes do depend somewhat on the binning used in the histograms. From studying various binnings an error of around 5% in v_s can be deduced. This is in fair agreement with independent signal-speed measurements performed on dedicated twin tubes⁶, which have determined the velocity to be 3.8 ns/m or 0.88 times the speed of light.

Based on the length of the chambers, the average errors the signal propagation induces in the times measured in DATCHA are equal to 0.3, 0.4 and 0.5 ns for the BIL, BML and BOL chambers respectively. This includes the uncertainty due to the 3 cm (0.8 cm r.m.s.) width of the trigger roads.

The errors in the drift times caused by the various calibration procedures are listed in table 6.3. The wire-offsets error corresponds to the 20 μm r.m.s. uncertainty in the wire position. The effect of multiple scattering on the drift times has been determined from dedicated Monte Carlo runs in which the material was selectively turned on or off. The final conversion from the drift-time errors to the corresponding errors in the drift distance is based on an average drift velocity of 11.5 $\mu\text{m}/\text{ns}$ (cf. figure 6.8).

| Effect | BIL | BML | BOL |
|---------------------|------------------|------------------|------------------|
| Trigger (hodoscope) | 1.0 | | |
| Time-of-flight | 0.1 | 0.4 | 0.9 |
| t_0 calibration | 0.5 | 0.6 | 0.7 |
| R-T relation | 2.3 | | |
| Signal propagation | 0.3 | 0.4 | 0.5 |
| Wire offsets | 1.7 | | |
| Multiple scattering | 0.4 | 0.4 | 0.7 |
| Total | 3.1 ns | 3.2 ns | 3.3 ns |
| | 36 μm | 37 μm | 39 μm |

Table 6.3 Drift-time errors in ns induced by the various calibration procedures. The last line shows the total errors in the drift distance.

6. In a twin-tube setup the wires of pairs of MDT tubes are connected at the high-voltage side so that the signal of a hit in one of the tubes is read out in both of them.

6.3 Reconstruction

The reconstruction of DATCHA events is a three-step process. First the RPC and hodoscope hits are used to create trigger roads. This is followed by the independent reconstruction of track segments in each of the three MDT chambers. Finally, the segments are matched together to form the global muon track. The result for a typical event is shown in figure 6.11. As can be seen from the deviation between the BML segment and the global track, for the analysis described

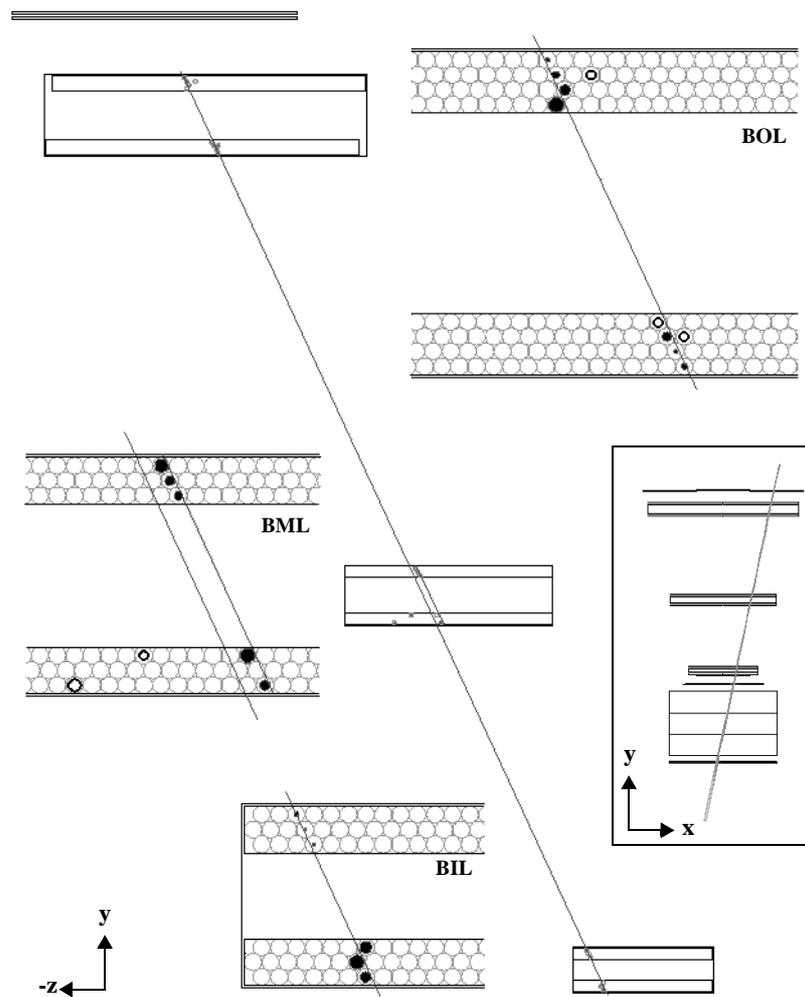


Figure 6.11 Display of one of the first reconstructed events, with the inset showing the trigger road in the xy-projection. The solid circles represent the hits assigned to the track segments, and the shift of the BML chamber is clearly visible.

in this section the results of the alignment systems, i.e. possible chamber deformations and displacements, are not used in the reconstruction.

6.3.1 Single-Tube Efficiency

The single-tube or hit efficiency for the BIL chamber is shown in figure 6.12. The result of a Monte Carlo simulation of 100%-efficient tubes up to their inner radius of 14.6 mm has been included for reference. The hit efficiency is defined as the fraction of tubes crossed by the track segments that contain a hit. The good-hit efficiency requires in addition to this that the hit in question has been assigned to one of the segments. The behaviour of the other two chambers is very similar to that of the BIL, all be it with different efficiencies.

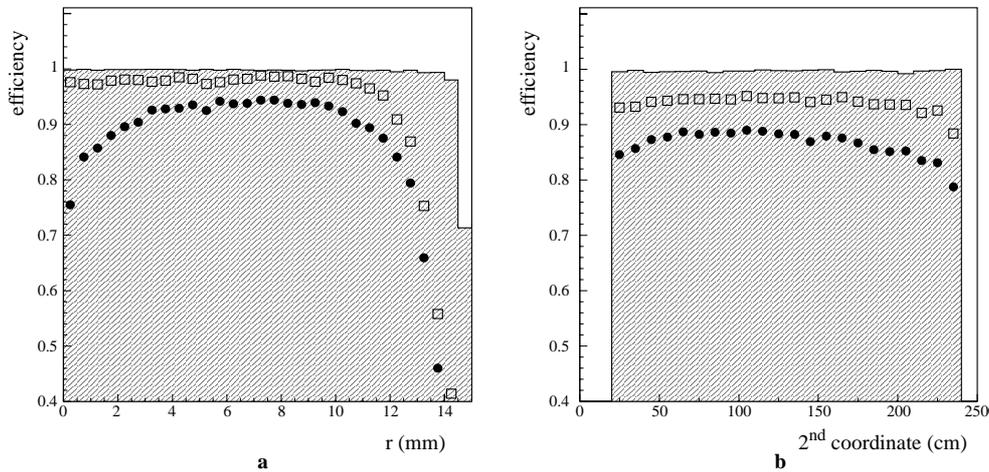


Figure 6.12 Hit efficiency (squares) and good-hit efficiency (circles) as a function of drift radius (a) and second coordinate (b) for the BIL chamber in run 2015. The shaded histograms show the results of a Monte Carlo simulation.

The highest hit efficiency is recorded in the BIL: Up to a drift radius of around 10 mm it is fairly constant at around 98%. The fact that it starts to drop for radii beyond that is explained in figure 6.13: A track with a certain angle α that generates a hit with a radius above a value given by

$$r_{\text{hit}} = p \sin \alpha - r \quad (6.5)$$

with r the tubes' radius and p their pitch, also crosses a neighbouring tube in the same layer. In seven out of eight times these two tubes belong to the same multiplexer, and hence only the address of one of the two tubes is retained. Due to the layout of the DATCHA setup (cf. figure 6.4) α falls in the range between 55° and 70° , which means that this effect starts to occur for hits with a radius of 10 mm ($r = 14.6$ mm and $p = 30.1$ mm) and reaches its maximum

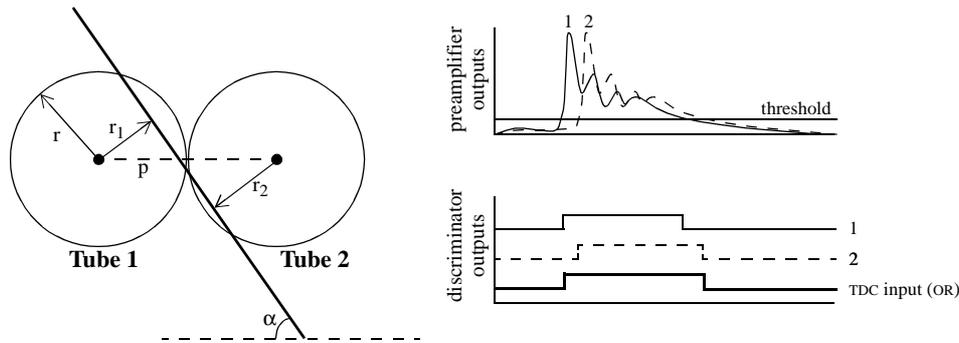


Figure 6.13 Explanation of the inefficiency caused by the multiplexed read-out at large drift radii. When a muon track generates hits in two neighbouring tubes, one of the hits is lost because the TDC takes the logical OR of the discriminator outputs.

inefficiency of 44% at a radius of 13.7 mm. Unlike this dependence on the drift distance, there is hardly any variation visible in the hit efficiency as a function of the second coordinate (see figure 6.12b).

The small inefficiency of the BIL chamber is in part explained by its one dead tube. The remaining 1.5% inefficiency could be the result of cross talk between the tubes, which as a result of the multiplexed read-out shows up as an inefficiency. This idea is confirmed by a dedicated run with every other tube disconnected, i.e. with the multiplexing scheme in principle disabled, in which this inefficiency is fully recovered.

The much higher inefficiencies observed in the BML and BOL chambers are almost entirely due to their disconnected tubes. In fact the fractions of disconnected tubes of around 34% and 59% respectively are much higher than the observed inefficiencies of 14% and 17%, but this is because the disconnected tubes cover entire regions of the chambers in which no tracks are found to start with.

Figure 6.12 also shows the good-hit efficiency. It is in general 5% to 10% lower than the standard hit efficiency due to δ -rays and incorrect results of the pattern recognition. The additional inefficiency observed at small drift distances is in part caused by the increased chance of reconstructing the wrong track segment because of an incorrect left-right assignment of the hits, and in part by the behaviour of the front-end electronics. When a track passes close to the anode wire, and therefore leaves a long ionization trail, it can generate many discriminator level crossings. Because the TDCs are only capable of storing the last eight leading and trailing edges, when there are more, the first and most important ones are lost. This means that a hit is still registered in that tube, but that its drift time is incorrectly measured. Hence the difference between the hit and the good-hit efficiencies.

When looking at the coordinate along the wire, the hit and good-hit efficiency follow each other nicely. The loss of efficiency near the ends of the tubes is a normal phenomenon in drift-tube operation.

6.3.2 Segment Reconstruction

The segment reconstruction follows the same path as described in the previous chapter, and uses the hit requirements listed in section 6.2.3. For the BIL the accuracy of the fit in the form of the errors in the angle and offset parameters of the reconstructed segments is shown in figure 6.14. These same quantities for all chambers are listed in table 6.4. They are slightly better than the results of the simulated data (see section 5.4), because of a higher single-tube resolution of the DATCHA chambers compared to what is expected of the future ATLAS chambers. This is most likely due to the much slower gas that is used here⁷.

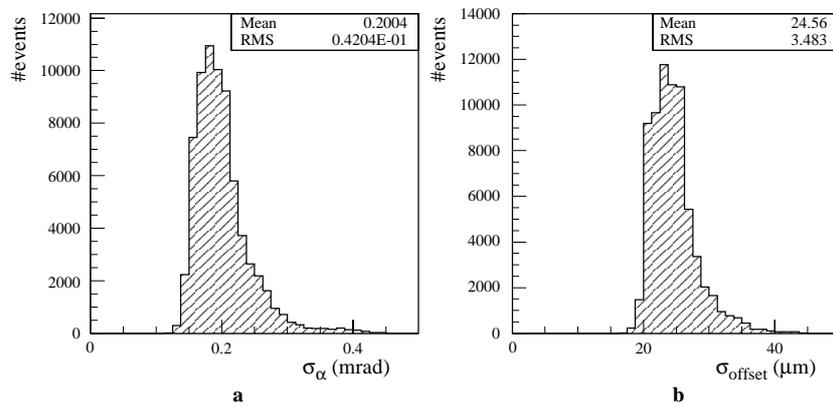


Figure 6.14 Angle (a) and offset (b) errors of the track fit in the BIL chamber.

| Chamber | Efficiency | σ_{α} (mrad) | σ_{offset} (μm) |
|---------|------------|--------------------------|--|
| BIL | 61% | 0.20 | 24.6 |
| BML | 42% | 0.17 | 28.1 |
| BOL | 42% | 0.15 | 35.0 |

Table 6.4 Results of the track-segment reconstruction in the three MDT chambers.

Table 6.4 also lists the efficiencies for reconstructing a track segment in the various chambers. In the case of the BML and BOL chambers they are a direct result of the single-tube efficiencies. On the other hand, for the BIL the inefficiency has a largely geometrical origin since the muons that cross the chamber at its edges fail to generate enough hits to pass the cuts.

The corresponding hit residuals are shown in figure 6.15. To convert them to single-tube resolutions, the same approach as described in section 5.5 is used, i.e. one hit at a time is removed from the track after which the fit is repeated. The residual of the removed hit, after

7. Based on the DATCHA results and those of ageing tests, the ATLAS MDT gas has recently been changed to an Argon-CO₂ mixture with a maximum drift time of 700 ns.

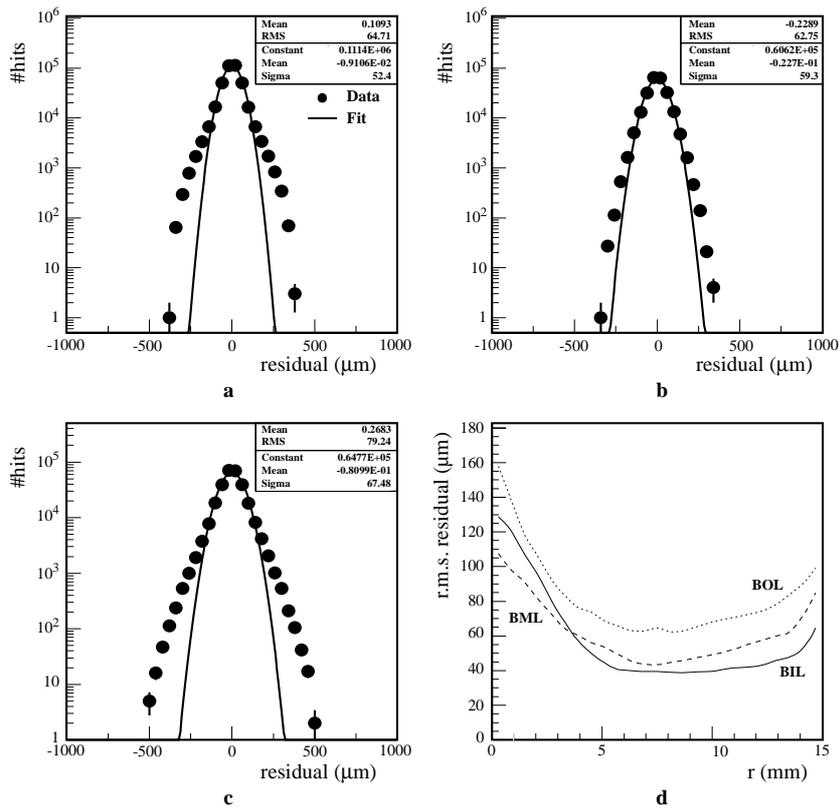


Figure 6.15 Residual distributions of the BIL (a), BML (b) and BOL (c) chambers, and as a function of the hit's drift distance (d).

it has been corrected for the finite precision of the track fit, is then used as an estimator of the resolution. After unfolding the errors of the various calibration procedures (see table 6.3), the resulting resolutions are 68, 82 and 86 μm for the BIL, BML and BOL chambers respectively.

6.3.3 Segment Matching

In the segment matching process, the information from the alignment systems is not used to correct for any chamber displacements or deformations. Therefore systematic shifts between the angles and positions of the segments from different chambers are to be expected, and the matching criteria are kept very loose to compensate for this.

As a first step, the segments in the BIL and BOL chambers are compared to form a global track (see figure 6.16). The difference in their position is determined by following the BIL segment to the BOL chamber, and reveals a shift in the relative positions of the chambers compared to the design values as used by the reconstruction. The widths of both distributions

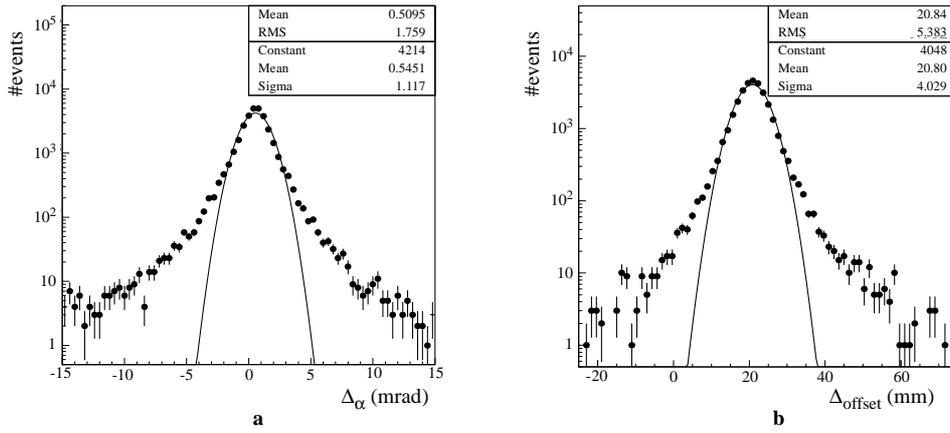


Figure 6.16 Angle (a) and offset (b) difference between the BIL and BOL track segments for run 2015.

are for the most part comparable to what can be expected from multiple scattering. The remainder is caused by misreconstructed track segments in either of the two chambers. Based on these figures, the cut on the difference in angle between the BIL and BOL segments is set at 10 mrad, while the cut on their offset difference is set to 30 mm around a mean value of 21 mm. This leads to an efficiency for finding a global track of around 28%, which means that the inefficiencies in the two chambers are almost completely uncorrelated.

The global track can then be compared to the segment found in the BML chamber. Because the BML chamber can be shifted in y and/or z , no cut on the offset difference is applied. Only a 10 mrad maximum on the difference in angle between the global track and the BML segment is enforced. The resulting efficiency then comes in at 16%.

6.3.4 Sagitta Measurement

The sagitta measured between the global track and the BML segment is converted to a horizontal, i.e. in the z -direction, shift of the BML chamber using the track's angle. Its distribution for run 2015 is shown in figure 6.17. The mean value is not consistent with zero as the chamber positions do not coincide with the design values used in the reconstruction. Its width is almost completely brought about by the multiple scattering in the BML chamber. This is not only clear from the fact that the r.m.s. sagitta of 1.8 mm far exceeds the accuracy of around $30\ \mu\text{m}$ in the reconstructed offset parameters of the track segments, but has also been confirmed by the results of a material-free Monte Carlo simulation run.

The same sagitta is also measured by the four projective RASNIK alignment systems, whose averaged value is equal to 1.005 ± 0.002 mm. But as the RASNIK systems have not been absolutely calibrated, this number can not be directly compared with the result of the cosmics reconstruction. To circumvent this, one must look at the change in sagitta between the various runs (cf. table 6.1) as displayed in figure 6.18. This procedure has the added advantage that the

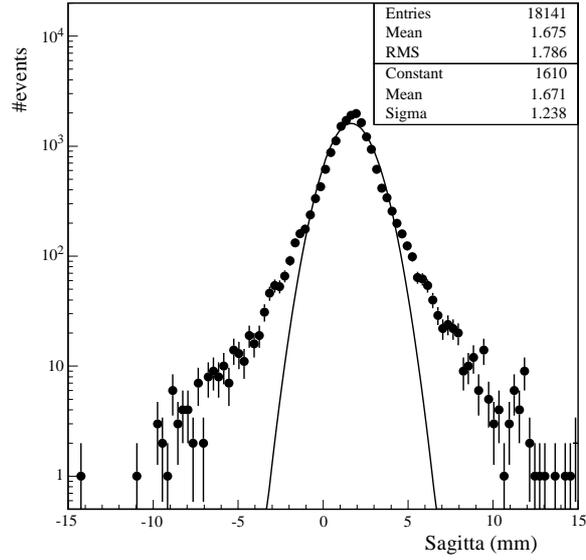


Figure 6.17 Distribution of the sagitta between the global tracks and the BML segments in run 2015.

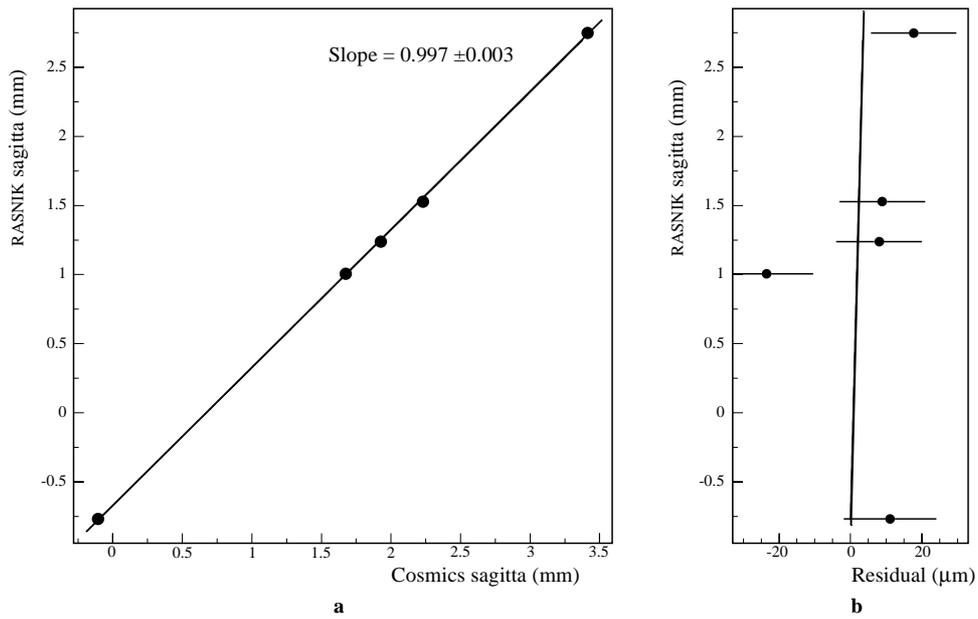


Figure 6.18 The sagitta measured by the RASNIK systems compared to the reconstructed mean sagitta values of the cosmic muons (a), and the residuals of the straight-line fit (b).

unknown chamber deformations, rotations and displacements as discussed at the beginning of section 6.2 factor out.

Figure 6.18 clearly shows that the sagitta measurement from the cosmic muons agrees very well with that of the RASNIK systems, as the slope of the straight-line fit through the data points is equal to unity within its error. The r.m.s. deviation of $15\ \mu\text{m}$ is dominated by the statistical uncertainty in the reconstructed cosmic-muon sagitta ($14\ \mu\text{m}$) and can therefore be significantly reduced by using larger data samples. However even now this number is already well below the ATLAS design target of $30\ \mu\text{m}$, although it must be realised that it does not contain any absolute-calibration error, which will be one of the larger contributions to that $30\ \mu\text{m}$.

6.4 Conclusion

After the initial problems with the MDT chambers such as gas leaks, random discharges and leak currents had been solved, the chambers operated quite well. When ignoring the disconnected tubes, the single-tube efficiencies were excellent and the noise levels were low.

The reconstruction also performed admirably, giving resolutions and track-parameter accuracies comparable to what is expected of the ATLAS software. In addition, an excellent agreement between the reconstructed sagittae of the cosmic muons and the measurements of the RASNIK alignment systems was achieved.

Happiness is a long walk with a putter.

Greg Norman

To reconstruct particles in the full ATLAS muon spectrometer the complete algorithm as described in chapter 4 must be applied. The most significant difference with the single barrel tower of the DATCHA setup is the presence of the magnetic field, requiring the use of a global track fit. To determine the performance of this algorithm and of its fit, single muons with varying momentum as well as multi-muon final states in the form of the Z^0 and Higgs decays will be investigated in this chapter.

For all these studies, the particles are created by Arve's internal generator [26] and then propagated through the magnetic field with a constant step size of 1 cm, taking into account the multiple scattering, energy loss and δ -ray production of the muons, but ignoring inner bremsstrahlung. The magnetic field is read in from the `bmagatlas02.data` file, which contains a precise 3-dimensional map of the field arising from the barrel and endcap toroids as well as from the central solenoid [15, 58]. The support structure of the toroids as shown in figure 7.1 is included in the detector description, as are the materials of the various chambers. The only

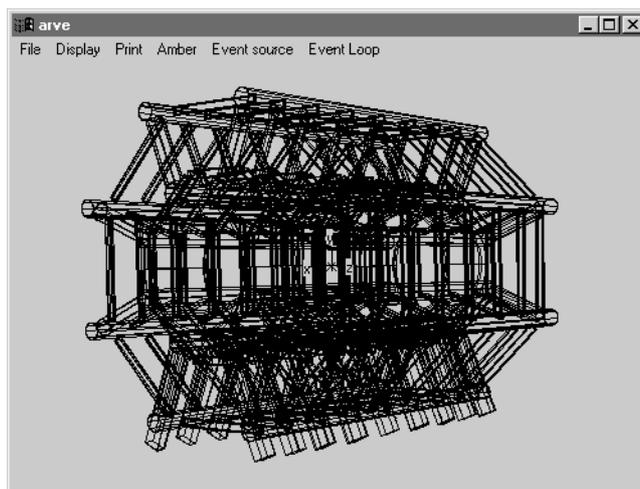


Figure 7.1 Drawing of the magnet support material included in the simulation.

difference with the material description given in section 5.1 is that in order to limit the number of volumes created, the individual tubes are replaced by sheets of aluminium of the appropriate thickness in the middle of each tube layer.

For the definition of the muon chambers the ATLAS muon database version M2.8 with the standard replacement of the CSC chambers by MDTs is used [33]. In all studies the nominal background levels and detector efficiencies as defined in section 5.1 are employed, and a perfect alignment of the chambers is assumed. Concerning the rest of the detector, the calorimeters, the coils of the solenoid and the iron return yoke are represented by cylinders containing the appropriate amount of material, whereas the inner detector is not simulated at all.

7.1 Single Muons

To study the performance of the reconstruction, samples containing both μ^+ and μ^- with various energies between 10 GeV and 1 TeV, and uniformly distributed in azimuth ($\phi \in [0, 2\pi)$) and pseudorapidity ($\eta \in [-2.5, 2.5)$) are generated. In this reconstruction a number of criteria and cuts are applied. The most important one is the required projectivity of the low- p_T trigger roads with respect to the interaction point. The three main effects that can cause a particle to deviate from a straight line are:

1. The bending in the inner detector region due to the solenoidal field. For a 2 Tesla field with a radius R of 1 meter, the deflection¹ is equal to

$$\Delta\phi_{id} \approx \frac{0.3 \cdot R \cdot B}{p_T} = \frac{0.6}{p_T} \text{ rad}, \quad \Delta\theta_{id} \approx 0 \quad (7.1)$$

2. The multiple scattering in the calorimeter region. Each scattering angle follows a normal distribution with a width σ given by [59]

$$\sigma = \frac{13.6 \text{ MeV}}{E} \sqrt{x} [1 + 0.038 \ln(x)] \quad (7.2)$$

with x the thickness of the material in radiation lengths. For the calorimeter region this number reaches a maximum of around 200, which means that for a 5σ window the scattering angles are equal to

$$\Delta\phi_c \approx \Delta\theta_c \approx \frac{1.15}{E} \quad (7.3)$$

3. The bending in the muon spectrometer. If one assumes a constant toroidal field of 1 Tesla in the barrel, the bending up to the middle station where the low- p_T trigger planes are located is equal to

1. The deflection is defined as the change in the track's direction while passing through the region in which the effect under consideration is present.

$$\Delta\phi_m \approx 0, \quad \Delta\theta_m \approx \frac{1.2}{p_T} \quad (7.4)$$

while in the endcaps the low- p_T deflection is equal to the bending caused by the endcap toroids, i.e.

$$\Delta\phi_m \approx 0, \quad \Delta\theta_m \approx \frac{1.8}{p_T} \quad (7.5)$$

In both cases the p_T is that in the muon spectrometer, which is different from the one at the vertex as used in equation 7.1.

4. The multiple scattering due to the spectrometer material. Because this effect is highly localized its quantification is difficult, which is why it is folded into the safety margin of the projectivity cuts.

By taking into account the path length to the interaction point, by using the value of the low- p_T trigger of 6 GeV (cf. section 2.3), and by including a 50% safety margin the following cuts on the projectivity of the trigger roads can be deduced:

| Region | $\max \phi_{\text{dir}} - \phi_{\text{pos}} $ (rad) | $\max \theta_{\text{dir}} - \theta_{\text{pos}} $ (rad) |
|----------------|---|---|
| barrel | 0.25 | 0.42 |
| endcaps | 0.16 | 0.45 |

Table 7.1 Projectivity requirements on the trigger roads with respect to the interaction point.

The extension of a low- p_T road into the high- p_T regime takes place only when the outer trigger stations contain a cluster that lies inside the road (cf. section 4.1.4). This not only guarantees that the clusters line up in ϕ , but also enforces the projectivity requirements on the high- p_T roads. Similarly, as the final reconstructed tracks also lie inside the trigger roads, they too fulfil these requirements.

The next set of reconstruction criteria is applied during the pattern recognition in a MDT chamber, and they are:

- For a hit to be added to a track segment, its distance to that track must be less than 500 μm , which is approximately five times the local resolution of a MDT tube;
- A track segment must consist of at least 4 hits coming from both multilayers of the chamber.

Lastly, the global track fit in the spectrometer is stopped when either the chi-squared has converged to a fixed value or when 10 iterations have been performed. In the former case, the chi-squared per degree of freedom must be less than 5 for the track to be accepted.

Most of the time that it takes to reconstruct a typical event with only one muon in the spectrometer is spent in the global track fit. The trigger reconstruction and MDT pattern recognition perform their task in a fairly constant time of between 1.8 and 2.4 seconds on a Pentium II 300 MHz machine (see also appendix B). The fit on the other hand can take anywhere between 1.4 and 6.5 seconds depending on the rate of convergence. On average, the total reconstruction time is around 6 seconds per muon.

The times quoted above are measured under nominal conditions. A reduction in detector efficiency does not lead to longer reconstruction times², so the only effect to consider is that of an increase in the number of background hits. Because the pattern recognition in the precision chambers and the subsequent global fit are restricted to the roads defined by the trigger system, an increase in the background levels in the MDTs has only a limited effect on the reconstruction times. This effect is in part caused by the increased number of combinatorials per chamber, but its contribution is small due to the small size of the roads. The largest increase in reconstruction time comes from having to perform the global fit more than once, which is needed when multiple valid track segments are found in a chamber. From the results presented in section 5.3 it follows that under the worst case conditions the fake track rate is at a level of 2% to 3% per chamber. As a consequence, the average number of attempted fits per real muon track is equal to 1.08. This means that in total the MDT background causes an approximately 10% increase in the average reconstruction time.

The chance of background hits in the trigger chambers that are not crossed by the muon to form a valid trigger road, and to then have a successful pattern recognition in the MDT chambers is small. In contrast, this is much easier for background hits that occur in the same trigger chambers that the muon traverses. Of course, they must still line up with the other clusters and with the MDT hits, but when a background hit is close enough to a real hit, multiple roads could be created, each one of them leading to a track fit. The level-1 fake trigger rate from backgrounds that create hits in a single trigger station at five times their expected levels is in the order of 10^5 - 10^6 Hz [60]. This means that even before requiring a successful pattern recognition in the MDT chambers, the effect is less than 2.5%.

7.1.1 Resolution

The resolution of the detector is evaluated by comparing the reconstructed track parameters with the Monte Carlo input. This comparison is performed at the entrance to the muon spectrometer³, thereby avoiding any contribution from the multiple scattering and energy-loss fluctuations in the calorimeter region. This also means that the interaction point can not be used in the global fit.

-
2. The only effect, which larger detector inefficiencies could have on the execution time is that by causing a reduction in the number of hits on the global track and hence a deterioration of the individual track-segment parameters, the global fit could need more iterations to converge. However, given the results of chapter 5 this effect is negligible for the inefficiencies expected in ATLAS.
 3. A cylinder with a radius of 425 cm and a length of 1364 cm.

From a reconstruction point of view, the spectrometer can be divided into 3 regions in pseudorapidity (see also figure 2.4), viz.

- The barrel ($0 \leq |\eta| \leq \sim 1.0$) in which a sagitta measurement with three chambers parallel to the beam axis is performed;
- The overlap region ($\sim 1.0 < |\eta| < \sim 1.4$) in which three vertical chambers are used to perform a sagitta measurement;
- The endcaps ($\sim 1.4 \leq |\eta| \leq \sim 2.7$), which contain three vertical chambers with the first placed before the end-cap magnet while the other two are located behind it. Hence a point-vector measurement is performed.

In the barrel this yields the $(z, R\phi, \phi, \theta)$ coordinates for the track's position and direction, while in the other two regions the z -coordinate is replaced by the radius R . For a sample of 50 GeV muons, the positional and directional resolutions in the barrel are shown in figure 7.2⁴. The coordinates on the left are determined solely based on the hits in the trigger chambers, and are therefore directly related to the widths and opening angles of the roads. These in turn follow from the size of the strips, which are around 35 mm, and the relative positioning of the stations. The resulting opening angles are about 10 mrad, which converts to an effective resolution of $10/\sqrt{6} = 4.1$ mrad due to the triangular shape of the angle distribution. This number compares well with the reconstructed value in figure 7.2c, with the difference being caused by multiple scattering.

The uncertainty in the $R\phi$ -position of the reconstructed tracks can be derived from the ϕ -resolution by following the trigger roads from their origin, which is the centre point between the two low- p_T trigger stations, back to the entrance of the spectrometer. If one also folds in the width of the strips, the resulting resolution is 1.9 cm (see figure 7.2a).

Both these numbers are obtained in the barrel. If the same RPC chambers would be used in the overlap regions and in the endcaps, the resolution there would be much worse. The reason for this is that the trigger layers in the endcaps are positioned much closer together than their counterparts in the barrel (cf. figure 2.4). To cancel this effect, the TGC chambers have a much finer granularity than the RPCs, which leads to comparable resolutions in the ϕ -projection throughout the whole spectrometer.

The remaining two coordinates in figure 7.2 are derived from the precision chamber hits. This immediately translates into higher resolutions, but it also means that they are much more susceptible to the effects of multiple scattering. Compared to the barrel, the overlap regions and endcaps show a significant deterioration of these resolutions. For the former, this is caused by the irregular bending power caused by the mixing of the barrel and endcap magnetic fields (cf. figure 2.5). This effect is still present at the edge of the endcap region, while for higher pseudorapidities the material of the endcap toroids starts to play a role.

4. There is no observable difference between the behaviour of the muons and the anti-muons, and hence their resolutions are combined into a single plot.

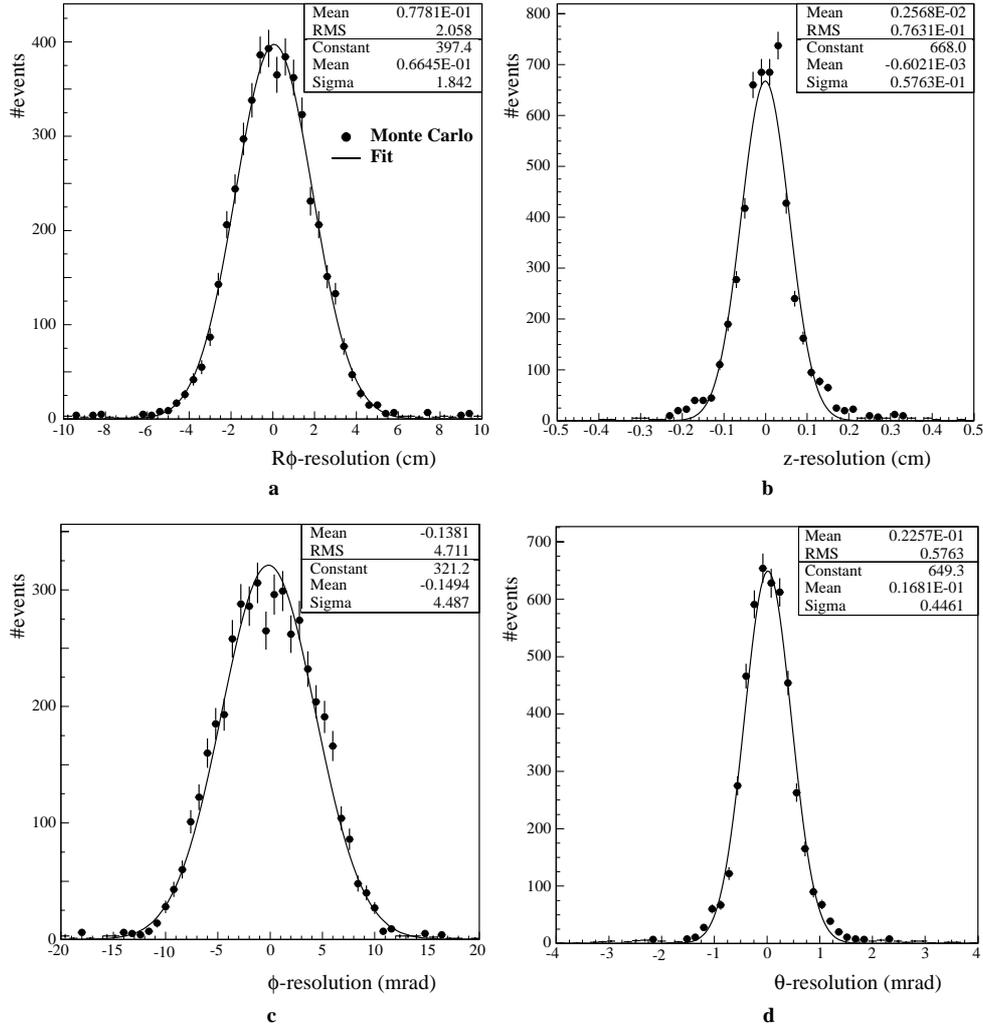


Figure 7.2 Resolutions of the Rφ (a) and z (b) positions, and of the φ (c) and θ (d) angles at the entrance to the barrel of the muon spectrometer ($|\eta| \leq 1$) for a sample of 50 GeV muons.

The best way to study this behaviour is to look at the resolution of the fifth independent track parameter, i.e. the transverse momentum. Its distribution is presented in figure 7.3a, followed by its dependence on the p_T , both for the barrel. The resolution is a combination of two effects, viz. the multiple scattering in the detectors and magnet support structures and the intrinsic resolution of the MDT chambers. From equation 7.2 one would expect the effect the multiple scattering has on the resolution to fall off with increasing momentum. However, because the hits in a chamber combine to form a vector measurement, the determination of the sagitta becomes more accurate at low muon energies. Therefore, for the barrel the effect of multiple scattering is fairly constant at around 2%.

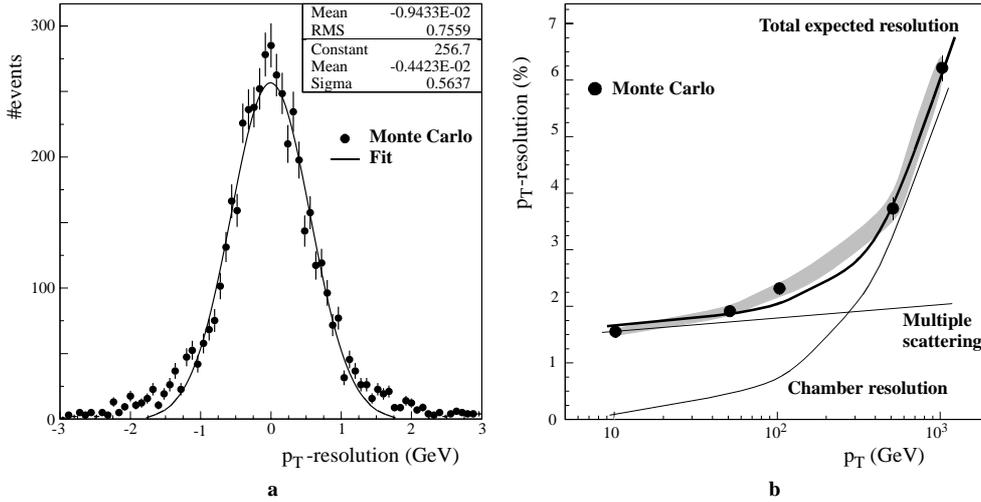


Figure 7.3 Momentum resolution of 50 GeV muons in the barrel, measured at the entrance to the muon spectrometer (a) and the p_T -dependence of this resolution (b). The grey band represents the error in the reconstructed resolution, while the solid lines are the results of theoretical calculations of the various contributions to that resolution [15].

The contribution of the intrinsic resolution of the precision chambers is small at low momenta. However, above a p_T of around 100 GeV it starts to rise sharply and quickly dominates the resolution. An analogous effect would be observed from possible chamber misalignments, which have not been considered in the simulation. They would deteriorate the p_T -resolution to around 8% at 1 TeV. Furthermore, to be able to compare the reconstructed tracks with those in the inner detector, the uncertainty in the correction of the energy loss in the calorimeters must also be factored into the resolution. This effect dominates at low energies, but it drops off quickly from 3% at 10 GeV to only 0.4% at 100 GeV.

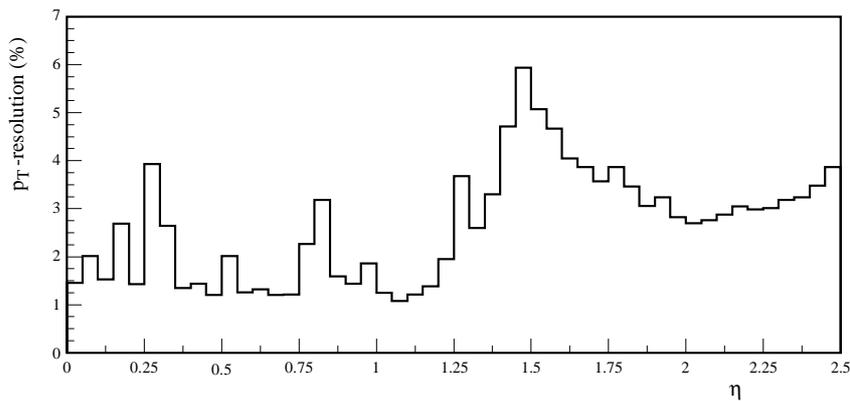


Figure 7.4 The p_T -resolution as a function of η for 50 GeV muons halfway between a barrel and endcap coil plane ($\phi = 11.25^\circ$).

In the endcaps, the behaviour of the resolution as a function of the p_T is about the same, all be it with slightly higher values in the multiple-scattering dominated range, i.e. for momenta below 100 GeV. From figure 7.4 in which the p_T -resolution is shown as a function of the pseudorapidity for an azimuthal angle halfway between a barrel and an endcap coil, it is clear that the resolution in the overlap region is severely degraded as a result of the reverse field of the endcap weakening the magnetic field generated by the barrel toroid. Also clearly visible are the magnet support structures, which show up as spikes in the p_T -resolution.

7.1.2 Efficiency

Based on the Monte Carlo information, reconstructed tracks can be classified as “good” when the fitted values of all five track parameters fall within 5σ of their true values. All other tracks are then designated “fake” and their rate normalized to the trigger efficiency is shown in figure 7.5 as a function of the muon’s momentum. The rise at high momenta is caused by the production of secondary particles by the muon. Their hits can obscure the real muon hits, which results in incorrect drift time measurements and therefore a lower segment reconstruction efficiency, and they can create hits in tubes not crossed by the muon, thereby increasing the chance that a fake track is found.

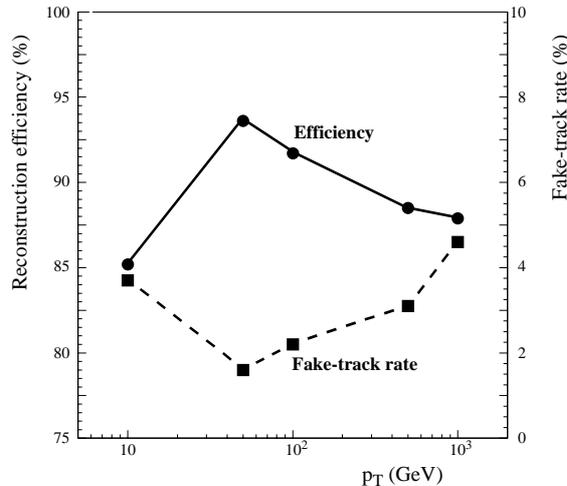


Figure 7.5 Single-muon reconstruction efficiency (circles) and fake-track rate (squares) as a function of p_T .

The increase at low momenta is mostly due to multiple scattering, which causes an incorrect assignment of the track parameters by the fit. But the real effect the multiple scattering has on the reconstruction is in the efficiency, which is also shown in figure 7.5. The enormous drop at low momenta is caused by the requirement that the chi-squared per degree of freedom

of the global track fit must be less than five, which is not attainable when the track has a kink caused by a piece of material.

The only way to resolve this problem is to have a detailed description of the matter in the spectrometer, and to explicitly take these multiple scattering points into account in the track fit. Each such point then introduces two new parameters to the track fit, viz. the scattering angles θ and ϕ with their starting values set to the result of equation 7.2 [57]. This does however require an efficient method for the determination of the material traversed by the muon. One possible solution would be the use of a precalculated lookup table in ϕ , η and p_T , but this requires further investigation.

Separate simulation runs without the magnet and chamber support structures present show that the efficiencies can be improved to 96% and 97% for 10 GeV and 50 GeV muons respectively. Compared to other results obtained by the ATLAS collaboration [10] this still falls somewhat short, which means that further development of the reconstruction algorithm and its fit are needed.

7.1.3 Charge Identification

A correct identification of the charge of a muon is essential for a wide range of physics topics, from the CP violation in B physics to the couplings of new heavy gauge bosons [10]. The problem is that with increasing momentum, the bending of a muon's trajectory diminishes, making it ever more difficult to correctly reconstruct not only its momentum, but also its charge. In figure 7.6 the probability of charge misidentification, which is defined as the ratio of muon

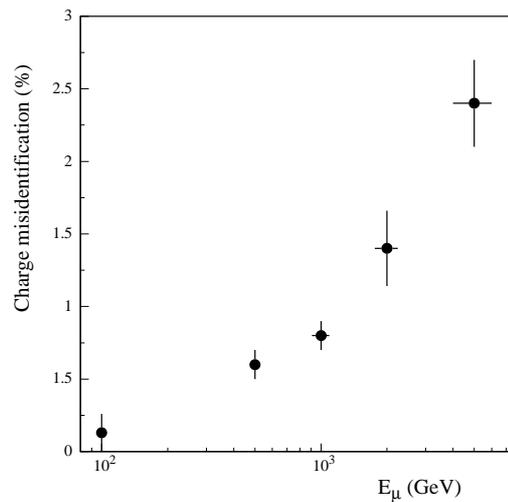


Figure 7.6 Percentage of reconstructed muons with a misidentified charge as a function of their energy. The error bars are purely statistical in origin.

5. Because in the second-coordinate projection the resolution is not dominated by multiple scattering, it might be possible to drop the ϕ -parameters, leaving only the θ -angles.

tracks reconstructed with the wrong charge to the total number of reconstructed tracks, is shown. Up to energies of 5 TeV, the charge misidentification rate remains below the 2.5%, making it a manageable effect.

7.2 $Z \rightarrow \mu^+ \mu^-$

In a large and high-precision system like the ATLAS muon spectrometer, good calibration and alignment are of vital importance. Physics resonances like the $Z \rightarrow \mu^+ \mu^-$ decay, which are abundantly produced by the LHC, offer an excellent way to both align the muon chambers [61] and to calibrate the absolute mass scale through an indirect determination of the magnetic field strength and the energy loss in the calorimeter [62]. In addition, the process forms the basis of the Higgs decay into 4 muons, which will be studied in the next section.

In a sample of Z bosons generated with varying boosts along the beam axis, the muons are reconstructed in the spectrometer and subsequently propagated back to the interaction point, taking into account the energy loss in the calorimeter. Although the actual energy deposited in the cells that the muon has traversed could be used, this procedure frequently overestimates the energy, and only works well for isolated high- p_T muons. So instead, an average energy loss based on the particle's trajectory and momentum is used.

The in this way reconstructed vector sum of the momenta of the two muons at their point of closest approach to the interaction point must be equal to the boost of the Z boson. As can be seen from figure 7.7a this relationship holds very nicely, although the reconstructed value does tend to slightly overestimate the simulated one. The reason for this is that the mean energy loss in the calorimeter is used during the backtracking of the muon, while the most probable energy loss is somewhat lower [62].

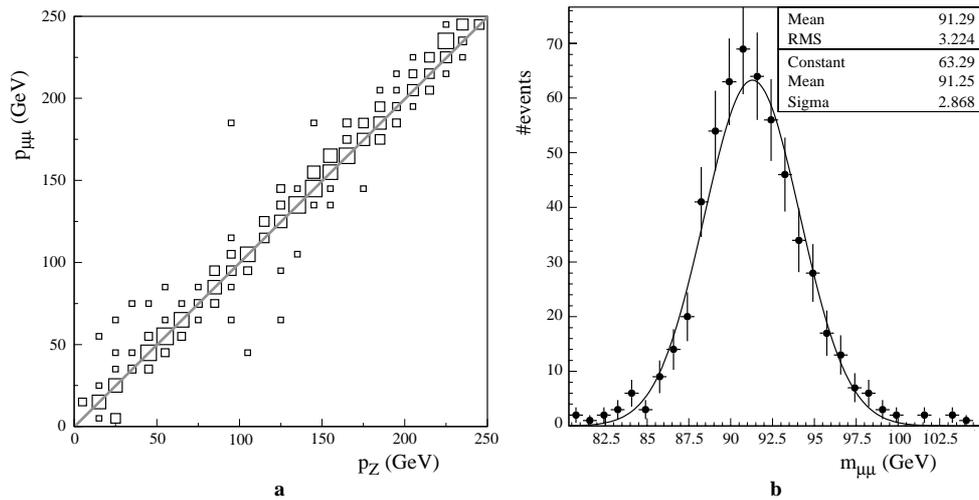


Figure 7.7 Reconstructed two-muon momentum versus the simulated boost of the Z boson (a) and the reconstructed mass distribution in the $Z \rightarrow \mu^+ \mu^-$ decay (b).

With the help of the four-momentum conservation law, the muon momenta can be converted back to the mass of the Z. When the muon mass is neglected, conservation of energy gives

$$m_Z^2 c^2 + p_Z^2 = p_1^2 + p_2^2 + 2p_1 p_2 \quad (7.6)$$

with p_1 and p_2 the absolute values of the momenta of the two muons. Similarly, conservation of the three-momentum results in

$$p_Z^2 = p_1^2 + p_2^2 + 2p_1 p_2 \cdot [\cos\theta_1 \cos\theta_2 + \sin\theta_1 \sin\theta_2 \cos(\phi_1 - \phi_2)] \quad (7.7)$$

with θ_i and ϕ_i the muons' polar angles. From these two equations p_Z^2 can be eliminated, leading to an expression of the Z mass in terms of the kinematical properties of the two muons:

$$m_Z^2 c^2 = 2p_1 p_2 \cdot [1 - \cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2 \cos(\phi_1 - \phi_2)] \quad (7.8)$$

which evaluates to $m_Z = 91.25 \pm 0.13$ GeV as shown in figure 7.7b. The measured standard deviation of 2.87 ± 0.11 GeV is a combination of the natural width of the Z, which contributes 1.1 GeV⁶, the measurement accuracy of the muon system (see section 7.1.1), the error introduced by the backtracking of the muons and the fluctuations of the energy loss in the calorimeter. It can therefore be improved by combining the tracks with those reconstructed in the inner detector. For muons with a transverse momentum in the range of 30 to 100 GeV, the sum of the intrinsic resolution of the muon system and the energy-loss fluctuations in the calorimeter is about the same as the resolution of the inner detector. For higher-momentum muons, the muon spectrometer is more accurate, while for low-momentum particles the inner detector does a better job [10].

7.3 $H \rightarrow ZZ^{(*)} \rightarrow \mu^+ \mu^- \mu^+ \mu^-$

In the same fashion as for the Z bosons, the Higgs decay into 4 muons is investigated using only the reconstruction in the muon spectrometer. A typical event, projected onto the yz-plane is shown in figure 7.8. Its efficiency, normalized to the trigger efficiency is equal to about 65%, which is a direct product of the single-muon efficiencies quoted in section 7.1.2.

For a 130 GeV Higgs, the resulting mass resolution is shown in figure 7.9a. Because one of the Z bosons is on-shell, a Z-mass constraint can be applied to the pair of different-sign muons whose combined mass lies closest to the Z-mass. This improves the resolution by around 14% to 2.5 GeV, with approximately 82% of the events inside a mass window of $\pm 2\sigma$ around the Higgs mass (see figure 7.9b).

6. The conversion is defined as $\Gamma \rightarrow \sigma = \Gamma / (2\sqrt{2\ln 2})$ with the full width of the Z equal to 2.490 ± 0.007 GeV.

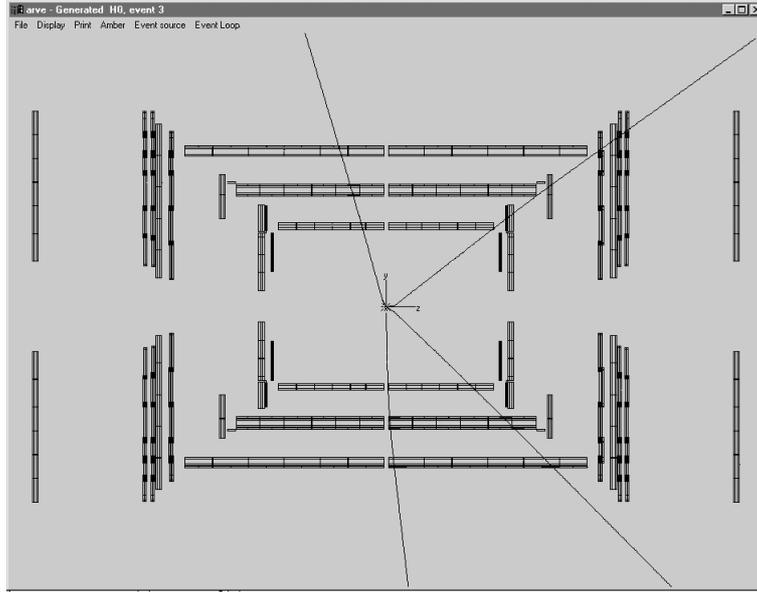


Figure 7.8 Side view of the muon spectrometer showing two large sectors, with a superimposed $H \rightarrow 4\mu$ event.

This 2% mass resolution at 130 GeV rises slowly with increasing Higgs mass as a result of the increasing natural width of the Higgs (see figure 7.10) and the deteriorating single-muon

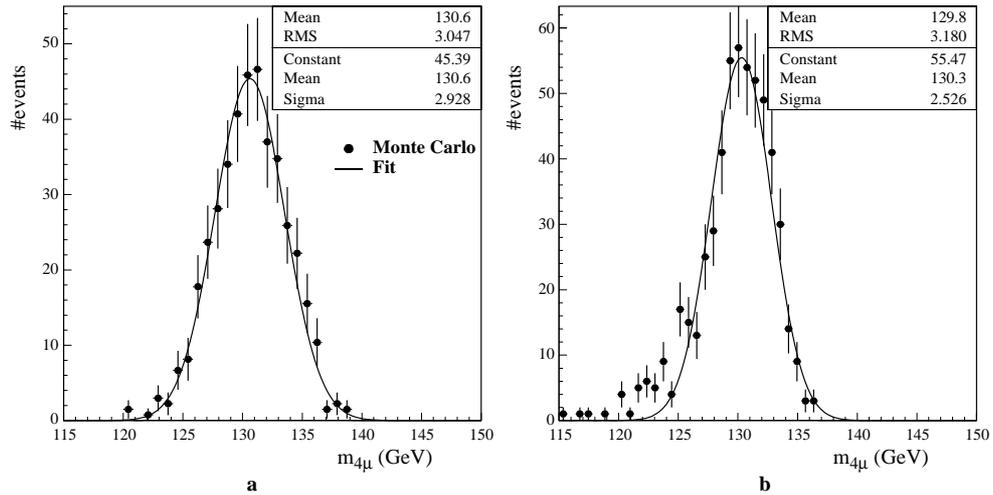


Figure 7.9 Reconstructed Higgs-mass distribution for a 130 GeV Higgs decaying into four muons without (a) and with (b) applying a Z-mass constraint.

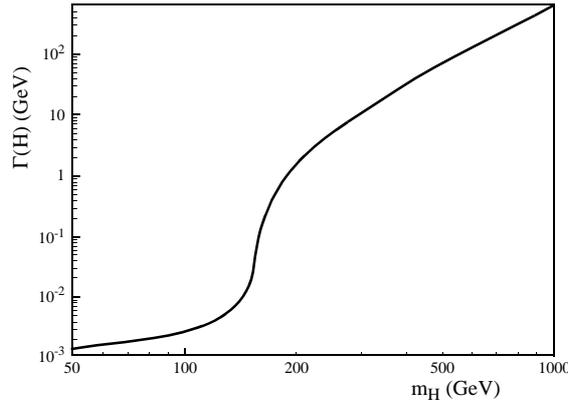


Figure 7.10 Natural width of the Higgs boson as a function of its mass.

resolution, reaching a value of 2.6% at 300 GeV. At this point the width of the Higgs starts to dominate over the resolution of the detector.

As for the Z boson, the mass resolution of the Higgs can be improved by using the inner detector information. This will be discussed in the next chapter in which a complete study of the Higgs decay into four leptons will be given.

7.4 Conclusion and Outlook

The performance of the event reconstruction as described in this chapter shows that AMBER is able to function well in a complicated environment like the ATLAS muon spectrometer. The various selection criteria used by the program are designed to optimize the accuracy of the reconstruction. Hence, the resolutions obtained in the single-muon events agree very well both with theoretical predictions and with other ATLAS results [10, 15]. The same is true of the Z and Higgs mass resolutions, although it must be kept in mind that the inner detector was not, and the calorimeter only partly included in the simulation.

On the downside, these stringent cuts lead to efficiencies well below what is desirable of the offline muon reconstruction. This is to the most part due to the fact that the global track fit does not take any multiple scattering into account. This is especially a problem when a muon crosses one of the chamber or magnet support materials. Detailed analysis on an event by event basis has shown that when a muon does not intersect with such a structure, the chance of it being reconstructed correctly is much higher. However, there remain cases in which the fit does not converge because the fluctuations of the magnetic field are too large. Furthermore, some algorithmic instabilities in the region between the barrel and endcap, and in the overlap region between two sectors are experienced. In all, the efficiencies would approach the values quoted in the technical design reports [10, 15], but some effort would still be required to capture the last couple of events.

*This is not the end. It is not even the beginning of the end.
But it is, perhaps, the end of the beginning.*

Winston Churchill

The decay of the Higgs into four muons as described at the end of the previous chapter is only one of the three possible processes that make up the gold-plated Higgs to four lepton decay channel. Because the currently available information points to a Higgs mass between 109 GeV (direct LEP limit) and 215 GeV (upper limit at 95% confidence level), both the decay into an intermediate state of two real Z bosons, and the decay into a real and a virtual Z have to be considered (cf. section 1.4). The results as presented in this chapter have been obtained using a full GEANT detector simulation including inner bremsstrahlung [43, 44].

8.1 Signal Reconstruction

The signal reconstruction of all Higgs to four lepton decay channels starts with the successful reconstruction of two pairs of leptons with the correct charges, after a set of simple kinematical cuts has been applied:

1. The signal is triggered on two leptons with a transverse momentum above 20 GeV and $|\eta| < 2.5$;
2. The other two leptons must also fall in the pseudorapidity range $|\eta| < 2.5$, and have a $p_T > 7$ GeV. In the case of electrons, both must fall outside the crack region between the barrel and the endcaps, i.e. $1.37 < |\eta| < 1.52$.

In addition, two mass cuts can be applied because of the Z boson(s) in the intermediate state. When the mass of the Higgs is less than $2m_Z$, these cuts are:

3. One pair of leptons of the correct charge and flavour is required to have an invariant mass m_{12} in a window around the Z mass, i.e.

$$|m_{12} - m_Z| \leq \Delta m_{Z \text{ window}} \quad (8.1)$$

4. The remaining pair must have an invariant mass above a certain threshold called $m_{34, \min}$.

When the Higgs is heavier than the two Z bosons, both pairs of leptons must fulfil the Z-mass requirement, and cut number 4 becomes obsolete.

It is clear from the definition above that the $m_{34, \min}$ threshold must be a function of the mass of the Higgs. But in addition, the size of the Z-mass window has also been made to depend on it in order to (partially) recover the acceptance losses due to bremsstrahlung. The values of these cuts for the different Higgs masses are listed in table 8.1:

| Higgs mass (GeV) | 130 | 150 | 170 | 180 | $> 2m_Z$ |
|----------------------|-----|-----|-----|-----|----------|
| Z-mass window (GeV) | 15 | 10 | 6 | 6 | 6 |
| $m_{34, \min}$ (GeV) | 20 | 30 | 45 | 60 | |

Table 8.1 Size of the Z-mass window for the first lepton pair, and the invariant-mass threshold for the other pair as a function of Higgs mass.

Because the three possible final states are subject to different detector performances, the results obtained in their reconstruction also differ.

8.1.1 $H \rightarrow ZZ^{(*)} \rightarrow e^+e^-e^+e^-$

The reconstruction of the four-electron final state is performed based on the hits in the inner detector and the clusters in the electromagnetic calorimeter [10]. When the Z-mass constraint is applied to the electron pair closest to the Z mass, the invariant mass distribution for a Higgs mass of 130 GeV is shown in figure 8.1. The resolution of 1.6 GeV shown there is the result at low luminosity; at high luminosity the value increases to 1.7 GeV.

The average electron reconstruction efficiency is 92%, corresponding to a 72% four-electron identification efficiency. Furthermore, approximately 82% of the events fall within a mass window of $\pm 2\sigma$ around m_H . This is less than the expected 95% for a gaussian distribution, which is mainly due to inner bremsstrahlung.

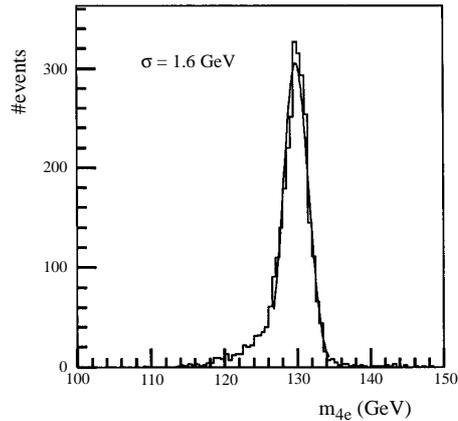


Figure 8.1 Higgs-mass distribution in the four-electron decay channel for $m_H=130$ GeV.

8.1.2 $H \rightarrow ZZ^{(*)} \rightarrow \mu^+ \mu^- \mu^+ \mu^-$

For the stand-alone muon spectrometer the mass resolution in the four-muon channel has been determined by AMBER (see the previous chapter). As this result agrees very well with that of the ATLAS production software, the latter's improvement by using the tracks found in the inner detector can also be applied to it [10]. The result is a resolution of 1.4 GeV for a Higgs mass of 130 GeV with an acceptance in the $\pm 2\sigma$ mass window around the Higgs mass of about 83%. This is after the efficiency for reconstructing all four muons, which is found to be 83.7% has been taken into account. Moreover, unlike in the electron channel, these numbers are not affected by pile-up and remain the same at high luminosity.

8.1.3 $H \rightarrow ZZ^{(*)} \rightarrow e^+ e^- \mu^+ \mu^-$

The mixed final state of 2 electrons and 2 muons is a simple combination of the reconstructions described above. For a Higgs mass less than twice the Z boson mass, the mass resolution differs somewhat between the state in which the on-shell Z decays into two electrons, and the state in which it decays into two muons. The reason for this is that these two leptons have a harder p_T -spectrum than the other ones, and while the resolution of the muon spectrometer degrades as the transverse momentum increases, it improves for electrons because of the calorimetric energy measurement.

For a Higgs mass of 130 GeV, the resolutions are 1.3 GeV and 1.6 GeV respectively, with an average value of 1.5 GeV. At high luminosity the degradation of the electron energy reconstruction causes the resolution to rise to 1.6 GeV. In both cases approximately 82% of the events are reconstructed in the $\pm 2\sigma$ centre of the mass distribution.

The cross sections times branching ratios for the $H \rightarrow 4l$ decay channel are listed in table 8.2 as a function of the Higgs mass. The number of expected events is derived by taking into account the acceptances of the kinematical cuts and those of the Higgs mass bin, as well as the lepton identification efficiencies.

| m_H (GeV) | 130 | 150 | 170 | 180 | 200 | 300 | 400 |
|--------------------|------|------|------|------|------|------|------|
| $\sigma * BR$ (fb) | 2.97 | 5.53 | 1.40 | 3.26 | 12.4 | 9.10 | 6.76 |
| #events | 69 | 164 | 46 | 119 | 525 | 415 | 336 |

Table 8.2 Expected number of events in the $H \rightarrow ZZ^{(*)} \rightarrow 4l$ decay channel for an integrated luminosity of 100 fb^{-1} . The number of signal events is calculated after having applied the kinematical cuts, and assuming the lepton reconstruction efficiencies and mass bin acceptances quoted above [8].

8.2 Background

The background to the Higgs decay into four leptons consists of three different processes. The first two, viz. the $t\bar{t}$ ($t\bar{t} \rightarrow WbW\bar{b} \rightarrow 4l$) and the $Zb\bar{b}$ productions are reducible, while the third

process is not. For intermediate Higgs masses the latter consists of the ZZ^* and the $Z\gamma^*$ continuum productions, with an additional small contribution from the ZZ continuum where one of the Z bosons decays into a tau anti-tau pair, which subsequently decay leptonically, and the other Z decaying into two electrons or two muons. When the mass of the Higgs is higher than $2m_Z$, their role is taken over by the ZZ and $\gamma^*\gamma^*$ continuum.

The expected number of background events integrated over a mass window of ± 5 GeV around the corresponding Higgs mass and after the kinematical cuts have been applied is listed in table 8.3. Combining these values with the number of signal events from table 8.2 shows that the signal significance for high Higgs masses easily exceeds the 5σ level after one year of high-luminosity running. However, when the Higgs has an intermediate mass, additional cuts are required to reduce the number of background events.

| m_H (GeV) | 130 | 150 | 170 | 180 | 200 | 300 | 400 |
|---------------------------------------|-----|-----|-----|-----|------|------|------|
| $ZZ^{(*)}/Z\gamma^*/\gamma^*\gamma^*$ | 24 | 24 | 22 | 19 | 290 | 152 | 113 |
| $t\bar{t}$ | 148 | 194 | 148 | 132 | | | |
| $Zb\bar{b}$ | 101 | 132 | 101 | 93 | | | |
| S/\sqrt{B} | 4.2 | 8.8 | 2.8 | 7.6 | 30.8 | 33.7 | 31.6 |

Table 8.3 Expected number of background events for an integrated luminosity of 100 fb^{-1} after the kinematical cuts have been applied, and assuming the lepton reconstruction efficiencies quoted above. The last line shows the signal significance based on the number of events from table 8.2.

8.2.1 Irreducible Background

Kinematical cut number 4, i.e. the requirement that the lepton pair with the lowest transverse momentum must have an invariant mass above a certain threshold, has already considerably reduced the number of $Z\gamma^*$ background events in the electron channel, as well as the contribution from cascade decays of b quarks¹. For the intermediate Higgs masses, no additional cuts can be applied to reduce this background, so the numbers in table 8.3 remain valid.

In the case of a heavy Higgs particle a rejection of the continuum ZZ background can however be achieved. Because the intermediate Z bosons in a Higgs decay are produced through a 2-body decay of a heavy object, they have a much harder p_T -spectrum than the Z bosons in the continuum background. By requiring that the maximum transverse energy of the two Z bosons is larger than a given threshold. i.e.

$$p_{T, \max}(Z_1, Z_2) > m_H/3 \quad (8.2)$$

the signal significances can be substantially improved:

1. A cascade decay of b quarks is a four-lepton event in which at least one lepton is not directly produced through a semileptonic decay of a W boson or a b quark.

| m_H (GeV) | 200 | 300 | 400 |
|---------------------|------|------|------|
| Signal | 211 | 352 | 298 |
| ZZ continuum | 27 | 66 | 54 |
| S/\sqrt{B} | 40.6 | 43.3 | 40.6 |

Table 8.4 Number of signal and background events for an integrated luminosity of 100 fb^{-1} after the cut on the maximum transverse energy of the Z bosons has been applied (cf. tables 8.2 and 8.3).

8.2.2 Reducible Background

At the production level, the non-resonant $t\bar{t}$ background dominates, but because of the Z-mass constraint (kinematical cut number 3) it is strongly suppressed. This is not the case for the $Zb\bar{b}$ background, because of the real Z boson in the final state. Only for Higgs masses above the $2m_Z$ -threshold when both lepton pairs must originate from an on-shell Z boson, can it be effectively suppressed.

As is clear from table 8.3, the kinematical cuts alone are not sufficient for a clear recognition of the Higgs in the intermediate mass range. In particular, it would be desirable to bring the reducible background below the irreducible one as there are large theoretical uncertainties in the former's calculation. In order to lower it to 10% of the reducible background, a rejection factor of around 100 is needed. To achieve this, both a lepton isolation and an impact-parameter cut must be applied.

Lepton Isolation Cut

In the $Zb\bar{b}$ and $t\bar{t}$ backgrounds the leptons tend to be non-isolated as they are accompanied by other decay products of the b quarks. In the inner detector a lepton isolation cut can be implemented by requiring that no charged tracks with a momentum above a certain threshold are found in a cone around the lepton. This same isolation cut can also be achieved by requiring that the sum of the transverse energy deposited in the calorimeter in a cone around the lepton is less than a given value. Because these criteria are strongly correlated, it turns out to be sufficient to apply only the one based on the inner detector information [63].

The distribution of the maximum transverse momentum of the charged tracks inside the cones with radius

$$R = \sqrt{\delta\phi^2 + \delta\eta^2} = 0.2 \quad (8.3)$$

around all four leptons is shown in figure 8.2a. That plot is for a Higgs mass of 130 GeV, but the results depend only very weakly on m_H . By varying the cut on this momentum, the achieved rejections as a function of the Higgs to four muon efficiency are displayed in figure 8.2b. The rejection in the $Zb\bar{b}$ channel is lower, because of the softer p_T -spectrum of its final state muons.

In the case of the 4-electron final state, the isolation criteria are partially spoiled by bremsstrahlung in the inner detector. This reduces the signal efficiency by about 10% compared

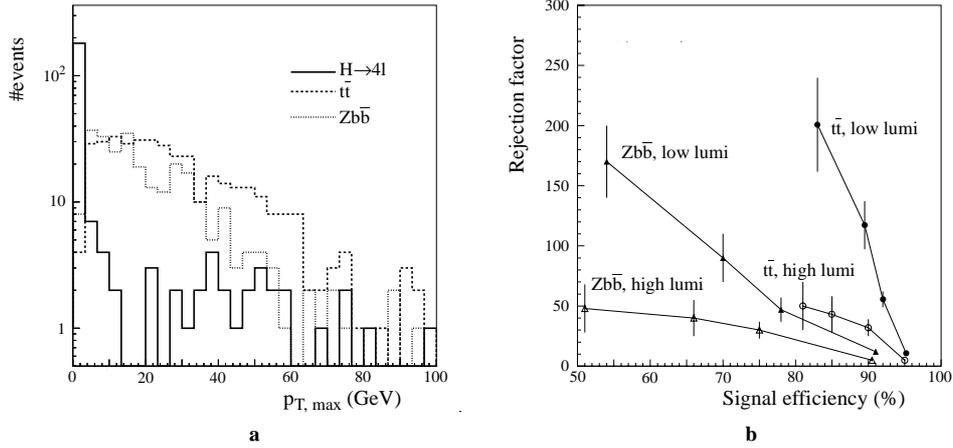


Figure 8.2 Maximum p_T in the cones of radius 0.2 around the 4 leptons for a Higgs mass of 130 GeV at low luminosity (a) and the rejection as a function of the $H \rightarrow 4\mu$ reconstruction efficiency after the kinematical cuts (b).

to those shown in figure 8.2b for a fixed rejection factor. In the same fashion, the pile-up at high-luminosity leads to a 25% loss in efficiency.

Impact Parameter Cut

The impact parameters of the four leptons provide a second means of rejecting part of the reducible background, because the leptons coming from the $t\bar{t}$ and $Zb\bar{b}$ backgrounds originate at a displaced vertex. However, as this displacement is in part masked by the $14 \mu\text{m}$ transverse spread of the beam, the maximum transverse impact parameter is not a very good selection criterion. Instead, the summed distance in the transverse plane between the six intersection points that result from considering the leptons two-by-two, i.e.

$$D = \sum_{i,j} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (8.4)$$

can be used. By requiring it to be less than 1.5 mm, a rejection of 12 of the $t\bar{t}$ and 5 of the $Zb\bar{b}$ background channels are achieved for a 90% signal efficiency in the 4-muon channel. As with the other method, the rejection in the $H \rightarrow 4e$ channel is affected by inner-bremsstrahlung, leading to a loss of efficiency of around 20% for the same rejection factors.

When going to high luminosity, the impact parameter method suffers considerably from the removal of the B-layer, i.e. the strip layer closest to the vertex. Hence, the signal efficiency for a fixed rejection factor falls by 40% for the $t\bar{t}$ events, and by 30% in the case of the $Zb\bar{b}$ background.

Combined Rejection Factors

The two background rejection methods are partially correlated as the lepton-isolation cut changes both the momentum distribution of the leptons and the fraction of cascade-decay events in the background. This correlation is found to be around 40% for the $Zb\bar{b}$ and 10% for the $t\bar{t}$ background. The higher number for the $Zb\bar{b}$ events is due to the fact that the lepton isolation cut softens the p_T -spectrum, making the impact parameter resolution multiple-scattering limited. By combining both methods, the rejection factors for low-luminosity running total around 1200 for the $t\bar{t}$ and 118 for the $Zb\bar{b}$ backgrounds at a signal efficiency of 69%. At high luminosity these values are respectively 810 and 70 for a signal efficiency of 52%. This means that the reducible background has been brought well below the irreducible one.

8.3 Conclusion

Because the signal rates in the $H \rightarrow 4l$ channel are small once all the cuts have been applied, an efficient reconstruction of the multi-lepton final states is of the utmost importance. But when that is achieved, the reconstruction is able to deliver the signal significances plotted in figure 8.3 for one year of running at high luminosity, and in figure 8.4 for three years of low-luminosity

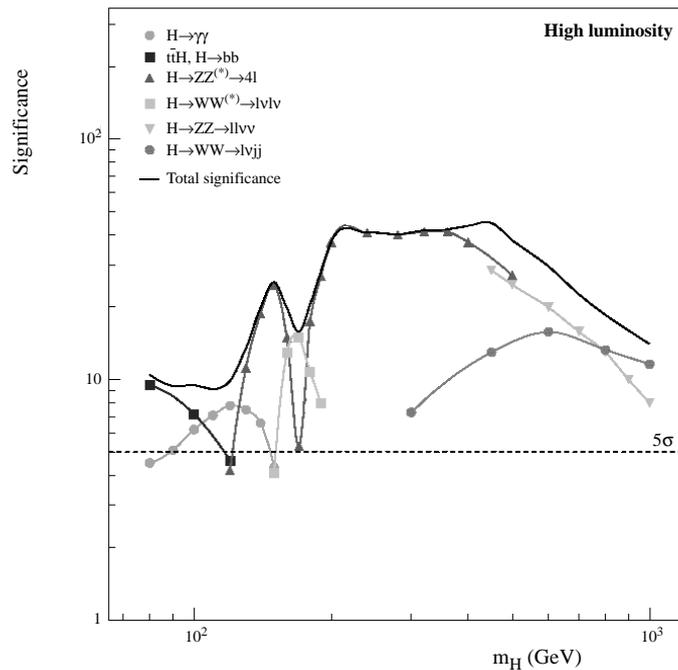


Figure 8.3 Sensitivity for the discovery of a standard model Higgs boson by the ATLAS detector as a function of the Higgs mass at an integrated luminosity of 100 fb^{-1} [10].

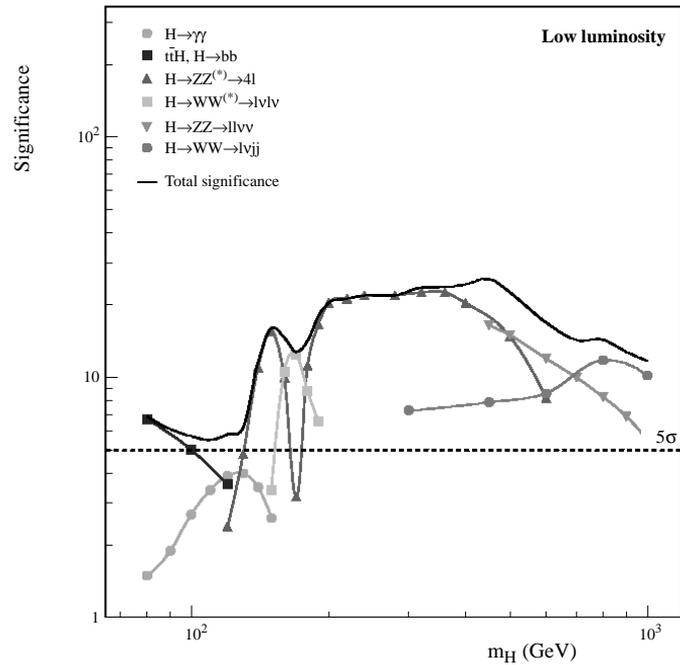


Figure 8.4 Sensitivity for the discovery of a standard model Higgs boson by the ATLAS detector as a function of the Higgs mass at an integrated luminosity of 30 fb^{-1} [10].

running. In the first case, the ATLAS detector will be capable of finding the Higgs boson with a high significance in the mass range between 130 and 500 GeV through its decay into four leptons. Below and above that range other decay channels are needed, and exist, to lift the significance above the 5σ level. Moreover, at low luminosity a discovery of 5σ of the Higgs over its full mass range is also possible after only a few years of running.

APPENDIX A | Notation

*When I use a word it means just what I choose it to mean;
neither more nor less.*

Humpty Dumpty

A.1 Unified Modelling Language

To communicate efficiently one needs a common language and software design is no exception to that rule. From the late 1980's onwards many object-oriented analysis and design methods¹ were introduced for exactly that reason, but sadly their proliferation negated their usefulness. Fortunately, out of this wave of methods a recent standard has arisen: the Unified Modelling Language or UML [64]. It defines a set of diagrams each representing a different part of, or view on the analysis or design. Only three of these diagrams are used in this thesis, viz. the package, class and interaction diagrams. Their basic syntax is explained in the next sections, but first some general concepts are presented.

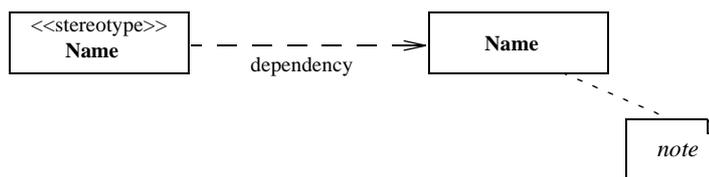


Figure A.1 General-purpose concepts in the UML.

An entity such as a package or a class is usually represented by a box of some sort, containing its name and an optional stereotype. A stereotype is a general attribute either defined by the UML or by the user. Examples of attributes used in this thesis are `<<interface>>` (all methods of the class are abstract), `<<abstract>>` (some of the methods are abstract) and `<<external>>`.

1. A method consists, in principle, of both a modelling language and a software process. The language is the (mainly graphical) notation, while the process is its advice on what steps to take in creating a design (see e.g. [24]).

A dependency between two entities is always represented by an arrow. The single-sided arrow shown in figure A.1 depicts a unidirectional relationship, but bidirectional ones are also possible. When a dependency line is crossed by a slash ('/'), it means that the connection is a derived one, i.e. it depicts a relationship existing somewhere else, e.g. between two base classes.

A.1.1 Package Diagrams

A package diagram is a high-level diagram showing groups of classes (the packages) and the dependencies among them. A package can be opaque, meaning that its internals are not visible from the outside, or it can be transparent. In the latter case, nested packages or even individual classes can be shown on the diagram. Dependencies can then not only point to the package itself, but also to an entity within it.

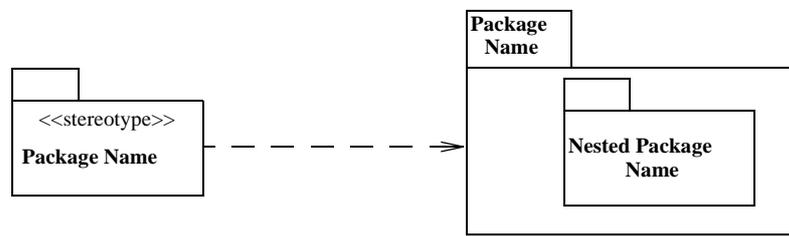


Figure A.2 Package diagram syntax.

One of the possible stereotypes of a package is `<<global>>`, which means that all other packages (can) reference it.

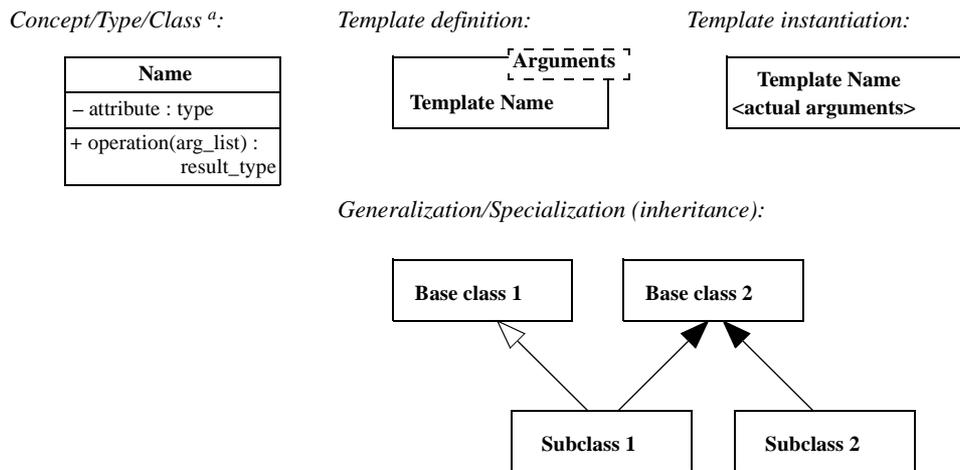
A.1.2 Class Diagrams

A class diagram can be used to show the static structure of either concepts, types or classes²:

- In its conceptual view the diagram can be used to depict the way users think about the world, but also to show an overall, high level view of the design;
- In its interface view it shows the interfaces of the software components;
- And finally in its implementation view the actual classes are displayed.

In all three cases the syntax is identical, which means that each diagram must be accompanied by an explicit statement of its type. However, unless otherwise noted, all class diagrams in this thesis depict an implementation view.

2. This subdivision is not explicitly defined by the UML.



^a A type can not contain any attributes.

Figure A.3 Class diagram basic syntax.

The basic syntax of a class diagram is shown in figure A.3. In it a class is depicted by a rectangular box that may contain in addition to its name and stereotype a listing of attributes and operations. Each of these is preceded by an access specifier: a '+' means public, i.e. part of the class' interface, while a '-' denotes a private, and a '#' a protected³ member. When a method is printed in italics it is abstract, which means that no implementation is provided by the class itself, but that instead its derived classes must supply one. When the name of the class is italicized, it is an interface, meaning all its methods are abstract.

Specialization or inheritance is depicted by an arrow pointing from the derived class to its super- or base class. A hollow arrow means an exclusive inheritance, while a filled arrow depicts an inclusive- or virtual-inheritance relationship. When the same base class appears multiple times in a class hierarchy and the inheritance is inclusive, all these base instances are merged into one. On the other hand, when the inheritance is exclusive, each appearance of the base object generates its own methods and attributes.

In figure A.4 the syntax for the associations between classes, interfaces and types is presented. The three different association types are:

Basic association

The semantic relationship between two or more entities that involves connections among their instances.

3. When a member is protected it can, in addition to the class to which it belongs, only be seen by instances of derived classes.

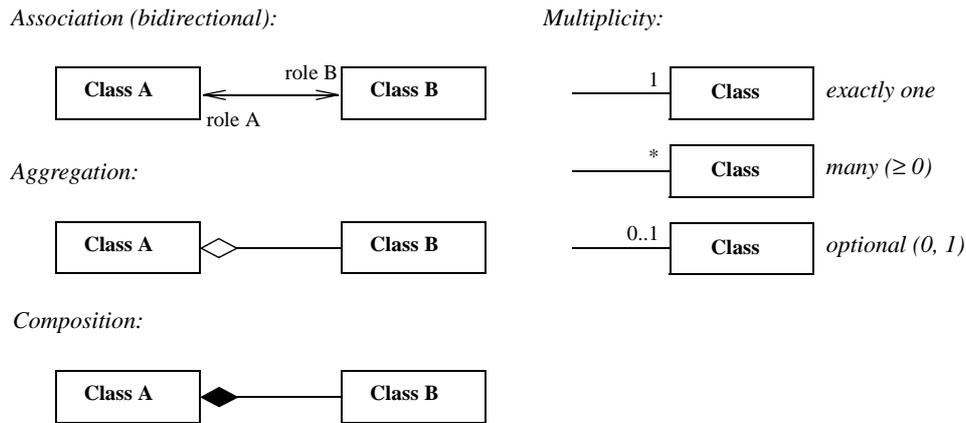


Figure A.4 Class diagram associations syntax.

Aggregation

A special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part.

Composition

A form of aggregation with strong ownership, and coincident lifetimes of the whole and the parts. Parts with non-fixed multiplicity may be created after the composite itself, but once created they share its lifetime.

As can be seen from these definitions, the three types of relationships form a continuous range, and the distinction between them is not always clear.

A.1.3 Interaction Diagrams

Interaction diagrams show how several objects interact with each other. The UML defines three different types of interaction diagrams, viz. sequence, collaboration and activity diagrams. They basically show the same information, but in a different format. Of the three, only the sequence diagrams are used in this thesis (see figure A.5).

Each vertical line represents the life span of an object with time flowing from top to bottom. The object's identifier consists of two parts, separated by a colon. The first part represents its name or identity, while the second part denotes the name of the class of which the object is an instance. When the object's name is omitted, the box represents an anonymous object, i.e. one whose identity is not relevant to the scope of the interaction diagram.

A message is represented by an arrow between the lifelines of two objects. The order in which these messages occur is shown from top to bottom. A message is always labelled with the message name, but the message's arguments and some control information can also be included.

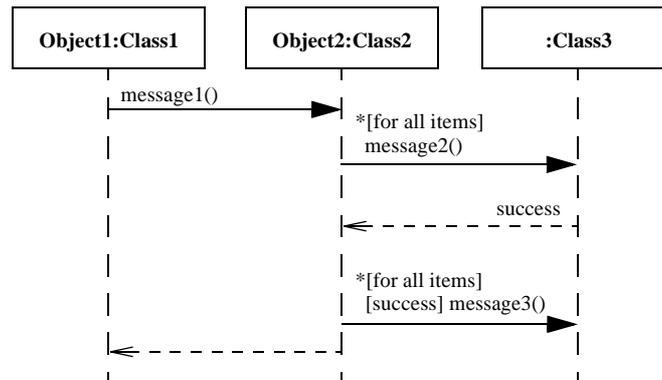


Figure A.5 Sequence diagram syntax.

The first type of control information used in this thesis is the *[condition]*, which indicates when a message is sent: A message is only dispatched when the condition evaluates to true. The second useful type of control info is the *iteration marker*, which shows that a message is sent many times to one or multiple receiver objects. Its syntax is **[type of iteration]*.

Finally, the end of a method can be shown as a dashed arrow, possibly accompanied by a return value.

A.2 Dataview Diagrams

In addition to the diagrams defined by the UML, component diagrams are used to represent dataviews and their interactions. A component is defined as a software entity, which completely decouples its interface from its implementation (see also section 3.4), and a dataview fulfils that requirement: It can be perceived as a back box, completely identified by its inputs and outputs.

A dataview is depicted by a box containing its name and optionally its type. Its inputs are arranged on the left and top edges of the box, its outputs on the right and bottom ones. A

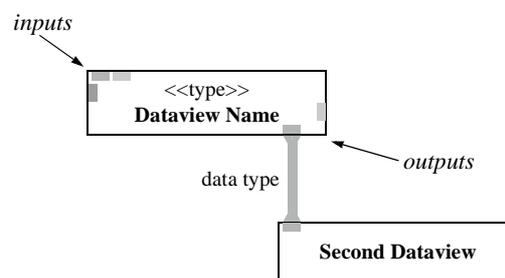


Figure A.6 Dataview diagram syntax [65].

connection between two dataviews is represented by a line, with the data flowing from the output of the first dataview to the input of the second one. The type of the transferred data can be listed alongside the line. Commands to change the state of a dataview flow in the direction opposite the data, i.e. from an input to an output.

By default, each connection is independent, which means that multiple links to the same output retrieve their value and change their state independent of each other. Only when two or more connections share a common line, does the connected output (and its dataview) have the same state.

APPENDIX B | Software Implementation

*The question is whether there will be any dramatic changes.
I mean something even more dramatic than
watching physicists using C and UNIX.*

Paolo Zanella, 1989 CERN School of Computing

The software described in this thesis has been developed on a PC running Windows 95 when it started and Windows NT 4.0 when the final results were obtained¹. As a compiler and development environment, Microsoft Visual C++ versions 4.2 to 6.0 were used. The latest version of the code can be found in the ATLAS repository

<http://atlasinfo.cern.ch/atlas-bin/cvsweb.pl>

In addition to NT, the programs can also be compiled and run on a linux machine using version 3.3f of the KAI compiler. Some statistics of the four packages described in chapter 3 are given below:

| Package | # classes | # lines of code |
|-----------|-----------|-----------------|
| AMBER | 163 | 25,838 |
| DRT | 48 | 10,089 |
| GDL | 109 | 12,824 |
| utilities | 63 | 8,955 |

Table B.1 Basic statistics of the four software packages.

The benchmark machine on which all studies were carried out is a dual Pentium II 300 MHz PC with 256 MB of memory and 8 GB of SCSI disks, running Windows NT version 4.

1. All trademarks and copyrights referred to in this thesis are acknowledged as such.

┌

┐

└

┘

APPENDIX C | Glossary

| | |
|---------------|---|
| AMDB | A TLAS muon database ; ASCII file containing the description of the muon spectrometer. |
| Arve | A TLAS reconstruction and visualization environment ; an object-oriented framework for reconstruction and physics analysis intended to facilitate fast and easy development. It was named after the river Arve, which joins the Rhône in Geneva. |
| ATLAS | A toroidal LHC apparatus ; one of the four future LHC experiments. |
| BIL, BML, BOL | Barrel inner/middle/outer large ; type of MDT chambers used in respectively the inner, middle and outer cylinders of the barrel spectrometer. These large chambers are alternated with similar small chambers (BIS, BMS, BOS). |
| CERN | Conseil Européen pour la recherche nucléaire ; European laboratory for particle physics, located near Geneva, Switzerland. |
| Class | Definition of the structure and behaviour of a set of objects (cf. object). |
| COM | Component object model ; a software architecture that allows applications to be built from binary software components. In its purest form it is simply an interface specification that can also be used as a design pattern. |
| Composite | Pattern in which objects are composed into tree structures to represent part-whole hierarchies. It lets clients treat individual objects and compositions of those objects uniformly. |
| CSC | Cathode strip chamber ; precision chamber used in the inner forward regions of the muon spectrometer (cf. MDT). |
| Dataview | An iterator adaptor, which when connected with other dataviews forms a dataflow network (cf. GDL). |

| | |
|---------------|--|
| DATCHA | D emonstration of ATLAS chamber alignment ; test setup at CERN consisting of the three MDT chambers that make up one tower of the ATLAS muon spectrometer, augmented by several RPC chambers. |
| DRT | D etector reconstruction toolkit ; toolkit of general reconstruction classes such as error points and cones, tracks, track fits and track propagation through a magnetic field. |
| EIL, EML, EOL | E ndcap inner/middle/outer large ; type of MDT chambers used in respectively the inner, middle and outer wheels of the spectrometer endcaps. These large chambers are alternated with similar small chambers (EIS, EMS, EOS). |
| Encapsulation | The hiding of all details of an object that do not contribute to its essential characteristics behind its interface. |
| GDL | G eneric dataview library ; library incorporating the dataflow principle into an object-oriented design, and providing a framework in which data-driven algorithms can be implemented in a straightforward and intuitive way (cf. Dataview). |
| Gismo | Predecessor and core of Arve defining the physics and simulation classes. |
| GUI | G raphical user interface . |
| Interface | A class with only abstract (virtual) methods and no member variables, defining the functionality that is implemented by other classes. |
| Iterator | An object that refers to a value, which in most cases is stored in a container. They come in five different flavours (input, output, forward, bidirectional and random-access), each with its own well-defined functionality. |
| LHC | L arge hadron collider ; proton-proton collider with a centre of mass energy of 14 TeV, which will become operational at CERN in 2005. |
| MDT | M onitored drift tube ; detector elements that make up the precision chambers, which cover most part of the muon spectrometer (cf. CSC). |
| Monostate | A class whose member variables are static. As a result, all objects of that class share the same state. |
| OO | O bject orientation ; a programming paradigm in which state and behaviour of real life entities are modelled together into objects. |
| Polymorphism | Mechanism that allows similar types of objects to respond to the same message in different ways. Run-time polymorphism is implemented through inheritance; at compile-time templates are used. |
| Object | Instance of a class with its own set of data, giving it a unique identity. |

| | |
|-----------------|--|
| Paso | P rovisional analysis skeleton for OO-ATLAS; the current OO framework, which provides a means to use the facilities offered by the reconstruction, simulation, event, detector description and database domains. |
| RASNIK | R ed alignment system of NIKHEF; alignment instrument consisting of a chessboard mask, a lens and a CCD camera, which allows to accurately measure relative displacements of these components. |
| ROA | R egion of activity; general term for a region in space to which a specific aspect under investigation is confined. Also called a region of interest (ROI) or (trigger) road. |
| RPC | R esistive plate chamber; trigger chamber used in the barrel of the muon spectrometer consisting of two gas gaps surrounded by readout strips (cf. TGC). |
| Singleton | A class of which only one object can be created, and which is accessible through its static instance method. |
| STL | S tandard template library; general-purpose C++ library containing a wide variety of data structures and generic algorithms. |
| Template method | Pattern in which a class defines the skeleton of an algorithm while deferring some steps to its subclasses. It lets these subclasses (re)define certain steps of an algorithm without changing the latter's structure. |
| TGC | T hin gap chamber; multi-wire proportional chamber used as the trigger system in the endcaps of the muon spectrometer (cf. RPC). |
| UML | U nified modelling language; notation in which the design of an object-oriented application can be expressed. Its class, package and interaction diagrams are used in this thesis. |
| Visitor | Class that defines an operation to be performed on the elements of a tree structure (cf. composite), with the details of the operation dependent on the exact type of the element. It makes it possible to define a new operation without changing the classes on which it operates. |

References

- [1] Booch G. : *Object-Oriented Analysis and Design with Applications, Second Edition*, Addison-Wesley, 1993.
- [2] Glashow S.L. : *Partial Symmetries of Weak Interactions*, Nucl. Phys. **22** (1961), 579.
- [3] Weinberg S. : *A Model of Leptons*, Phys. Rev. Lett. **19** (1967), 1264.
- [4] Salam A. : *Relativistic Groups and Analyticity*, Proc. 8th Nobel Symposium (1968), 369.
- [5] Goldstone J., Salam A., Weinberg S. : *Broken Symmetries*, Phys. Rev. **127** (1962), 965.
- [6] Higgs P.W. : *Broken Symmetries, Massless Particles and Gauge Fields*, Phys. Lett. **B351** (1964), 132.
- [7] Higgs P.W. : *Spontaneous Symmetry Breakdown without massless Bosons*, Phys. Rev. **145** (1966), 1156.
- [8] Richter-Was E. et al. : *Standard Model and Minimal SuperSymmetric Standard Model Higgs Rates and Backgrounds in ATLAS*, ATLAS Physics Note no. **48**, 17 July 1995.
- [9] LEP-C meeting, 9 November 1999.
- [10] ATLAS Collaboration : *ATLAS Detector and Physics Performance Technical Design Report*, CERN/LHCC 99-14, 25 May 1999.
- [11] The LHC Study Group : *The Large Hadron Collider - Conceptual Design Report*, CERN/AC/95-05 (LHC), 20 October 1995.
- [12] ATLAS Collaboration : *ATLAS Technical Proposal for a General Purpose pp Experiment at the Large Hadron Collider at CERN*, CERN/LHCC 94-43 LHCC/P2, 15 December 1994.
- [13] ATLAS Collaboration : *ATLAS web page*, <http://atlasinfo.cern.ch/Atlas>.
- [14] CMS Collaboration : *The Compact Muon Solenoid - Technical Proposal*, CERN/LHCC 94-38 LHCC/P1, 15 December 1994.
- [15] ATLAS Muon Collaboration : *ATLAS Muon Spectrometer Technical Design Report*, CERN/LHCC 97-22, 5 June 1997.
- [16] ATLAS Software Working Group : *Why ATLAS Software should be Object Oriented and written in C++*, ATLAS Software Note no. **25**, 29 May 1996.

- [17] RD41 Collaboration : *Final Report on MOOSE - Object Oriented Software Development for LHC Experiments*, CERN/LHCC/97-42, 11 June 1997.
- [18] ATLAS Computing Collaboration : *ATLAS Computing Technical Proposal*, CERN/LHCC 96-43, 15 December 1996.
- [19] International Standards Organization : *Programming Languages - C++*, ISO/IEC Publication **14882:1998**, 1998.
- [20] Stroustrup B. : *The C++ Programming Language, Third Edition*, Addison-Wesley, 1997.
- [21] Meyers S. : *Effective C++: 50 Specific Ways to Improve Your Programs and Designs*, Addison-Wesley, 1992.
- [22] Meyers S. : *More Effective C++ : 35 New Ways to Improve Your Programs and Designs*, Addison-Wesley, 1996.
- [23] Barton J.J., Nackman L.R. : *Scientific and Engineering C++*, Addison-Wesley, 1994.
- [24] Fisher S.M., Bos K., Candlin R. : *The ATLAS Software Process*, ATLAS Software Note no. **27**, 14 June 1996.
- [25] ATLAS Domain Interface Group : *Requirements for Atlas Software*, ATL-SW-Requirements-V1, 27 June 1997, <http://www.cern.ch/Atlas/GROUPS/SOFTWARE/00/deliverables/ATL-SW-Atlas-Requirements-V1>.
- [26] Burnett T. : *Arve - a prototype ATLAS reconstruction environment*, Proceedings of the Computing in High-Energy Physics conference, April 1997, <http://home.cern.ch/b/burnett/www/arve> and <http://www.ifh.de/CHEP97/chep97.html>.
- [27] Lakos J. : *Large-Scale C++ Software Design*, Addison-Wesley, 1996.
- [28] Martin R.C. : *Large-Scale Stability*, C++ report **9(2)**, February 1997.
- [29] Hendriks P.J. : *Design of Muon Software*, ATL-SW-Muon-Design-V1, 22 May 1997, <http://www.cern.ch/Atlas/GROUPS/SOFTWARE/00/deliverables/ATL-SW-Muons-Design-V1>.
- [30] Hendriks P.J. : *AMBER Design Overview*, 1 August 1997, <http://home.cern.ch/~patrickh/Projects/amber/Amber.html>.
- [31] Ball S., Crawford J. : *Monostate Classes: the Power of One*, C++ Report **9(5)**, May 1997.
- [32] Gamma E. et al. : *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.
- [33] Chevalier L. et al. : *AMDB_SIMREC: A Structured Database for the ATLAS Spectrometer Simulation Program*, ATLAS Muon Note no. **148**, 15 April 1997, http://atlasinfo.cern.ch/Atlas/GROUPS/MUON/AMDB_SIMREC/amdb_simrec.html.
- [34] ATLAS Graphics Group : *Design of Graphics Software*, ATL-SW-Graphics-Design-V1, 17 April 1998, <http://www.cern.ch/Atlas/GROUPS/SOFTWARE/00/deliverables/>

- ATL-SW-Graphics-Design-V1.
- [35] Bos K., Clifft R., Hendriks P.J., Poppleton A. : *A Track Class for ATLAS*, 14 July 1999, <http://www.cern.ch/Atlas/GROUPS/SOFTWARE/00/domains/Reconstruction/packages.html>.
- [36] Melanson H. : *Cuts Package*, D0 software distribution, <http://www-d0.fnal.gov/d0dist/dist/packages/cuts/v01-00-00/doc>.
- [37] Myers N. : *A New and Useful Template Technique: "Traits"*, C++ Report **7(6)**, June 1995.
- [38] Martin R.C. : *Acyclic Visitor*, Proceedings of the Joint Pattern Languages of Programs conference, September 1996, <http://www.oma.com/Publications/publications.html>.
- [39] Lavrijsen W. : Private communication;
BrockSchmidt K. : *How COM Solves the Problems of Component Software Design, Part II*, Microsoft Systems Journal **11(6)**, June 1996.
- [40] Bos K., Hendriks P.J. : *Design of Magnetic Field Software*, ATL-SW-MagneticField-Design-V1, 12 March 1999, <http://www.cern.ch/Atlas/GROUPS/SOFTWARE/00/deliverables/ATL-SW-MagneticField-Design-V1>.
- [41] Musser D.R., Saini A. : *STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library*, Addison-Wesley, 1996.
- [42] Candlin D., Candlin R. : *PASO: A Provisional Analysis Skeleton for OO-ATLAS*, <http://www.cern.ch/Atlas/GROUPS/SOFTWARE/00/applications/Paso>.
- [43] Application Software Group, Computing and Networks Division, CERN : *Geant 3 - detector description and simulation tool*, CERN Program Library Long Writeup **W5013**, 1993.
- [44] Artamonov A. et al. : *DICE-95*, ATLAS Software Note no. **14**, 22 March 1995.
- [45] Ambrosini G. et al. : *A proposal for the ATLAS Level-2 Muon Trigger in the barrel region*, ATLAS Muon Note no. **61**, 15 January 1995.
- [46] Palamara O., Petrera S. : *Pattern recognition at the second level muon trigger in the ATLAS barrel region*, ATLAS DAQ Note no. **17**, 27 December 1994.
- [47] Viehhauser G. : *Detector Physics of the ATLAS Precision Muon Chambers*, Ph.D. thesis, Technical University of Vienna, 1996.
- [48] Riegler W. : *MDT Efficiency, Double Track Separation*, ATLAS Muon Note no. **173**, 5 October 1997.
- [49] Van Der Graaf H. et al. : *First system performance experience with the ATLAS high-precision muon drift tube chambers*, Nucl. Instrum. Methods Phys.Res. **A419** (1998 2-3), 336.
- [50] Groenstege H. : *The RASNIK/CCD 3D alignment system*, ATLAS Muon Note no. **63**, 15 December 1994.

- [51] Bringle M.P. : *Cosmic Ray Muon Monte Carlo Generator Using the Hemisphere Method*, BaBar Note no. **163**, 19 September 1994.
- [52] Linde F. et al. : *μ -Track Reconstruction in DATCHA*, ATLAS Muon Note no. **220**, 31 January 1998.
- [53] Creti P. et al. : *Testbeam results from the Calypso MDT chamber*, ATLAS Muon Note no. **196**, 24 October 1997.
- [54] Vreeswijk M. et al. : *Global alignment of MDT chambers in DATCHA-CERN*, ATLAS Muon Note no. **246**, 1998.
- [55] Linde F., Woudstra M. : *Geometry Reconstruction using Muon Tracks in DATCHA CERN*, ATLAS Muon Note no. **250**, 28 September 1998.
- [56] Woudstra M. : Ph.D. thesis, in preparation.
- [57] Clift R., Poppleton A. : *IPATREC: inner detector pattern-recognition and track-fitting*, ATLAS Software Note no. **9**, 9 June 1994.
- [58] Chevalier L. et al. : *BMAGATLAS - The ATLAS Magnetic Field Database*, http://atlasinfo.cern.ch/Atlas/GROUPS/MUON/magfield/mag_database.html.
- [59] Particle Data Group : *Review of Particle Properties*, Phys. Rev. **D54**, 1 July 1996.
- [60] ATLAS Level-1 Trigger Group : *Level-1 Trigger Technical Design Report*, ATLAS TDR-12, 20 August 1998.
- [61] Nisati A. : *Alignment of Muon Chambers with $Z \rightarrow \mu^+ \mu^-$ Events*, ATLAS Muon Note no. **9**, 30 August 1992.
- [62] Aleksa M. : *Absolute Mass Scale Calibration using $Z \rightarrow \mu^+ \mu^-$* , ATLAS Muon Note no. **99-001**, 30 April 1999.
- [63] Linossier O, Poggioli L. : *$H^0 \rightarrow ZZ^* \rightarrow 4l$ channel in ATLAS - Signal reconstruction and reducible backgrounds rejection*, ATLAS Physics Note no. **101**, 11 April 1997.
- [64] Booch G., Jacobson I., Rumbaugh J. : *The Unified Modeling Language Reference Manual*, Addison-Wesley, December 1998.
- [65] Tuura L.A. : *Component Based Reconstruction Prototype for ATLAS Electromagnetic Calorimeters*, Master's Thesis, Helsinki University of Technology, 21 January 1997, <http://home.cern.ch/1/1at/www/exports/thesis/thesis.pdf>.

Summary

Elementary particle physics is the study of the fundamental building blocks of nature and the interactions between them. All matter is constructed out of quarks and leptons, which are subject to one or more of the four fundamental forces, viz. gravity, electromagnetism and the weak and strong interactions. Except for gravity, these forces are united into a single theory called the Standard Model. To a large extent, it predicts with great accuracy the experimental data that is available today.

The definition of mass in the Standard Model is based on the spontaneous breaking of the symmetry between electromagnetism and the weak interaction through the so-called Higgs mechanism. This results in the prediction of a new particle, the Higgs boson, which with its predicted mass between 109 and 215 GeV has remained beyond the reach of all current experiments. Its detection is therefore one of the main goals of a new collider and corresponding detectors that are currently being developed at the European Laboratory for Particle Physics (CERN) near Geneva, Switzerland.

In this thesis the reconstruction of muons in one of these detectors called ATLAS has been studied. The development of the reconstruction software has been performed using the full potential of object-orientation (OO) and C++. The core of this software is formed by an ATLAS specific program called AMBER. It builds the detector description, reads in the events, defines the reconstruction algorithm and outputs the results to a graphics window or to a file.

In the spirit of good OO, all classes that are not specific to the reconstruction of events in the muon spectrometer of the ATLAS detector have been separated off into several general-purpose packages. The most important ones are the Detector Reconstruction Toolkit (DRT) and the Generic Dataview Library (GDL). The first is basically that what its name suggests, viz. a toolkit of general reconstruction classes such as tracks, fitting algorithms and the propagation of tracks through a magnetic field. In addition to AMBER, it is also used by other ATLAS packages and even by the D0 software.

The second package, the GDL, builds on top of the iterator concept introduced by the Standard Template Library. Its dataviews are iterator adaptors that can be connected together in any way the user sees fit to form complete dataflow networks. Moreover, its genericness makes it completely independent of the type of data it is handling. In addition to the architecture, the GDL also provides a set of predefined dataviews to handle some of the most common tasks such as filtering, sorting and the creation of combinatorials.

The reconstruction algorithm inside AMBER is completely defined in terms of these two packages. It starts with the creation of roads from the hits in the trigger chambers, continues with the pattern recognition of the precision hits that fall inside these roads, and ends with a global fit through both the trigger and precision hits. Its performance has been evaluated in three different environments of increasing complexity.

The first of these consists of a stand-alone MDT chamber, and is used to analyse the pattern recognition for different detector efficiencies and background conditions. Under nominal conditions the track-segment reconstruction efficiency is 99.8% both for the 2×3 and 2×4 chambers used in the ATLAS detector, while the fake-track rate remains well below the 1%. In addition, the reconstructed single-tube resolution is identical to the value that was used in the simulation of the events. In a worst-case scenario consisting of tube inefficiencies of 10% and background levels 5 times the nominal value, the reconstruction efficiency deteriorates to around 98.2%, which corresponds to a global inefficiency of 5%. Simultaneously, the fake-track rate rises to 2% for the 2×4 chamber and to 3% for the other type of chamber, which converts to a 8% global rate before any segment matching criteria are applied.

In the second environment, that of the DATCHA test setup, the combined trigger and precision-chamber reconstruction in a full barrel tower of the muon spectrometer is evaluated under real-life conditions. After calibration, track segments are reconstructed with accuracies comparable to those found in the simulated stand-alone chambers. Based on the hit residuals of 52, 59 and 67 μm of the three MDT chambers, single-tube resolutions between 68 and 86 μm can be derived. By combining the segments, a comparison can be made with the measurements of the RASNIK alignment systems. The observed difference in sagitta of 15 μm is well below the design target of 30 μm and is limited only by statistics.

As the final test, AMBER is used to reconstruct events in the full muon spectrometer. The obtained resolutions agree very well with theoretical predictions. For example, a p_T -resolution of 1.5% for 10 GeV muons has been measured, which subsequently rises to about 6% at 1 TeV. Also, the measured Z and Higgs mass resolutions of respectively 2.7 and 2.5 GeV (for a Higgs mass of 130 GeV and with the Z-mass constraint applied) agree well with previously obtained results.

The obtained reconstruction efficiencies are however well below what is desirable of the offline muon reconstruction. This is in part due to the various selection criteria used by the program, which are designed to optimize the accuracy of the reconstruction. But mostly, it is caused by the fact that the global track fit does not take any multiple scattering into account. This is especially a problem when a muon crosses one of the chamber or magnet support materials. Simulation runs without these structures present show that efficiencies of 96% and 97% for 10 GeV and 50 GeV muons respectively can be obtained. This still falls a somewhat short of other ATLAS results, which means that further development of the reconstruction algorithm and its fit are needed.

By using the information from the inner detector and calorimeter, the reconstruction of the Higgs decay into four muons can be improved, and the results of the other two final states of the $H \rightarrow 4l$ channel can be added to it. With the help of a lepton isolation and an impact parameter cut the background to this channel can be substantially suppressed, leading to high significances for the intermediate (130 to 160 GeV) and high (180 to 700 GeV) Higgs masses. For the remaining mass ranges other channels exist, which means that the ATLAS detector will be capable of finding the Higgs boson with a significance well above 5σ after either three years of low-luminosity running or after just one year at high luminosity.

Samenvatting

Elementaire deeltjesfysica is de studie van de fundamentele bouwstenen van de natuur en van hun onderlinge interacties. Alle materie is opgebouwd uit quarks en leptonen welke onderhevig zijn aan een of meer van de vier fundamentele krachten, namelijk de zwaartekracht, elektromagnetisme en de zwakke en sterke kernkrachten. Deze laatste drie zijn verenigd in één theorie genaamd het Standaard Model. Deze is voor het grootste deel in staat om met een hoge nauwkeurigheid de experimentele gegevens die vandaag de dag beschikbaar zijn te verklaren.

De definitie van massa in het Standaard Model is gebaseerd op de spontane breking van de symmetrie tussen elektromagnetisme en de zwakke kernkracht door middel van het zogenaamde Higgs mechanisme. Dit resulteert in de voorspelling van een nieuw deeltje, het Higgs boson, welke met zijn voorspelde massa tussen de 109 en 215 GeV buiten het bereik van alle huidige experimenten is gebleven. De waarneming van de Higgs is dan ook een van de belangrijkste redenen voor de huidige ontwikkeling van een nieuwe versneller en bijbehorende detectoren op het Europees Laboratorium voor Deeltjesfysica (CERN) bij Genève, Zwitserland.

In dit proefschrift is de reconstructie van muonen in één van deze detectoren genaamd ATLAS bestudeerd. De ontwikkeling van de reconstructie software heeft plaatsgevonden gebruik makend van alles wat object oriëntatie (OO) en C++ te bieden hebben. De kern van de software wordt gevormd door een ATLAS specifiek programma genaamd AMBER. Het is verantwoordelijk voor de bouw van de detector beschrijving, voor het inlezen van de events, het definieert het reconstructie algoritme en verzorgt de uitvoer van de resultaten naar een grafisch scherm of naar een bestand.

Volgens de richtlijnen van een goede OO ontwikkeling zijn alle klassen welke niet specifiek zijn voor de reconstructie van events in de ATLAS muon spectrometer afgesplitst in een aantal algemene pakketten. De meest belangrijke zijn de Detector Reconstruction Toolkit (DRT) en de Generic Dataview Library (GDL). De eerste is precies wat zijn naam al zegt, namelijk een gereedschapskist vol met algemene reconstructie klassen zoals sporen, fit algoritmes en de propagatie van sporen door een magneetveld. Naast AMBER wordt het ook gebruikt door andere ATLAS pakketten en zelfs door de DO software.

Het tweede pakket, de GDL, borduurt voort op het iterator concept van de Standard Template Library. Zijn dataviews zijn iterator adapters die in een willekeurige volgorde met elkaar kunnen worden verbonden zodat hele dataflow netwerken ontstaan. Bovendien is het door zijn generiekheid volledig onafhankelijk van het type data dat het verwerkt. Naast de architectuur levert de GDL ook een aantal standaard dataviews voor het uitvoeren van de meest voorkomende taken zoals het filteren en sorteren van data en het creëren van combinatorials.

Het reconstructie algoritme binnen AMBER is volledig gedefinieerd in termen van deze twee pakketten. Het begint met het aanwijzen van interessegebieden uit de hits in de trigger kamers, gevolgd door de patroon herkenning van de precisie hits die in die gebieden vallen,

en het eindigt met een globale fit door beide type hits. De prestaties van dit algoritme zijn geëvalueerd in een drietal verschillende omgevingen van oplopende complexiteit.

De eerste hiervan wordt gevormd door een enkele MDT kamer en dient ervoor om de patroon herkenning te analyseren bij verschillende detector efficiënties en achtergrond condities. Onder nominale omstandigheden is de spoorsegment-reconstructie efficiëntie gelijk aan 99.8% voor zowel de 2×3 als de 2×4 kamers terwijl de kans op de reconstructie van valse sporen ruim onder de 1% blijft. Daarnaast is de gereconstrueerde resolutie van een MDT buis gelijk aan de waarde gebruikt in de simulatie. In het ergste geval van een 10% inefficiëntie en een achtergrond niveau gelijk aan 5 maal de nominale waarde daalt de reconstructie efficiëntie tot ongeveer 98.2%, wat overeenkomt met een globale inefficiëntie van 5%. Gelijktijdig stijgt het aantal foutief gereconstrueerde sporen tot 2% in de 2×4 kamer en tot 3% in het andere type. Zonder toepassing van enige criteria op het goed op elkaar aansluiten van de verschillende spoorsegmenten is dit gelijk aan een 8% kans op het vinden van een onjuist globaal spoor.

In de tweede omgeving, dat van de DATCHA test opstelling, wordt de gecombineerde trigger en precisie-kamer reconstructie in een volledige toren van de muon spectrometer in de praktijk geëvalueerd. Na calibratie bereikt de spoorsegment reconstructie een nauwkeurigheid die overeenkomt met die in de gesimuleerde events. Gebaseerd op de hit residuals van 52, 59 en 67 μm gevonden in de drie MDT kamers kan een resolutie per buis van tussen de 68 en 86 μm worden afgeleid. Daarnaast kan door het combineren van de segmenten een vergelijking worden gemaakt met de metingen van de RASNIK uitlijnsystemen. Het waargenomen verschil in sagitta van 15 μm ligt ruim onder de maximaal toelaatbare waarde van 30 μm , en kan nog worden verbeterd door het vergaren van meer statistiek.

Als een laatste test is AMBER gebruikt om events in de volledige muon spectrometer te reconstrueren. De verkregen resoluties komen zeer goed overeen met de theoretische voorspellingen. Zo is bijvoorbeeld een p_T -resolutie van 1.5% voor 10 GeV muonen gemeten welke vervolgens toeneemt tot 6% bij 1 TeV. Daarnaast zijn ook de gemeten Z en Higgs massa resoluties van respectievelijk 2.7 en 2.5 GeV (voor een Higgs massa van 130 GeV en met de Z-massa constraint toegepast) in overeenstemming met eerder behaalde resultaten.

De verkregen reconstructie efficiënties zijn daarentegen veel lager zijn dan wat wenselijk wordt geacht voor de offline muon reconstructie. Dit komt gedeeltelijk doordat de selectie criteria van het programma gericht zijn op het optimaliseren van de nauwkeurigheid van de reconstructie. De belangrijkste reden is echter dat er geen multiple scattering in de globale fit wordt meegenomen. Dit is vooral een probleem wanneer een muon één van de kamer- of magneetstructuren doorkruist. Simulaties zonder dit materiaal laten zien dat efficiënties van respectievelijk 96% en 97% voor 10 GeV en 50 GeV muonen gehaald kunnen worden. Dit schiet nog steeds iets te kort vergeleken met andere ATLAS resultaten, vandaar dat een verdere ontwikkeling van het reconstructie algoritme en de fit nodig zijn.

Door gebruik te maken van de informatie in de inner detector en de calorimeter kan de reconstructie van het Higgs verval naar 4 muonen verbeterd worden. Daarnaast kunnen dan ook de resultaten van de andere twee eindtoestanden van het $H\rightarrow 4l$ kanaal er aan toegevoegd worden. Door eisen te stellen aan de isolatie van de leptonen en aan de spreiding in de impact parameters kan de achtergrond van dit kanaal voldoende worden onderdrukt om hoge significanties te halen voor gemiddelde (130 tot 160 GeV) en hoge (180 tot 700 GeV) Higgs massa's. Voor de overige massa's bestaan andere kanalen waardoor de ATLAS detector in staat zal zijn om na drie jaar draaien bij lage luminositeit of al na 1 jaar bij hoge luminositeit het Higgs boson te vinden met een significantie ver boven de 5σ .

Acknowledgements

Over the past four years many people have helped me with the work described in this thesis. First and foremost has been the support by Kors Bos, my supervisor and friend. He helped me to get started in the world of computing and continuously believed in what we were doing. I very much liked his enthusiasm, his embrace of new technologies and even the many hours he talked about golf.

I would also like to thank Frank Linde who during our work on DATCHA has greatly helped me in improving both my reconstruction algorithms and my analysis of the data. Also, his detailed feedback on this thesis has been very valuable. I would also like to thank Martin Woudstra and Marcel Vreeswijk for helping me come to grips with the hardware and online software of the DATCHA setup. Without them I would never have been able to figure out what all those bits meant.

I would like to thank the ATLAS software community and especially the ‘dead moose’ for the many (un)productive discussions we had, and of course for our very pleasant traditional software-week diners. In particular, I am in a large way indebted to Toby Burnett. Without his Arve framework my work would have been much more difficult. His support during the time I was learning how to use Arve, and the speed with which he updated it and fixed the occasional bugs were very much appreciated. During my time at CERN I enjoyed working with Lassi Tuura. His help in mastering OO design and C++ has been invaluable. I am also very grateful to Alan Poppleton who despite being constantly buried in work took the time to explain to me the intricate details of track fitting.

At NIKHEF I was very fortunate to be able to share an office with Onne Peters, Dies Köper, Paul Balm and Michiel Vogelvang. Their company has been fun, inspiring, insightful and exhausting and has cost me many hours typing away in ICQ. It was time well spent.

En als laatste wil ik mijn ouders bedanken. Zonder hun volledige en onaflaatbare ondersteuning en vertrouwen was ik nooit zo ver gekomen.

