

TEL AVIV UNIVERSITY

RAYMOND AND BEVERLY SACKLER
FACULTY OF EXACT SCIENCES
SCHOOL OF PHYSICS & ASTRONOMY



אוניברסיטת תל-אביב

הפקולטה למדעים מדויקים
ע"ש ריימונד ובברלי סאקלר
בית הספר לפיזיקה ואסטרונומיה

Hough Transform Track Reconstruction in the Cathode Strip Chambers in ATLAS

by

Nir Amram

Thesis submitted towards the degree of M.Sc. in physics

Under the supervision of **Prof. Erez Etzion**

CERN-THESIS-2008-062
19/03/2008



Tel-Aviv University
Raymond and Beverly Sackler
Faculty of Exact Sciences

July 2008

Abstract

The world's largest and highest energy particle accelerator, the Large Hadron Collider (LHC), will collide two highly energetic proton beams in an attempt to discover a wide range of new physics. Among which, the primary ambitions are the discovery of the Higgs boson and supersymmetric particles. ATLAS, one of its primary particle detectors, was designed as a general-purpose detector covering a broad range of energies and physical processes. A special emphasis on accurate muon tracking has led the ATLAS collaboration to design a stand-alone Muon Spectrometer, an extremely large tracking system extending all the way around the detector. Due to its immense size and range, parts of the spectrometer were designed to withstand a high rate of radiation, sifting the muon signals from the rest of the signals (primarily neutrons and photons).

The Cathode Strip Chambers (CSCs) are special multiwire proportional chambers placed in the high η region of the Muon Spectrometer, where flux of background particles is highest. Their purpose is to efficiently filter out the background particle, tracking only the muons traversing it with high degree of accuracy. In order to do that, this special algorithm was designed using a novel modification of the Hough Transform. This thesis will detail the key elements of this algorithm, how it is used for better muon track detection and parameterization, and give a preliminary evaluation of the performance of this algorithm.

Table of Contents

1	Introduction.....	9
2	LHC Physics	10
3	The ATLAS detector in the LHC.....	13
3.1	The Inner Detector	14
3.2	The Calorimeters.....	15
3.3	The Magnet System	16
3.4	The Muon Spectrometer system	17
3.4.1	Resistive Plate Chambers (RPCs).....	20
3.4.2	Thin Gap Chambers (TGCs).....	21
3.4.3	Monitored Drift Tubes (MDTs).....	24
3.4.4	Cathode Strip Chambers (CSCs)	26
4	Track Reconstruction in ATLAS	30
4.1.1	iPatRec	32
4.1.2	xKalman.....	32
4.1.3	MOORE and MUID.....	32
4.1.4	MuonBoy	33
4.1.5	MuGirl	33
4.2	MDT Tracking	34
5	Atlas Software Environment (ATHENA).....	38
5.1	Simulation	39
5.2	The Event Data Model	40
5.3	Geometry and GeoModel.....	41
5.3.1	Global Coordinate Frame.....	41
5.3.2	Local Coordinate Frame	42
5.3.3	Chamber Coordinate Frame.....	43
6	CSC Tracking.....	45
6.1	Hough Transform.....	45
6.2	Approach.....	46
6.3	The Tracking Algorithm	46
6.3.1	Readout	49
6.3.2	Activity Detection.....	49
6.3.3	Crude Track-Finding.....	50
6.3.4	Clusterization	52
6.3.5	Fine Track-Fitting	57
6.3.6	Storage	58
6.4	Software Structure	58
6.4.1	The converters.....	58
6.4.2	Packages structure in ATHENA	59
6.4.3	Packages description.....	60
6.5	Analysis.....	61
7	Summary	72
8	Appendices.....	73
8.1	CSCs	73
8.1.1	Software Flow	73
8.1.2	Software Usage	76
8.1.3	Class Descriptions.....	83
8.2	Thin Gap Chambers	93
8.2.1	Testbench	93
8.2.2	Hole-tracking software	93

8.2.3 TightTGC.....	97
10 References.....	108

Table of Figures

Figure 1: Sensitivity for the discovery of a SM Higgs boson in the LHC experiments as a function of the Higgs mass.	11
Figure 2: Schematic illustration of particle signatures in the four sub-detector layers of ATLAS.....	13
Figure 3: Schematic view of the ATLAS detector.....	14
Figure 4: Schematic view of the ATLAS Inner Detecor.	15
Figure 5: Schematic view of the ATLAS Electromagnetic and Hadronic Calorimeters	16
Figure 6: Schematic view of the ATLAS magnet system.....	17
Figure 7: Schematic view of the ATLAS Muon Spectrometer System.	18
Figure 8: A schematic side-view of the ATLAS Muon Spectrometer system, showing the different chamber technologies.	19
Figure 9: Schematic structure of Resistive Plate Chambers.	21
Figure 10: Schematic view of a Thin Gap Chamber.....	22
Figure 11: Top view of a T8 TGC chamber.	23
Figure 12: The efficiency map of the doublet U08F3I-529.8.....	24
Figure 13: Charge avalanche in a single MDT tube.	25
Figure 14: A view of a MDT chamber.....	26
Figure 15: Schematic View of CSC precision strips.	27
Figure 16: 3D view of the CSC chamber layout.....	27
Figure 17: Bipolar shape of a signal in a single CSC strip.	28
Figure 18: Tracking reconstruction chain.	30
Figure 19: Reconstruction of a simulated event in ATLAS as shown by Persint event viewer.....	31
Figure 20: A schematic example of a muon track reconstruction in the MDT chamber.	34
Figure 21: Geometrical representation of the MDT tube scenarios.....	37
Figure 22: Schematic view of the ATLAS detector oriented in the global coordinate system.	42
Figure 23: Illustration of the local to global transformations.	43
Figure 24: An illustration of the chamber coordinate frame.....	44
Figure 25: Illustration of the discrete Hough Transform.	45
Figure 26: Flowchart of the CSC reconstruction algorithm.....	48
Figure 27: Testbeam bipolar signals of CSC strips.	50
Figure 28: Hough parameter space for the four CSC layers, taken from the CSC_DHough program.....	51
Figure 29: A view of the muon charge cluster in the four CSC layers.	52
Figure 30: Mathieson Distribution.....	53
Figure 31: Data converters usage in the track finding algorithm.....	59
Figure 32: CSC and MDT Reconstruction packages structure in the ATHENA framework.	60
Figure 33: Accepted and rejected strips for the η and ϕ planes of the CSC chambers.	62
Figure 34: Hit acceptance rates after the Hough-Transform, averaged over strip number.	63
Figure 35: Change in acceptance rates for η (right) and ϕ (left) planes, using various parameter cuts.	64
Figure 36: Efficiency as a function of fake rate for η (right) and ϕ (left) planes.....	66

Figure 37: Distribution of the maximum charge in a cluster.	67
Figure 38: ROOT Event Viewer of a CSC chamber.....	68
Figure 39: Fit residuals in each of the CSC layers.....	69
Figure 40: Total residuals of the reconstructed CSC tracks.	70
Figure 41: Extrapolated CSC and ID tracks to the MDT middle station.....	71
Figure 42: Global CSC Reconstruction software flow.	73
Figure 43: Workflow of the Crude Track Finding stage.....	74
Figure 44: Workflow of the Clusterization stage.....	74
Figure 45: Workflow of the Fine Track Fitting stage.	75
Figure 46: Workflow of the Storage stage.....	75
Figure 47: The testbench schematic structure.....	93
Figure 48: An efficiency mask of a TGC chamber.....	95
Figure 49: A hole classification plot of a TGC chamber.	95
Figure 50: TightTGC system.	97
Figure 51: TGC-Lite readout card.	99
Figure 52: TGC Lite switch system.	100
Figure 53: TightTGC main screen	103
Figure 54: TGC-Lite hit count and multiplicity in two threshold settings.	105
Figure 55: TightTGC hit count and multiplicity for noisy/missing channels.....	106
Figure 56: TightTGC channel hit count and multiplicity for cross-talk channels.	107

1 Introduction

The Large Hadron Collider (LHC) is a particle accelerator and collider being built in the European Organization for Nuclear Research (CERN), near Geneva, Switzerland. It will collide two 7 TeV proton beams in the purpose of broaden understanding of the physical processes governing particle interactions in high energies.

The LHC spans across a circular 27km underground tunnel at depths ranging from 50 to 175m. It will house two high-luminosity experiments, ATLAS [1] and CMS, two low-luminosity experiments, LHCb and TOTEM, and one dedicated ion experiment, ALICE. Due to the high luminosity needed for both ATLAS and CMS, the peak design-luminosity of LHC is set to $10^{34} \text{ cm}^{-2}\text{s}^{-1}$.

ATLAS and CMS are both general-purpose detectors that will explore pp collisions in depth. LHCb will focus on b-physics, using one proton beam to hit a fixed target. Pb-Pb nuclei collisions will be studied in ALICE at a center mass energy of 5.5TeV per nuclei.

The Tel-Aviv University High Energy Group is collaborating in one of the two pp experiments, the ATLAS experiment. This experiment has a great discovery potential for new physics. For example a Standard Model Higgs boson can be discovered over the full kinematical allowed energy range.

ATLAS (Chapter 2), a bulky cylindrical detector, is 44m long and 22m in diameter, weighing 7000 tons. It is comprised of four specialized sub-detectors – Inner Detector (Chapter 3.1), Electromagnetic Calorimeter, Hadron Calorimeter (Chapter 3.2) and Muon Spectrometer (Chapter 3.4), each targeting a different type of particles.

In this thesis I will introduce my work on track reconstruction in the Cathode Strip Chambers (Chapter 3.4.4) in the Muon Spectrometer and its place in the global tracking environment of ATLAS.

2 LHC Physics

The Standard Model (SM) [2] provides a very successful description of interactions of the constituents of matter down to the smallest distances (10^{-18} m) and up to highest energies ($\sim 2\text{TeV}$) accessible to current experiments. It is based on quantum field theory in which interactions of spin $1/2$, point-like fermions are mediated by spin 1 gauge bosons.

The gauge-theory part of the SM has been well tested, but in spite of many indirect evidences there is no direct proof either for or against the Higgs [3] mechanism for electroweak symmetry breaking. In the SM all masses are tied to the mass scale of the Higgs sector; however the model does not provide guidance for the Higgs mass. Present experimental results interpreted in term of the SM Higgs, point to the mass of the Higgs boson in the range 160 to 200 GeV. The experimental observation of one or several Higgs bosons will be fundamental for a better understanding of the mechanism of electroweak symmetry breaking.

In the SM, one doublet of scalar field is assumed for symmetry breaking, leading to the existence of one neutral scalar Higgs particles, H. In supersymmetric theories, the Higgs sector is extended to contain at least two doublets of scalar fields. In the minimal version, the so-called minimal supersymmetric SM (MSSM) model, there are five physical charged Higgs particles: CP even, h, H, one CP odd, A, and two charged H^\pm . Two parameters, which are generally chosen to be the mass of the A Higgs, m_A and $\tan \beta$, the ratio between the vacuum expectation values of the Higgs doublets, determine the structure of the Higgs sector at tree level.

The dominant production mechanism of a light Higgs boson at LHC energies is gluon-gluon fusion, which proceeds via heavy quark loops.

The overall sensitivity for the discovery of a SM Higgs boson is shown in Figure 1 for various decay channels, assuming an integrated luminosity of 100 fb^{-1} .

The decay channel $H \rightarrow ZZ^* \rightarrow 4l$ provides a rather clean signature in the mass range between $\sim 120 \text{ GeV}$ and $2m_Z$, above which the gold-plated channel with two real Z bosons in the final state opens up. Both electrons and muons are considered in the final state, thus yielding $eeee$, $ee\mu\mu$ and $\mu\mu\mu\mu$ event topologies. While production channels with W bosons are also of interest, they usually provide lower sensitivity.

If the SM Higgs boson were to be discovered at LHC, its mass, m_H , would be measured with a precision of 0.1% for $m_H < 400 \text{ GeV}$ and of

0.1-1% for $400 < m_H < 700$ GeV. The Higgs boson width can be precisely determined for masses above 200 GeV using the $H \rightarrow ZZ \rightarrow ll\nu\nu$ channel.

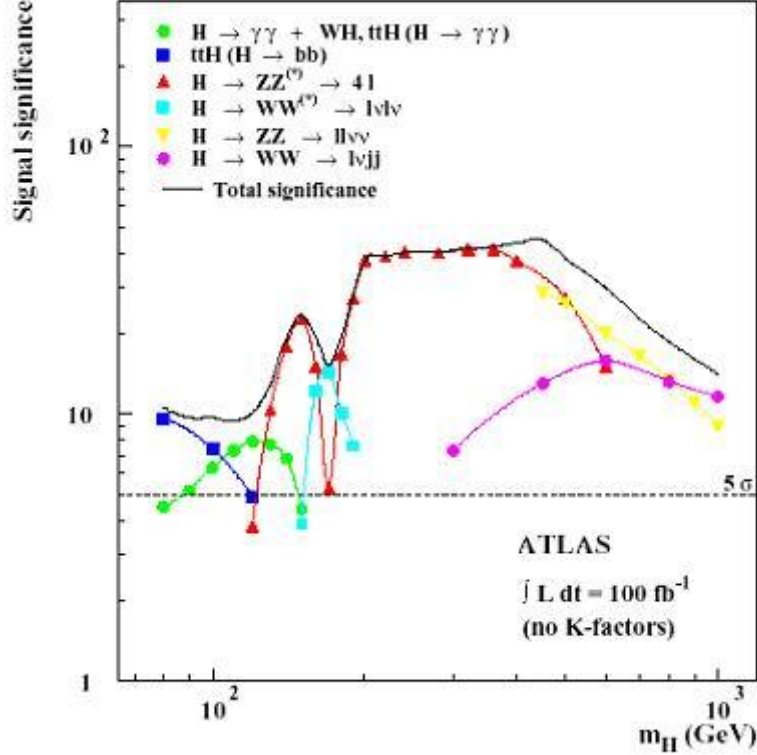


Figure 1: Sensitivity for the discovery of a SM Higgs boson in the LHC experiments as a function of the Higgs mass.

The signal significance, assuming an integrated luminosity of 100 fb^{-1} , is plotted in terms of standard deviations, for individual channels (different symbols as defined in the figure), as well as for the combination of all channels (continuous black line) (See ref. [5]).

The capability of LHC experiments to detect MSSM Higgs bosons has been studied in depth over the last few years [4]. It is usually assumed that the supersymmetric particles are heavy enough, so that the decay of the Higgs bosons proceeds through channels involving the known particle spectra. In the MSSM, various decay modes accessible also in the case of the SM Higgs boson, are predicted such as $h \rightarrow \gamma\gamma$, $h \rightarrow b\bar{b}$, $H \rightarrow ZZ^* \rightarrow 4l$. In addition, some channels such as $H/A \rightarrow \tau\tau$ and $H/A \rightarrow \mu\mu$ are strongly enhanced if $\tan \beta$ happens to be large. Complete coverage of the region will be possible at LHC. Over a considerable fraction of the parameter space, at least two channels are accessible and/or more than one Higgs bosons can be observed. In most cases, the experiments will be capable of distinguishing between a SM and an MSSM Higgs boson.

If supersymmetry (SUSY) indeed exists at the electroweak scale, then the SUSY cross-section is dominated by gluinos and squarks production

[5], and the cross-sections are expected to be large. Gluinos and squarks decay sequentially into the lightest supersymmetric particle (LSP) (which may decay further, if the supersymmetric quantum number, the R-parity is violated). These decay chains lead to a variety of signatures in the final state involving multiple jets, leptons, photons, heavy flavors, W and Z bosons, as well as missing energy. The combination of a large production cross-section and distinctive signatures makes it easy to separate SUSY from the SM background. Therefore, it is conceivable that the main challenge will not be to discover SUSY, but to separate the many SUSY processes that occur and to measure the masses and other properties of the SUSY particles. In most cases, the backgrounds from other SUSY events dominate over the reducible SM backgrounds.

In summary, if a Higgs boson, with $m_H < 1$ TeV, exists, it will be discovered at LHC. The same is true of supersymmetric particles.

The properties of the production and decay mechanisms define the required performance of the detector, presently at the construction stage.

The LHC open a new scale of energy and therefore we should keep our eyes and minds open for all sort of new surprises.

3 The ATLAS detector in the LHC

The ATLAS detector is now towards the end of its construction phase and is scheduled to be operational by May 2008. It consists of a series of ever-larger concentric cylinders around the interaction point where the proton beams from the LHC collide. It can be divided into four major parts: the Inner Detector, the calorimeters, the muon spectrometer and the magnet systems. Each of these is in turn made of multiple layers. The detectors are complementary: the Inner Detector tracks charged particles precisely, the calorimeters measure the energy of easily stopped particles, and the muon system makes additional measurements of highly penetrating muons. The two magnet systems bend charged particles in the Inner Detector and the muon spectrometer, allowing their momenta to be measured. The onion-layers concept of the ATLAS detector is best illustrated in Figure 2.

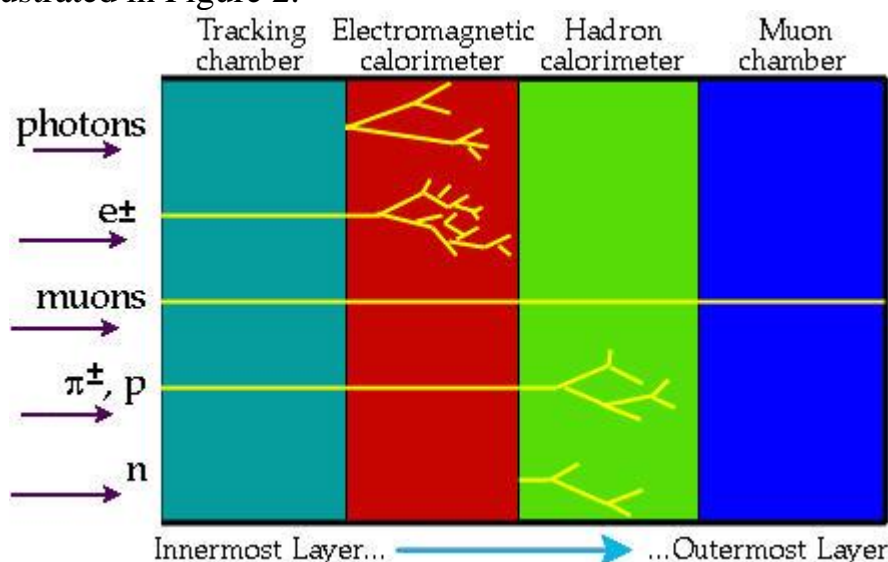


Figure 2: Schematic illustration of particle signatures in the four sub-detector layers of ATLAS.

The design criteria of the ATLAS detector include: [5]

- Efficient tracking at high luminosity for high- p_T lepton-momentum measurements, electron and photon identification, τ -lepton and heavy-flavor identification, and full event reconstruction capability at lower luminosity;
- Very good electromagnetic calorimetry for electron and photon identification and measurements, complemented by full-coverage hadronic calorimetry for accurate jet and missing transverse energy ($E_{T\text{miss}}$) measurements;
- High-precision muon momentum measurements, with the capability to guarantee accurate measurements at the highest luminosity using the external muon spectrometer alone;

- Large acceptance in pseudorapidity (η) with almost full azimuthal angle (ϕ) coverage everywhere. The azimuthal angle is measured around the beam axis, whereas pseudorapidity relates to the polar angle (θ) where θ is the angle from the z direction (For a definition of ATLAS coordinate system, see chapter 5.1).
- Triggering and measurements of particles at low- p_T thresholds, providing high efficiencies for most physics processes of interest at LHC.

The overall detector layout is shown in Figure 3.

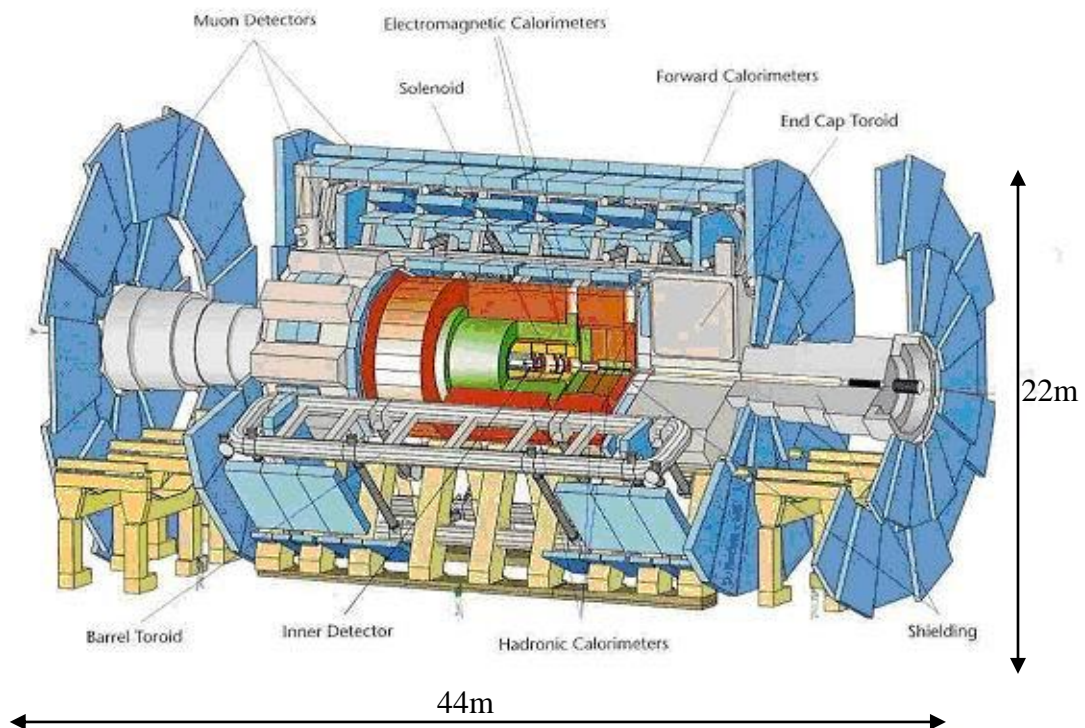


Figure 3: Schematic view of the ATLAS detector.

3.1 The Inner Detector

The Inner Detector (ID) begins just a few centimeters from the proton beam axis and is contained within a cylinder of length 7 m and a radius of 1.15 m, in a solenoidal magnetic field of 2T [6]. Pattern recognition, momentum and vertex measurements, and electron identification are achieved with a combination of discrete high-resolution semiconductor pixel and strip detectors in the inner part of the tracking volume, and continuous straw-tube tracking detectors with transition radiation capability in its outer part.

The high granularity needed for the high-precision measurement, characterized by the large number of tracks, is achieved by three types of detectors: The Pixel Detectors, Semiconductor Tracking detectors (SCTs)

and the Straw Tube Trackers (TRTs). The pixel detector is composed of an array of small pixel modules designed to provide a very high-granularity, high-precision set of measurements as close to the interaction point as possible. The SCT system is an additional layer of silicon subdivided into narrow strips designed to provide eight precision measurements per track in the intermediate radial range and good pattern recognition by the use of high granularity. The TRT is a collection of gas-wire drift detectors which can operate at the very high rates expected at the LHC by virtue of their small diameter and the isolation of the sense wires within individual gas volumes. Together they all provide a robust pattern recognition system of 36 tracking points per track and an $|\eta| < 2.5$ coverage.

The inner detector layout is shown in Figure 4.

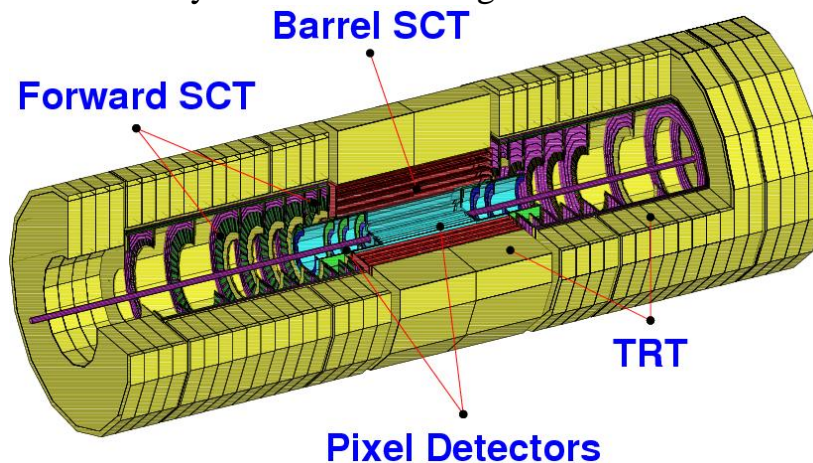


Figure 4: Schematic view of the ATLAS Inner Detector.

3.2 The Calorimeters

The calorimeters are situated outside the solenoidal magnet that surrounds the inner detector and consist of two separate systems: an inner electromagnetic calorimeter and an outer hadronic calorimeter. The tasks of the calorimeters at hadron colliders are: accurate measurement of the energy and position of electrons and photons; measurement of the energy and direction of jets, and of the missing transverse momentum of the event; particle identification, for instance separation of electrons and photons from hadrons and jets, and of τ hadronic decays from jets; event selection at the trigger level.

The electromagnetic (EM) calorimeter [7] is a lead Liquid Argon (LAr) detector with accordion geometry meant to absorb energy from particles that interact electromagnetically, which include charged particles and photons. Its barrel and endcap regions cover a total of $|\eta| < 3.2$.

The hadron calorimeter [8] is a sampling calorimeter using iron as the absorber material and scintillating tiles as active material. It absorbs energy from particles that pass through the EM calorimeter, but do not interact via the strong force; these particles are primarily hadrons. Its barrel and endcap regions cover a total of $|\eta| < 5$. The calorimeters layout is shown in Figure 5.

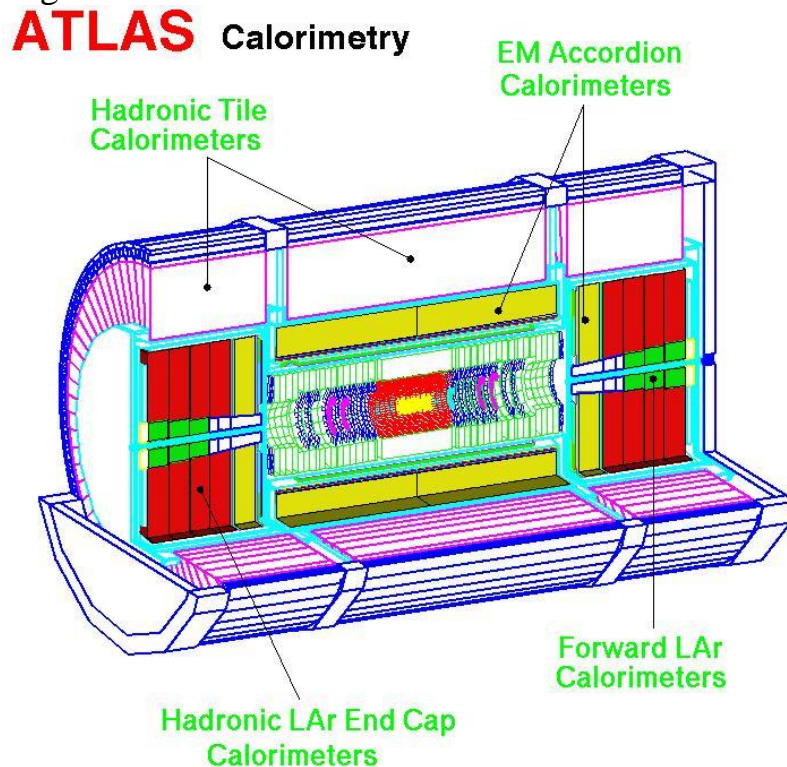


Figure 5: Schematic view of the ATLAS Electromagnetic and Hadronic Calorimeters

3.3 The Magnet System

The ATLAS detector uses two large magnet systems to bend charged particles so that their momenta can be measured. The magnet system is an arrangement of a central solenoid (CS) providing the Inner Detector with magnetic field, surrounded by a system of three large air-core toroids generating the magnetic field for the muon spectrometer. The barrel toroid (BT) is comprised of 8 large magnet toroids surrounding the end-caps and CS. The two end-cap toroids (ECT) are inserted in the barrel toroid at each end and line up with the CS.

The CS provides a central field of 2T [5] with a peak magnetic field of 2.6T at the superconductor itself. The peak magnetic fields on the superconductors in the BT and ECT are 3.9T and 4.1T respectively. The magnets are indirectly cooled by forced flow of helium at 4.5 K through tubes welded on the casing of the windings.

The ATLAS magnet system layout is shown in Figure 6.

The overall dimensions of the magnet system are 26m in length and 20m in diameter.

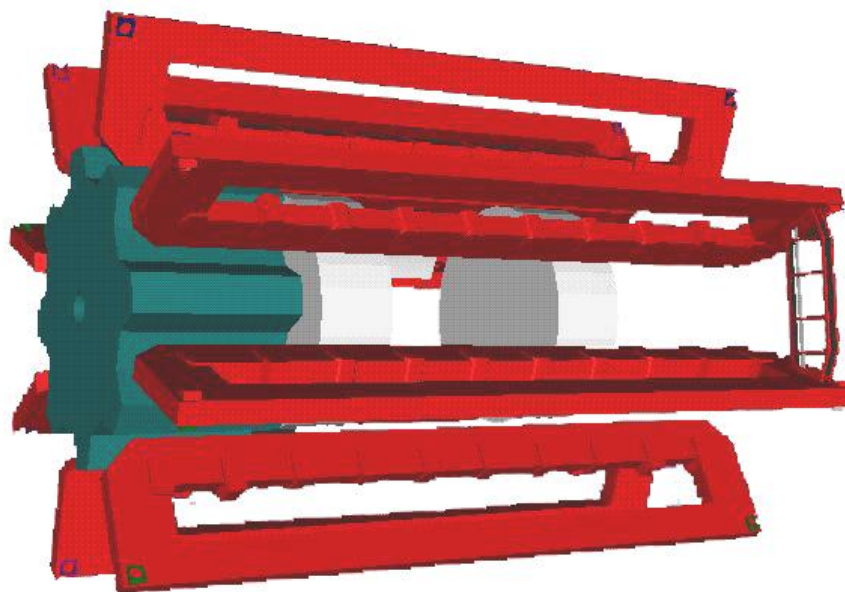


Figure 6: Schematic view of the ATLAS magnet system. The red Barrel Toroids produce a field which curves around the detector, through the openings in the toroids; the green-blue End-cap Toroids produce a field around the End-cap calorimeters; the white disk Solenoids produce fields which are parallel to the beam pipe axis.

3.4 The Muon Spectrometer system

The massive Muon Spectrometer was built all around the barrel and endcaps of the ATLAS detector. Its tremendous size is required to measure with great accuracy the momentum and tracks of muons escaping the inner layers of the detector. Precise muon measurements are vital because one or more muons are a key element of a number of interesting physical processes, and because the total energy of particles in an event could not be measured accurately if they were ignored.

The Muon Spectrometer covers an area of roughly 12000 m², and using approximately one million readout channels. It is comprised of two types of chambers – trigger chambers (Resistive Plate Chambers and Thin Gap Chambers) and tracking chambers (Monitored Drift Tubes and Cathode Strip Chambers).

The conceptual layout of the Muon Spectrometer [9] is based on the magnetic deflection of muon tracks in a system of three large superconducting air-core toroid magnets instrumented with separate-function trigger and high-precision tracking chambers (See Figure 7). It functions similarly to the inner detector, with muons curving so that their momentum can be measured, albeit with a different magnetic field configuration, lower spatial precision, and a much larger volume. It also serves the function of simply identifying muons, as very few particles of

other types are expected to pass through the calorimeters and subsequently leave signals in the muon spectrometer.

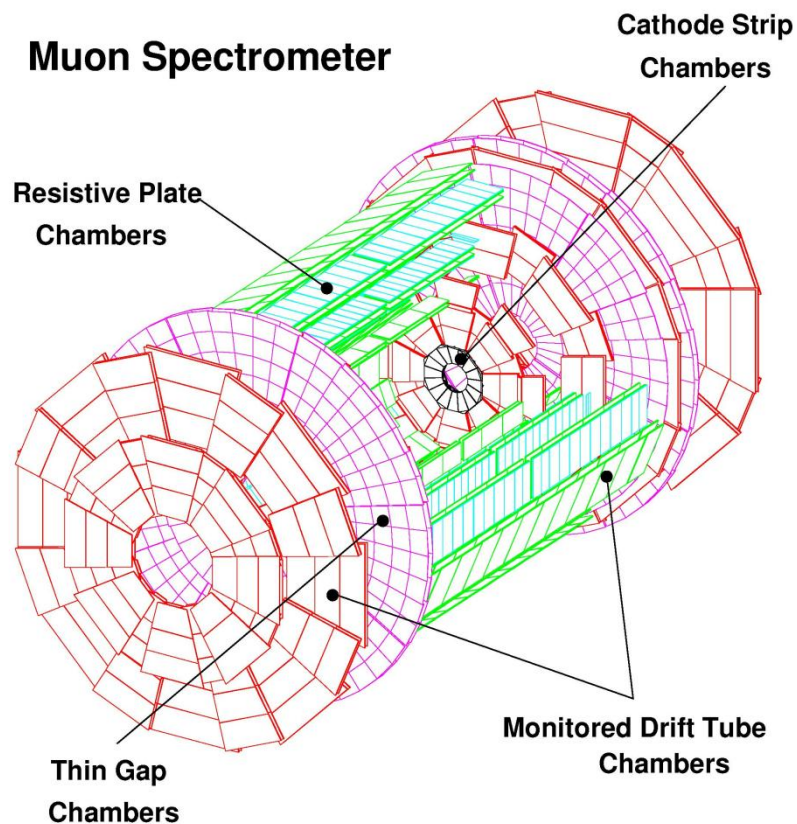


Figure 7: Schematic view of the ATLAS Muon Spectrometer System. Different chamber technologies are marked by arrows.

As seen in Figure 7, the Muon Spectrometer is roughly segmented into three regions – the Barrel region (in green) covering $|\eta| < 1$, the transition region (region of transition between barrel and end-caps) around $|\eta| \sim 1$ and the end-caps (in red, pink and black) covering $1 < |\eta| < 2.7$. In the barrel region, tracks are measured in chambers arranged in three cylindrical layers ('stations') around the beam axis; in the transition and end-cap regions, the chambers are installed vertically, also in three stations, with the exception of the CSCs which are inclined in a 11.59° angle (See Figure 8).

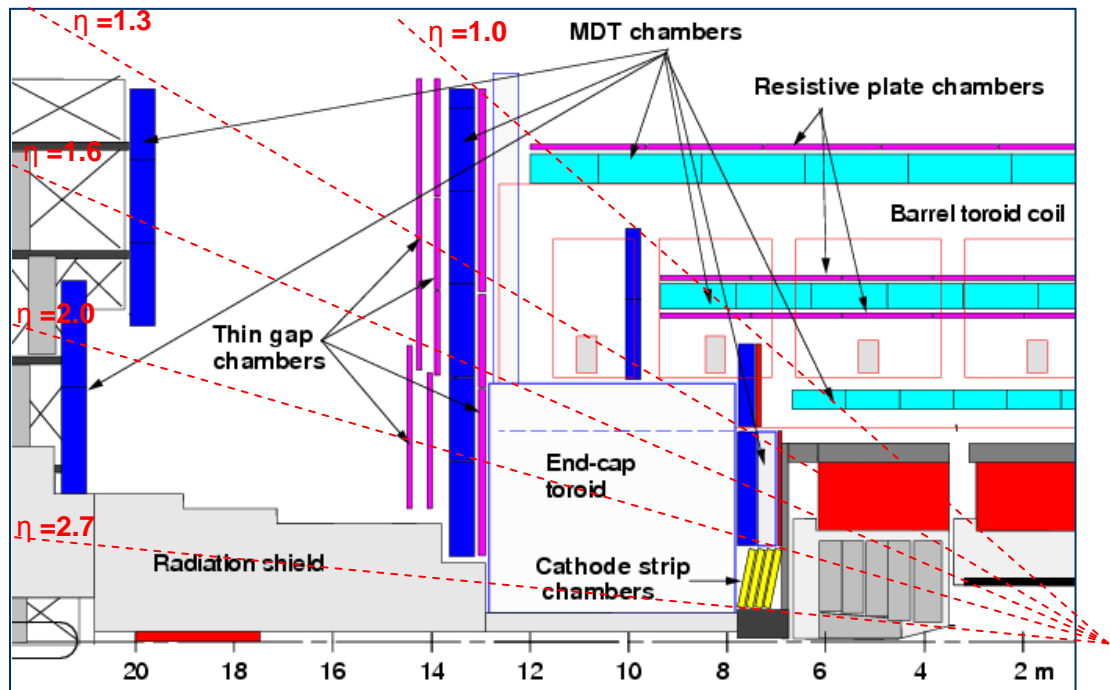


Figure 8: A schematic side-view of the ATLAS Muon Spectrometer system, showing the different chamber technologies.

A cross-section through a quarter of the detector in the z - y plane is shown. The dotted line shows the various η regions. The axis origin is the interaction point.

Two separate systems with distinct functionality are used:

Trigger: Consists of Resistive Plate Chambers (RPC) in the barrel region and Thin Gap Chambers (TGC) for the end-caps covering the spectrometer acceptance up to $|\eta| = 2.4$.

Both types of chambers generate fast signals with a time resolution of a few nanoseconds which are used for level-1 triggering and bunch crossing identification. A spatial resolution of 5–20 mm is adequate for these chambers. It is used in the pattern recognition algorithm and provides the only measurement of the track coordinate in the non-bending plane ('second-coordinate' measurement).

Precision measurement: Monitored Drift Tube chambers (MDT) for 99.5% of the area and Cathode Strip Chambers (CSC) for the remaining, a very small forward area where particle fluxes are highest. The CSCs have higher granularity and are used to sustain the demanding rate and background conditions. Although small in physical size this area covers a large range in pseudorapidity ($|\eta| = 2$ – 2.7).

The precision chambers measure the track coordinates in the bending plane with high precision. For the MDTs no information on the non-bending coordinate and on the bunch crossing time is available. The CSCs do measure both quantities, although the non-bending coordinate has a very poor resolution.

3.4.1 Resistive Plate Chambers (RPCs)

The RPCs are trigger chambers located in the barrel region of the Muon Spectrometer covering a range of $|\eta| \leq 1$. Due to high background rates at the LHC, the Monitored Drift Chambers (See section 3.4.3) will have to operate at high levels of occupancy. For this reason, it was decided to use an independent, dedicated, fast and hence low-occupancy chamber system for trigger purposes.

The trigger detector in the barrel is made up of three stations each with two detector layers. Two stations installed at a distance of 50 cm from each other are located near the centre of the magnetic field region and provide the low- p_T trigger ($p_T > 6$ GeV) while the third station, at the outer radius of the magnet, allows increasing the p_T threshold to 20 GeV, thus providing the high- p_T trigger. The trigger logic requires hits in three out of four layers in the middle stations for the low p_T trigger and, in addition, one of the two outer layers for the high- p_T trigger.

Each RPC consists of two gas gaps filled with a tetrafluoroethane / Sulfur hexafluoride mixture with two resistive plates, made of Bakelite planes and kept parallel to one another by insulating spacers (See Figure 9). Two planes of readout strips – one in the transverse and one in the longitudinal direction – provide trigger information. The transverse strips (η strips) measure the bending coordinate, the longitudinal strips (ϕ strips) measure the second coordinate. Once an incident muon traverses the planes, the primary ionisation electrons are multiplied into avalanches by a high, uniform electric field of 4.5 kV/mm and the signal is amplified by the front-end electronics. In such a way a space–time resolution of 1cm x 1ns can be reached.

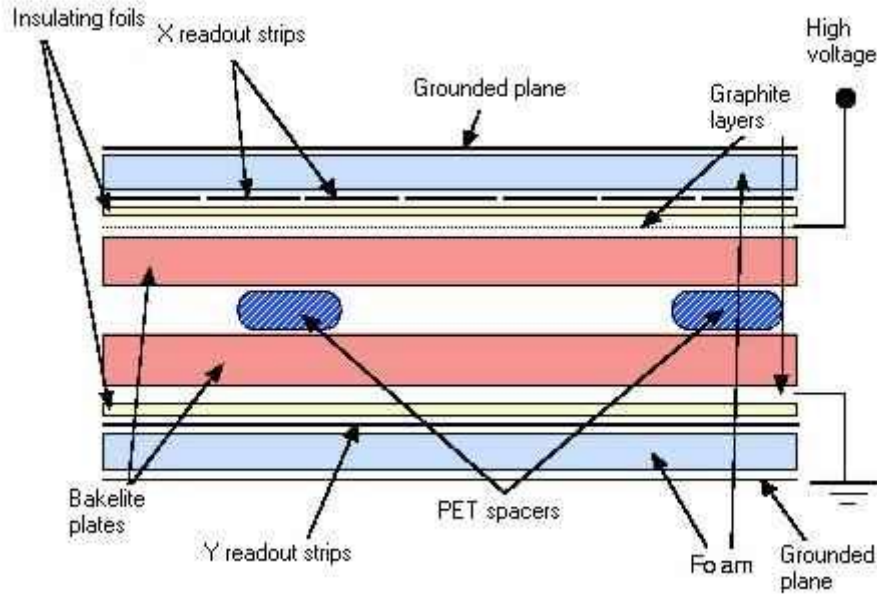


Figure 9: Schematic structure of Resistive Plate Chambers.

3.4.2 Thin Gap Chambers (TGCs)

The TGCs are another technology of trigger chambers located in the end-caps of the Muon Spectrometer, covering a range of $1 \leq |\eta| \leq 2.4$. They are similar in design to multiwire proportional chambers, with the exception that the anode wire pitch is larger than the cathode-anode distance. Thin Gap Chambers are filled with CO_2 n-pentane mixture (55:45) and operate in a saturated mode at the nominal high voltage of 2.9 kV.

The TGCs are constructed in units of doublets and triplets of TGC chambers. The inner station consists of one plane of triplets and the outer station of two planes of doublets (See Figure 8).

In a doublet, the TGC layers are separated by 20 mm thick paper honeycomb panel, which provides a rigid mechanical structure. For the triplet TGC unit another wire plane is added.

The TGC structure in Figure 10 shows the anode plain sandwiched between two cathode planes made of 1.6 mm G-10 plates on which the graphite cathode is deposited. On the backside of the cathode plates, facing the center plane of the chamber, etched copper strips provide the readout of the azimuthal coordinate.

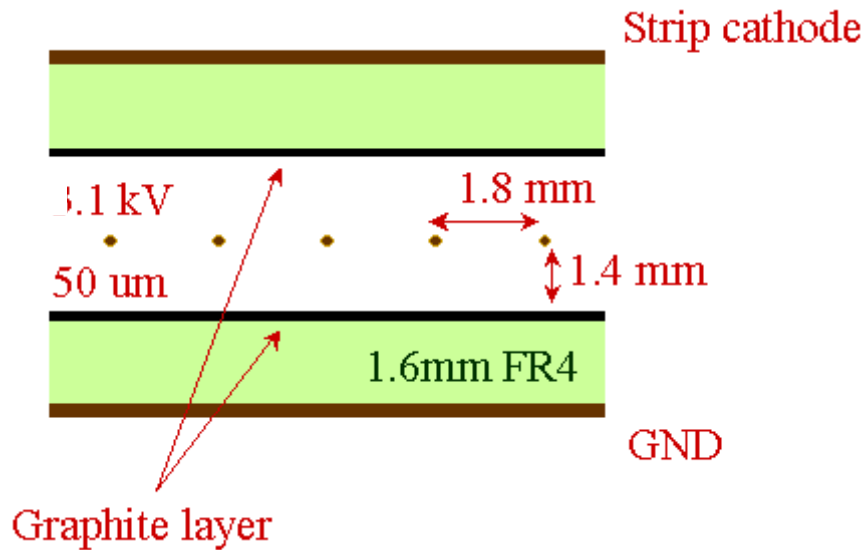


Figure 10: Schematic view of a Thin Gap Chamber.

Several anode wires (varies between 4 to 20) are grouped together and fed to a common readout channel. Signals from the anode wires together with the readout strips arranged orthogonal to the wires provide the trigger information. The signals generated by TGCs are amplified, discriminated and shaped on the detector by a two-stage amplifier in an Amplifier Shape Discriminator (ASD) circuit.

The chamber layout is shown in Figure 11 (the dimensions correspond to the so called T8 detector).

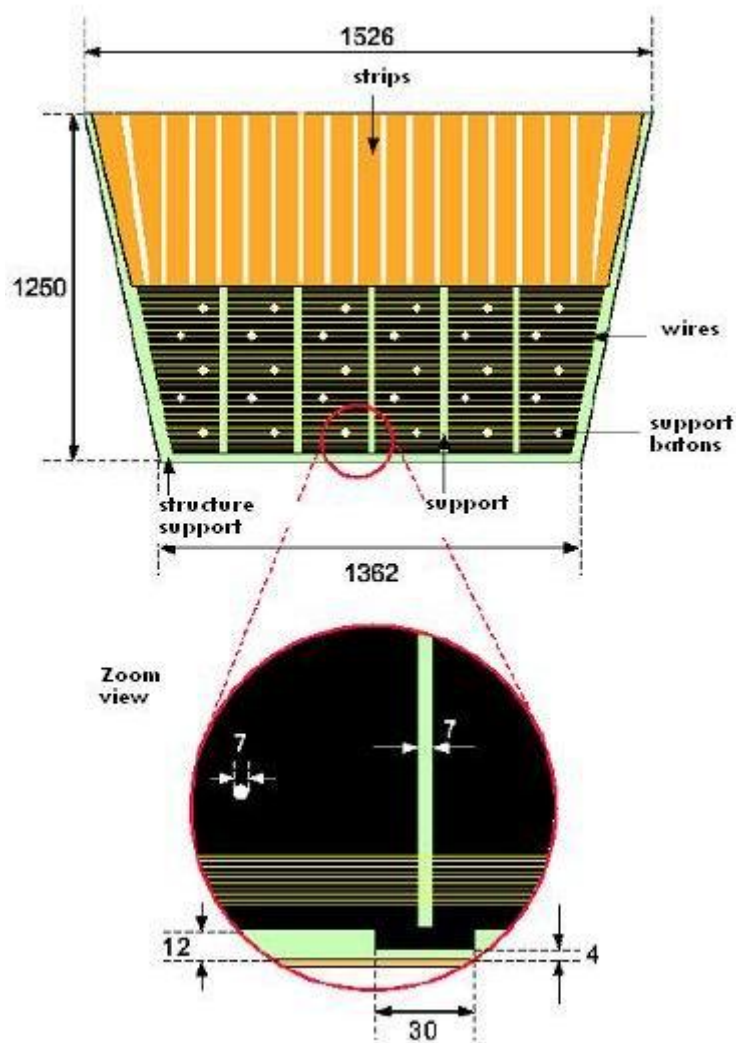


Figure 11: Top view of a T8 TGC chamber.
 The upper yellow area depicts the geometry of the strips. The lower area, under the strips, depicts the mesh of wires, perpendicular to the strips. The supports of the wires are also shown.

The ATLAS TGCs were built at the Weizmann Institute in Israel, at KEK in Japan as well as at the Shandong University in China. The chambers went through a detailed QA test procedure. All the units were tested at a cosmic testbench. There are three test sites to check the performance of the TGC units, two in Israel – at the Technion and at Tel-Aviv University – and one at Kobe University in Japan. The purpose of these tests is to provide a detailed map of detection efficiency for each detector (See Figure 12).

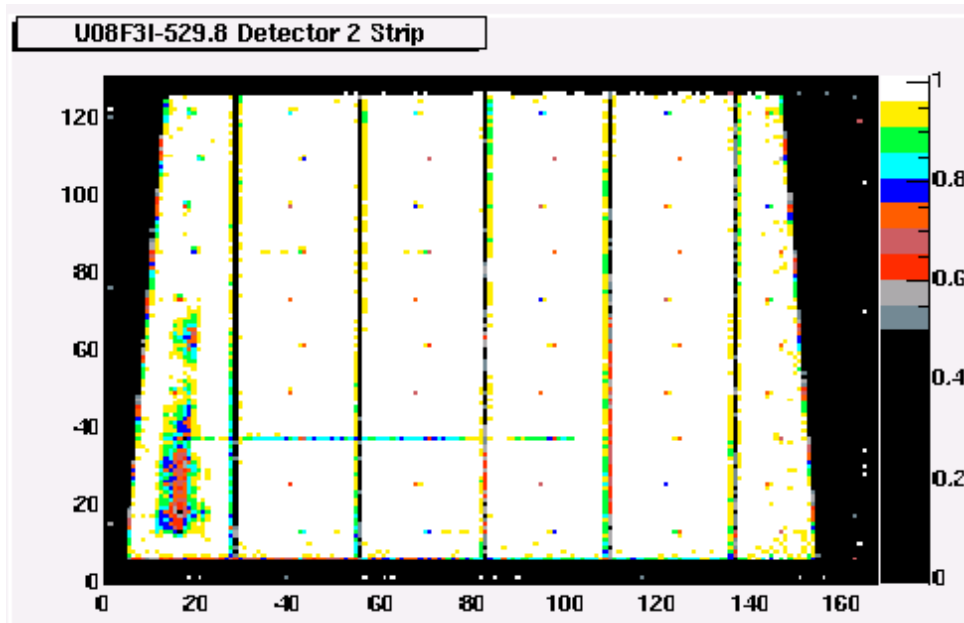


Figure 12: The efficiency map of the doublet U08F3I-529.8. For efficiency >95% the area is white, for 90-95% it is yellow, for 85-90% it is green. Efficiency below 50% is plotted in black. The scale of the colors is shown on the right of the map.

3.4.3 Monitored Drift Tubes (MDTs)

The Monitored Drift Tube Chambers perform the precision coordinate measurement in the bending direction of the air-core toroidal magnet and therefore provide the muon momentum measurement. They cover almost the entire area of $5,500 \text{ m}^2$ which is needed for a good momentum determination of the muons with rapidities between -2.7 and $+2.7$. In the innermost plane of the two end-caps, where the background is highest, they are replaced by $2 \times 14 \text{ m}^2$ of Cathode Strip Chambers.

The basic detection element is a cylindrical aluminum drift tube of 30 mm diameter and a central wire of $50 \text{ }\mu\text{m}$ diameter (See Figure 13a). It is operated with nonflammable gas composed of 93% Ar and 7% CO_2 . The wire is at a potential of 3270 V. A charged particle traversing through the tube will ionize the gas inside creating a cluster of electrons drifting towards the anode wire in a charge-avalanche. The time takes the ionized cluster to reach the anode wire and generate an electric signal is closely related to the distance between the particle hit and the wire (known as r-t relation, shown in Figure 13b).

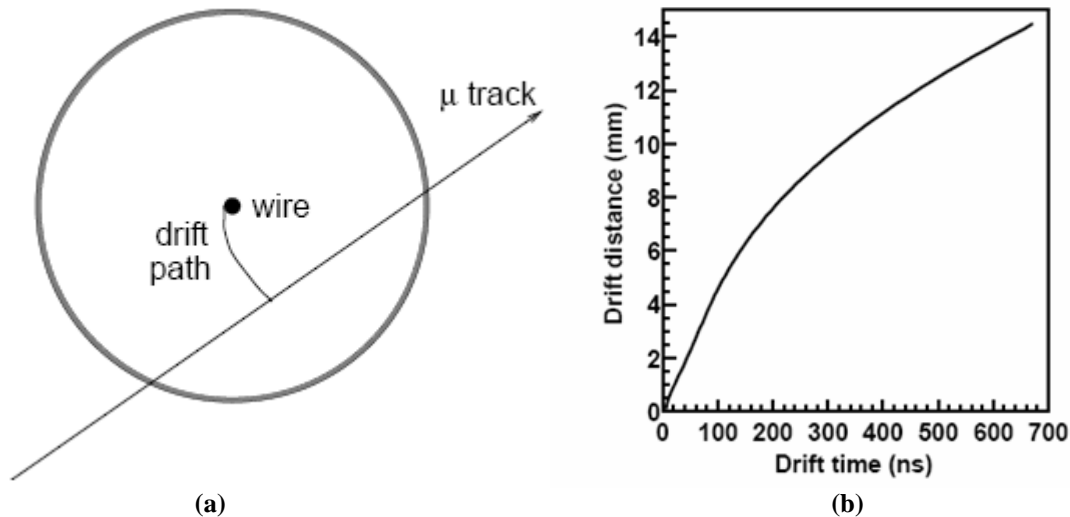


Figure 13: Charge avalanche in a single MDT tube.
 (a) An illustration of an avalanche through the anode wire in a single drift tube.
 (b) Graph of the r-t relation in a single drift tube.

Using the right r-t relation one can estimate the particle hit position in the tube. By registering the drift times of the electrons in the gas and comparing them to the expected distance-time (r-t) relation, one determines six to eight coordinates of a typical track in the plane of the layer and in the direction across the tubes. This results in a measurement of effectively one coordinate with $40 \mu\text{m}$ precision and one angle with 3×10^{-4} precision.

To improve the resolution of a chamber beyond the single-wire limit and to achieve adequate redundancy for pattern recognition, the MDT chambers are constructed from 2×4 monolayers of drift tubes for the inner station and 2×3 monolayers for the middle and outer stations. The tubes are arranged in multilayer pairs of three or four monolayers, respectively, on opposite sides of a rigid support structure (See Figure 14). The tubes are closely spaced so that each multilayer has a thickness of about 82 mm for 3 monolayers and 109 mm for 4 monolayers.



Figure 14: A view of a MDT chamber.
Two groups of 3 monolayers form a multilayer.

The chosen working point provides for a non-linear space-time relation with a maximum drift time of ~ 790 ns and a small Lorentz angle. The single-wire resolution is ~ 80 μm .

3.4.4 Cathode Strip Chambers (CSCs)

The cathode strip chambers (CSCs) cover the region in the inner station where particle fluxes are highest. They are located at approximately 7 m distance from the interaction point (See Figure 8) and occupy the pseudorapidity range $2.0 \leq |\eta| \leq 2.7$. They replace the MDTs in areas where counting rates greater than 200 Hz/cm^2 are expected. The baseline CSC gas is a non-flammable mixture of 30% Ar, 50% CO_2 and 20% CF_4 , with a total volume of 1.1 m^3 . Its characteristics are small electron drift times (30 ns), good time resolution (7 ns), good two-track resolution, and low neutron sensitivity. All of which are imperative for the high flux environment the CSCs will be operating in.

The CSC structure is based on a multiwire proportional chamber with cathode strip readout with a symmetric cell in which the anode-cathode spacing is equal to the anode wire pitch. The precision coordinate (η coordinate) is obtained by measuring the charge induced on the segmented cathode by the avalanche formed on the anode wire. The second coordinate (ϕ coordinate) is similarly obtained from a transverse set of cathode strips and wires. There are 192 precision strips and 48 non-precision strips in each CSC layer. Good spatial resolution is achieved by segmentation of the readout cathode and by charge interpolation between neighboring strips (See Figure 15b). The position resolution is around ~ 70 μm .

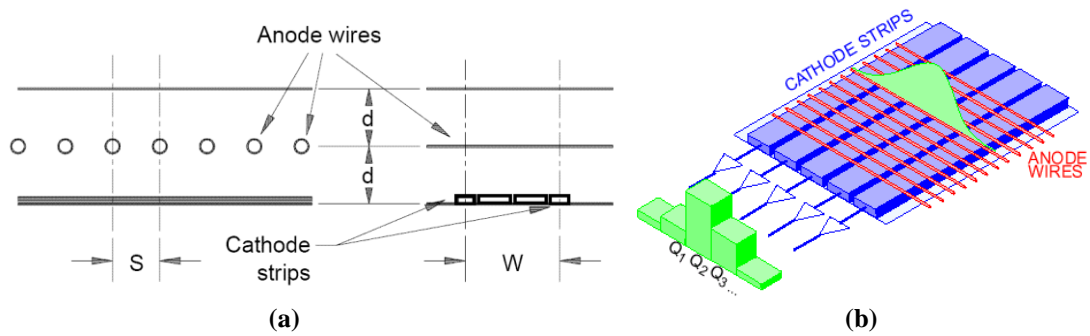


Figure 15: Schematic View of CSC precision strips.
 (a) 2D slice of a CSC layer showing wire and strip pitch.
 (b) Charge interpolation on CSC strips.

The CSCs come in two types of modules (CSS and CSL) characterized by their different sizes and are arranged in two rings of eight chambers for each type (as shown in Figure 16) and four layers for each chamber. Two identical modules form a chamber, similar to the MDTs although the CSCs do not have a spacer structure connecting the two modules. Each chamber module consists of four wire planes leading to a configuration much like the layers and multilayers of the MDT system, but with much finer granularity.

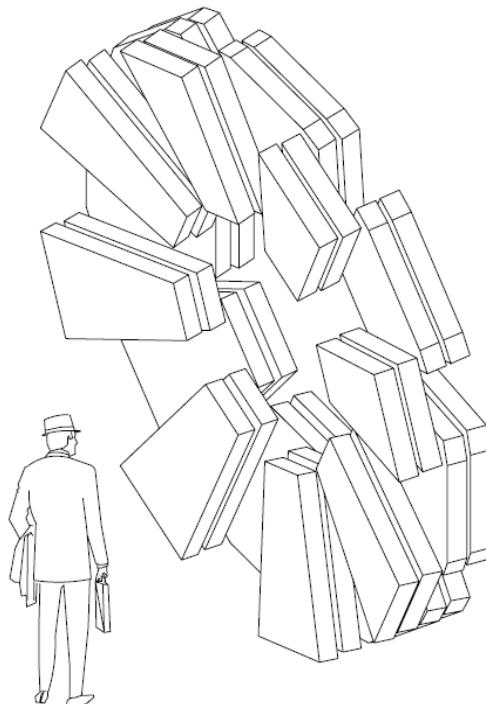


Figure 16: 3D view of the CSC chamber layout.

The spatial resolution of the CSCs is sensitive to the inclination of tracks and the Lorentz angle. To minimize degradations of the resolution due to these effects, they will be installed in a tilted 11.59° angle such that stiff tracks originating from the interaction point are normal to the chamber surface.

The cathode strips for the precision (η coordinate) measurement are oriented orthogonal to the anode wires as seen in Figure 15a. A measurement of the transverse (ϕ) coordinate is obtained from orthogonal strips, i.e. oriented parallel to the anode wires, which form the second cathode of the chamber. The charge avalanche induces charge on these two sets of cathode strips. The induced charge is spread out over adjacent strips; each strip receiving a fraction of the total induced charge. The spread of strips that receive charge is called a hit-cluster. Using the knowledge of the interpolated total charge passing through a layer, calculating the relative magnitudes of both the charge on each strip and the position of the strip in the hit-cluster will give enough information to find a centroid of the charge. The centroid is the point in the chamber where the ionization cluster originated, thus, the position of the particle's track.

Each cathode strip is connected to a preamplifier and shaper circuit, which creates a bipolar pulse with a 140 ns shaping time [10] to mitigate pile-up effects, and a strip signal response of about 500 ns. The shape of the bipolar signal and its maximum determines the induced charge over the strip. Its corresponded time (maximum time) is related to the electron clusters drift time and expected to be about 30 ns. When compared to a time reference from the trigger chambers, this defines a muon time window – the time frame in which a muon is expected to cross the chamber. The time window corresponds to the 5th to 8th time sample after the trigger hit (See Figure 17, in which the first sample is in $t=0$).

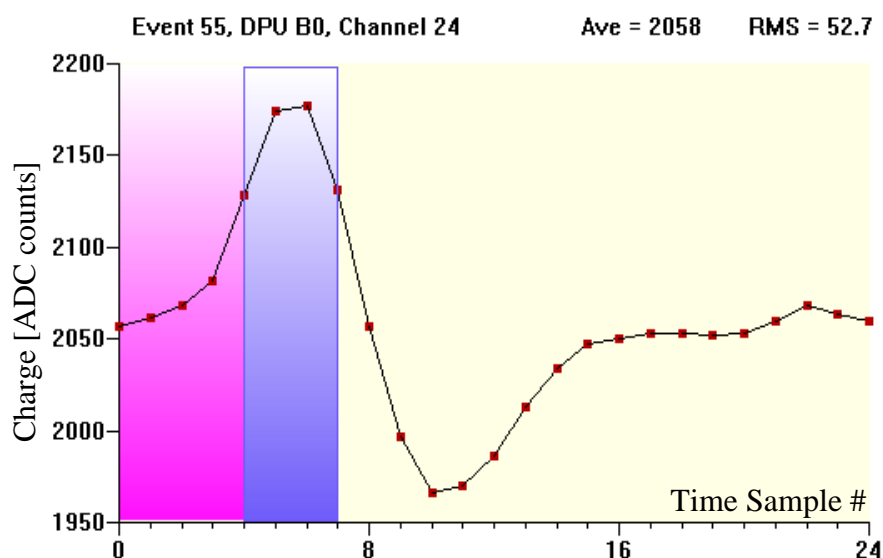


Figure 17: Bipolar shape of a signal in a single CSC strip. The central blue rectangle marks expected peak.

After the hit-time and charge are evaluated for each strip, they are fed into the reconstruction algorithm. They go through Hough Transform to find collinear hits, clusterization of these collinear hits, track finding and track-fitting of those clusters. A detailed discussion of the CSC reconstruction algorithm is found in Chapter 6.

4 Track Reconstruction in ATLAS

The role of reconstruction in HEP is to derive from the stored raw data the relatively few particle parameters and auxiliary information necessary for physics analysis. Pattern recognition programs attempt to reconstruct the passage of photons, electrons, muons, τ -leptons, K_0 s, jets as well as the footprints of missing transverse energy, primary and secondary vertices. Information from all detectors is combined so that the four-momentum reconstruction is optimal for the full momentum range, full rapidity range and any luminosity, and so that particles are identified with the least background, with the understanding that the optimum between efficiency and background rejection can be analysis-dependent.

Tracking is done in several stages [11], as illustrated in Figure 18. The Byte-stream converters take the data from the sub-detectors, and form the raw data objects. These are then used to create "prepared raw data" (PrepRawData), i.e. clusters from the pixel detector or drift circles from the muon monitored drift tubes. This requires detailed knowledge of the properties of the detector calibration and of the position and alignment of the detector. The PrepRawData (along with the SpacePoints) is then used to find local tracks (called segments) inside the sub-detectors. Several algorithms then try to combine local segments to form track candidates. This may be done by either refitting the entire track using the original hits from all the sub-detectors it crossed, or amalgamating the segments themselves into a smooth track. Candidates are carefully tested to distinguish between real tracks and background arising from duplication. Discarded track elements are freed for further use and a second pass through track finding is done. Finally, the tracks can be used to find vertices, and to create the TrackParticles (for physics analysis at the AOD level).

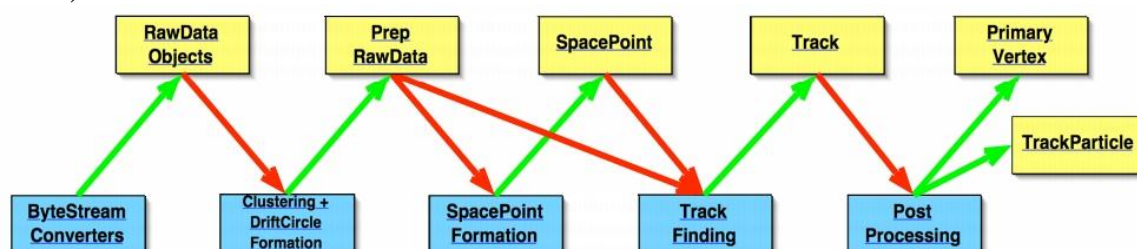


Figure 18: Tracking reconstruction chain.

The boxes on the top represent data objects, whilst the boxes on the bottom show the algorithms which work on them. The arrows show the direction of data flow.

A typical reconstruction algorithm takes one or more collections as input, calls a set of modular tools, and outputs typically one collection of reconstructed objects. Common tools are shared between tracking

detectors on one side (inner detector and muon spectrometer) and calorimeters on the other side (liquid argon electromagnetic calorimeter, hadronic endcap and forward calorimeter, and tile hadronic detector). Reconstruction tools can share interfaces, for example for different types of calorimeter cluster corrections, or track extrapolation.

Muon track reconstruction in ATLAS is handled by two independent tracking systems – the inner particle detector and the outer muon spectrometer. While resolution in the inner detector is better, reaching to $12\mu\text{m}$ in the pixel detectors, the long lever arm of the muon spectrometer and the relatively clean detector space (See Figure 19) give rise to two muon identification approaches: **outside-in** and **inside-out**.

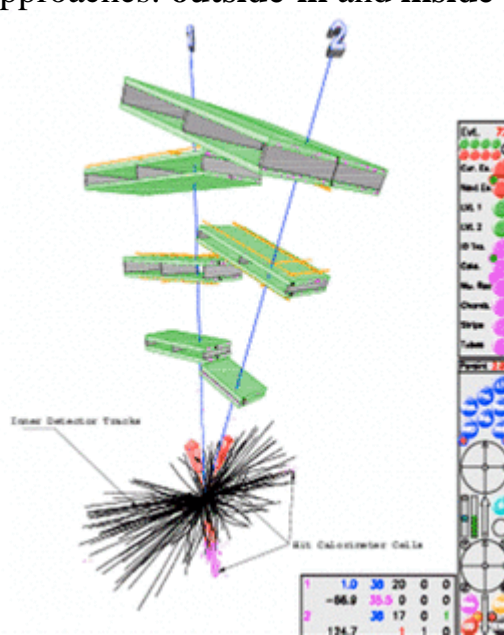


Figure 19: Reconstruction of a simulated event in ATLAS as shown by Persint event viewer. Black tracks in the center represent the Inner Detector tracks; Blue tracks (marked 1, 2) represent 2 high- P_T muons passing through MDTs in the Muon Spectrometer.

The **outside-in** approach based on two main steps. The first step is a “standalone” muon reconstruction in the Muon Spectrometer. In this step the trigger chambers are usually used for building seeds for finding track candidates, and then a fine track is reconstructed using the precision chambers. In the second step, the muon tracks or segments are combined with Inner Detector tracks to obtain the muon momentum at the interaction point.

The **inside-out** strategy identifies muons in the traditional way, associating muon hits and segment to an Inner Detector track in order to flag the track as a muon. The muon momentum and position are taken from the Inner Detector track parameters for low momenta muons (up to about 50 GeV), and from the Muon Spectrometer track parameters for higher momenta muons. Reconstruction is done by the iPatRec

(Section 4.1.1) and xKalman (Section 4.1.2) algorithms in the Inner Detector and by MuonBoy or MOORE in the Muon Spectrometer. Track combination in ATLAS is done by Muonboy (Section 4.1.4) and MuID (Section 4.1.3), which perform the outside-in strategy and MuGirl (Section 4.1.5) which perform the inside-out strategy.

Local tracking in the Muon Spectrometer is utilizing the Monitored Drift Tubes or the Cathode Strip Chambers in the high eta region. A summary of the tracking algorithm in the MDTs is found in section MDT Tracking 4.2. The detailed discussion on the CSC tracking system is found in chapter 6.

4.1.1 iPatRec

iPatRec [12] began life in 1992 as an Inner Detector reconstruction program, written in FORTRAN. The current version, using the same strategy, is a set of C++ packages. This modular set of algorithmic packages creates track-candidates using space-point combinatorials classified by maximum curvature and crude vertex region projectivity. Candidates undergo a track-fit procedure and the track follower algorithm propagates the fit parameters through the inhomogeneous magnetic field to form an intersect with error ellipse at each silicon detector layer in turn.

4.1.2 xKalman

xKalman++ [13] is a reconstruction package used by the ATLAS Inner Detector. The package was born in ATRECON in FORTRAN and the C++ code is still very much written in this style, making it in parts hard to read. xKalMan's strategy follows these general stages:

1. Define the Region of Interest (ROI).
2. Produce space points.
3. Pattern recognition in the TRT. Primary tracks found set possible track candidate trajectories.
4. Extrapolate track candidates to the SCT and the Pixel sub detectors and produce local pattern recognition in these detectors.
5. Compare all track candidates, keeping tracks in the storage only if the number of unique clusters (belong only to a given track) are above some threshold.
6. Reconstruct the primary vertex.

4.1.3 MOORE and MUID

MOORE (Muon Object Oriented Reconstruction) [14] is the software package for track reconstruction in the Muon Spectrometer, developed in C++ in the ATHENA framework according to modern Object Oriented

design principles. Its design was driven by the goal of performing track reconstruction in a highly modular way, with the highest possible efficiency in all the pseudorapidity range covered by the Muon Spectrometer and with the best possible resolution needed for muon identification in ATLAS.

The purpose of the MUID (Muon Identification) package is to associate tracks found in the Muon Spectrometer with the corresponding Inner Detector track and calorimeter information in order to identify muons at their production vertex with optimum parameter resolution.

MUID is written in C++. It shares some general reconstruction classes and methods with the inner detector reconstruction package iPatRec. It uses currently iPatRec for ID reconstruction and accesses the results from Muon Standalone packages MOORE.

4.1.4 MuonBoy

MuonBoy [9] (or in its former name MuonBox) is a FORTRAN software package developed mainly by the Saclay ATLAS muon group and designed to reconstruct tracks in the muon spectrometer. The principal algorithms used in MuonBoy include treatment of multiple scattering, dead matter, energy loss in calorimeters and inner tracker information to form a powerful muon identification scheme.

The pattern recognition strategy used by MuonBoy can be summarized in four main steps:

1. Identification of "Regions Of Activity" (ROAs), guided by trigger chamber information. ROAs are identified in (η, ϕ) space using trigger chamber data.
2. Reconstruction of local straight track segments in each muon station belonging to a ROA.
3. Combination of track segments in different muon stations, to form muon candidates using three-dimensional tracking in the magnetic field.
4. Global re-fit of muon track candidates using individual hit information.

4.1.5 MuGirl

MuGirl [15] presents a new approach lead by the Technion team (written in C++) which meant to identify muons in by associating muon hits and segment to an inner detector track in order to flag the track as a muon. Contrary to MuonBoy and MOORE, the initial muon momentum and position are taken from the inner detector track parameters, giving a good measure of the muon momentum for muons with P_T up to about 50 GeV.

This approach follows the inside-out strategy and has advantages in low P_T muons, which lose a significant part of their energy in the calorimeter. Some of them cannot be reconstructed in the muon spectrometer since there are no sufficient hits for the track fit. This is because the track was modified too much by energy loss and multiple scattering, or because of incomplete Muon Spectrometer coverage. This approach may suffer from high complexity where the number of track candidates in the Inner Detector is large, especially for high radiation background.

4.2 MDT Tracking

The Monitored Drift Tube chambers (MDTs) cover the most of the Muon Spectrometer area, providing good hit position measurements. The drift tubes are very efficient – a particle that crosses a tube produces a hit with a very high probability. In a high radiation background, the drift tubes suffer from background hits, which often deteriorate the measurements of the particles of interest. In such environment, naïve tracking algorithms may suffer from high fake track rate.

As explained in section 3.4.3, the particle hit position in an MDT tube can be measured using a so called r-t relation. The hit positions in all layers of the two MDT multilayers are combined to form tracks through the chamber. Since the curvature of the muon track in a single chamber is negligible, the tracking problem can be simplified using only straight tracks for a single MDT chamber tracking. Tracking, as seen in Figure 20, is done by finding all lines that are tangent to as many hit radii as possible. The more radii a line traverses, the more probable it originated from a real particle passing the detector rather than a random culmination of background/noise hits (Explained below).

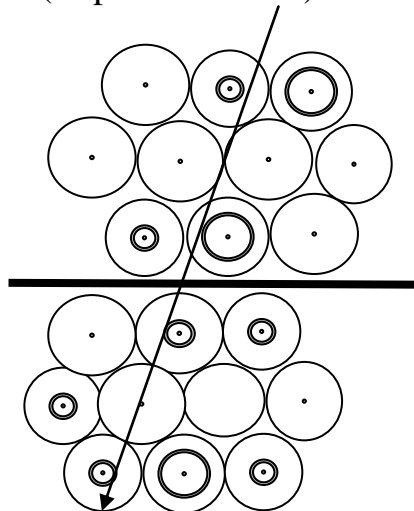


Figure 20: A schematic example of a muon track reconstruction in the MDT chamber. Circles denote the radii in which hits were found.

The fact that an MDT tube has a very long dead time of about 790ns presents a complex problem of missing information. Missing information can arise from numerous reasons:

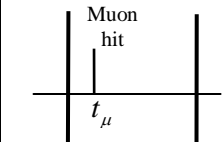
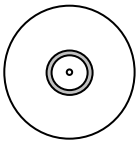
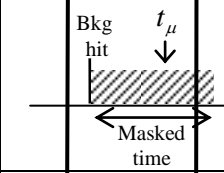

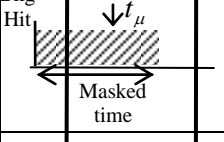
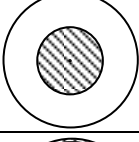
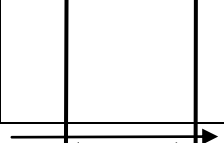
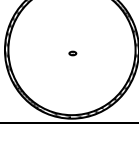
- Background particle crossing the tube before the muon, causing dead time. When the muon passes through the tube, it is not registered.
- Particle from a previous event crossing the tube.
- Muon crosses the walls of the tube.

It is possible to use this information [16] to calculate the regions a muon track might have crossed without being registered.

Since the drift time of the electron clusters in the tube is relatively long compared to the proton beam collision rate, a trigger chamber provides a reference time t_0 . It is the earliest time a muon may be expected to pass the tube. The maximum possible time a muon may have crossed is the tube's drift time (790ns). Using these quantities, a time window $[t_0, t_0+790]$ may be defined within which a muon is expected. All information coming from the MDT tube can then be broken down to one of four possible scenarios described in

Table 1:

- a. A proper muon hit within the time window. This is the “simple” case in which the hit radius registered in the tube represents the real muon hit position.
- b. A background particle hit the tube within the time window, causing it to fire and mask a subsequent muon later passing through this tube. If the masking particle hit the tube closer to the anode wire than the real muon, its signal will be the one registered and not the muon. In this case the muon might have hit anywhere outside the masking particle's hit radius, i.e. from the registered radius to the tube walls.
- c. A background particle hit the tube before the time window, causing it to fire and mask subsequent muon later passing through this tube. If the subsequent muon hits the tube at a smaller radius than the masking particle, it would not be registered since this area is still recovering from the masking particle's charge avalanche. Therefore, a muon might have hit anywhere inside the masking particle's hit radius, i.e. from the registered radius to the tube center.
- d. Muon crossed the tube walls producing no hit.

Scenario	Time diagram	Geometrical representation
(a) Muon hit in a valid time window		
(b) Background hit in a valid time window		
(c) Hit before the valid time window		
(d) No hit		

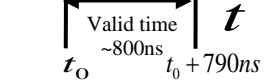


Table 1: The different scenarios given for the muon to cross the tube. The left column describes the scenario, the middle column describes the hit time, and the right column describes the geometrical representation of the possible muon track distance to the tube center. The grey area in the middle column describes the tube dead time. The grey area in the right column describes the possible regions in which the muon track crosses the tube. A possible muon crossing the tube walls is represented by a grey ring around the tube wall. The time t_μ in cases (b) and (c) is a possible muon hit time.

A possible muon track is one which crosses through all grey areas as specified in scenarios (a)-(d). Using a combination of these scenarios for muon track in an MDT chamber, a weighted sum is calculated for the purpose of accepting or rejecting that track. If Λ_i represents number of tubes belonging to scenario i which the track has crossed ($\Lambda_i \in (0, 1, \dots, M)$ where M is the number of tubes crossed), then the muon track will be accepted if the weighted sum of Λ_i is above a predefined threshold:

$$(1) \quad \Lambda = \sum_{i=1}^4 \Lambda_i \cdot w_i > \lambda$$

where λ is the predefined threshold and w_i are the weights for each basic scenario. During the calibration process one can determinate the background level and measure the empirical probability for each scenario of table 2. Thus, the weights can be used as known prior information. The geometrical representation of the result is show in Figure 21.

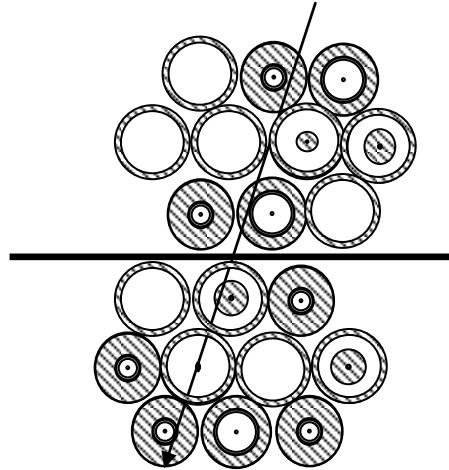


Figure 21: Geometrical representation of the MDT tube scenarios.
The geometrical representation for each tube is the combination of the basic scenarios of Table 1. The track candidate crosses one possible track path area for each tube corresponding to one scenario. The algorithm takes the track whose weighted sum is above a predefined threshold.

To find these tracks, the MDT reconstruction uses a modified form of the Hough Transform adopted for the drift circle problem. One of the problems it tackles is the hit-position ambiguity. Since the hit position registers as a radius, it is unclear whether the particle crossed the MDT on the left or right side of a tube. A detailed discussion of the discrete Hough Transform for CSC detectors is found in section 6.1.

5 Atlas Software Environment (ATHENA)

The ATHENA framework [17] is an enhanced version of the Gaudi framework that was originally developed by the LHCb experiment, but is now a common ATLAS-LHCb project and is in use by several other experiments. ATHENA and Gaudi are realizations of a component-based architecture (also called Gaudi) which was designed for a wide range of physics data-processing applications. The fact that it is component-based has allowed flexibility in developing both a range of shared components and, where appropriate, components that are specific to the particular experiment and better meet its particular requirements.

ATHENA is comprised of many interconnected components. Its main building blocks are:

- Application Manager: The application manager is the overall driving intelligence that manages and coordinates the activity of all other components within the application. There is one instance of the application manager and it is common to all applications.
- Algorithms: Algorithms share a common interface and provide the basic per-event processing capability of the framework. Each Algorithm performs a well-defined but configurable operation on some input data, in many cases producing some output data.
- Tools: A Tool is similar to an Algorithm in that it operates on input data and can generate output data, but differs in that it can be executed multiple times per event. In contrast to Algorithms, Tools do not normally share a common interface so they are more specialized in their manipulation. Each instance of a Tool is owned, either by an Algorithm, a Service, or by default by the AlgToolSvc.
- Transient Data Stores (TDS): The data objects accessed by Algorithms are organized in various transient data stores depending on their characteristics and lifetimes. The event data itself is managed by one store instance, detector conditions data, such as the geometry and alignment, by another store, etc.
StoreGate is the ATLAS implementation of the TDS. It manages the data objects in transient form, it steers their transient/persistent conversion and it provides a dictionary allowing to identify and retrieve data objects in memory. There exists also a long-term data store called DetectorStore.

Key feature in the ATHENA framework is the Event Data Model (EDM) which provides a common interface and data objects for all

associated algorithms in the framework. It will be discussed in detail in section 5.1.

5.1 Simulation

The ATLAS event simulation chain is long and complex. Input for simulation comes from event generators after a particle filtering stage. Data objects representing Monte Carlo truth information from the generators are read by simulation and processed. Hits produced by the simulation can be directly processed by the digitization algorithm and transformed into Raw Data Objects (RDOs). Alternatively they can be sent first to the pile-up algorithm and then passed to the digitization stage.

RDOs produced by the simulation data-flow pipeline are used directly by reconstruction. Thus the simulation and reconstruction pipelines are coupled together by the RDOs which act as the output from the simulation pipeline and the input to the reconstruction pipeline.

In the first stage of the simulation chain, various event generators model the physics processes, producing hundreds of particles per event at LHC energies. Generators model the physics of hard processes, initial- and final-state radiation, multiple interactions and beam remnants, hadronization and decays, and how these pieces come together. They also illustrate uncertainties in the physics modeling.

At the second stage of the chain, the ATLAS detector simulation suite is called. The simulation programs are written within the Geant4 [18,19] simulation package. It provides both a framework and the necessary functionality for running detector simulation in particle physics and other applications. The primary functionalities it provides include optimized solutions for geometry description and navigation through the geometry, the propagation of particles through detectors, the description of materials, the modeling of physics processes, and visualization. Pileup events (i.e. the overlaying of signal and background events) can also be added to the simulation chain.

In the last stage, digitization, the hits produced either directly by the simulation, or from the merging of pile-up events, need to be translated into the output actually produced by the ATLAS detectors. The propagation of charges (as in the tracking detectors and the liquid argon calorimeter) or light (as in the case of tile calorimeter) into the active media has to be considered as well as the response of the readout electronics. The final outputs of the digitization step are RDOs that should resemble the real detector data. In addition to RDOs, Simulation Data Objects (SDOs) are created to save some simulation information

that may be useful to the downstream user. The navigation between SDOs and RDOs is achieved by using identifiers.

5.2 *The Event Data Model*

The Event Data Model (EDM) [20] is a crucial element of the overall infrastructure that defines the management and use of data objects in its transient state. It improves commonality across the detector subsystems and subgroups such as trigger, test beam reconstruction, combined event reconstruction, and physics analysis by providing a common interface for all algorithms to work in. The common interface exists in the form of common data objects for raw data, preprocessed data, regions of interest, segments, tracks etc, and in the form of common general-use algorithms. For the purpose of this paper only the relevant component of the ATLAS EDM will be discussed – The Tracking EDM [21].

The concept of a common track object implies that the basic track must contain information describing the path of the track in the detector. Depending on the algorithm creating the tracks, the information content of the tracks can differ. The track can be filled with one or multiple sets of parameters, describing the track trajectory and position at different surfaces. The track must also be flexible enough to handle both local and global position coordinates (discussed in the next section). Further it must be possible to store information regarding the quality of the overall track fit.

The input data object for track reconstruction is `PrepRawData`, which is a common base input class for all tracking detectors. The output is given in the form of a common container class, `Track`, mentioned above.

As a track passes through the ATLAS detector, it crosses many ‘surfaces’ (such as detector elements, dead material etc.), and the quantities which make up a track are usually defined on a surface: that is, measurements are taken on a detector element, and track parameters are always expressed with respect to one. It is therefore useful to group information defined on the same surface, by that surface. In practice `Track` contains collection of `TrackStateOnSurface` objects, which in turn can contain information representing local coordinates and momentum of the track, the measurement found on that surface, an outlier measurement, an interaction with material on that surface and the fit quality of the measurement.

Measurement information in `TrackStateOnSurface` contains various measurement type classes derived of a common base class,

MeasurementBase. The ones relevant for this paper are the RIO_OnTrack, the clusters and drift circles after applying additional calibration corrections, and MuonSegment, a set of fitted RIO_OnTracks representing a track in a detector module (i.e. a CSC chamber).

5.3 Geometry and GeoModel

Geometry information is vital for the correct spatial interpretation of the detector measurements and for many reconstruction processes like track search and calculation of material effects. For this purpose, the tracking EDM recognizes two specific measurement frames: global frame and local frame. In addition to these, the CSC tracking algorithm defines its own measurement frame named chamber-frame.

The GeoModel toolkit is a library of geometrical primitives that can be used to describe detector geometries. The toolkit is designed as a data layer, and especially optimized in order to be able to describe large and complex detector systems with minimum memory consumption. Its main purpose is to support a central store for detector-description information that can be accessed by two main clients - simulation and reconstruction programs.

5.3.1 Global Coordinate Frame

The global coordinate system is a general three-dimensional coordinate frame in which positions of all detector elements in ATLAS are defined. This frame is the one used for combining track segments into a global track and when discussing track parameters crossing several detector elements. The origin of this frame is the ATLAS interaction point. Figure 22 illustrates the coordinate system.

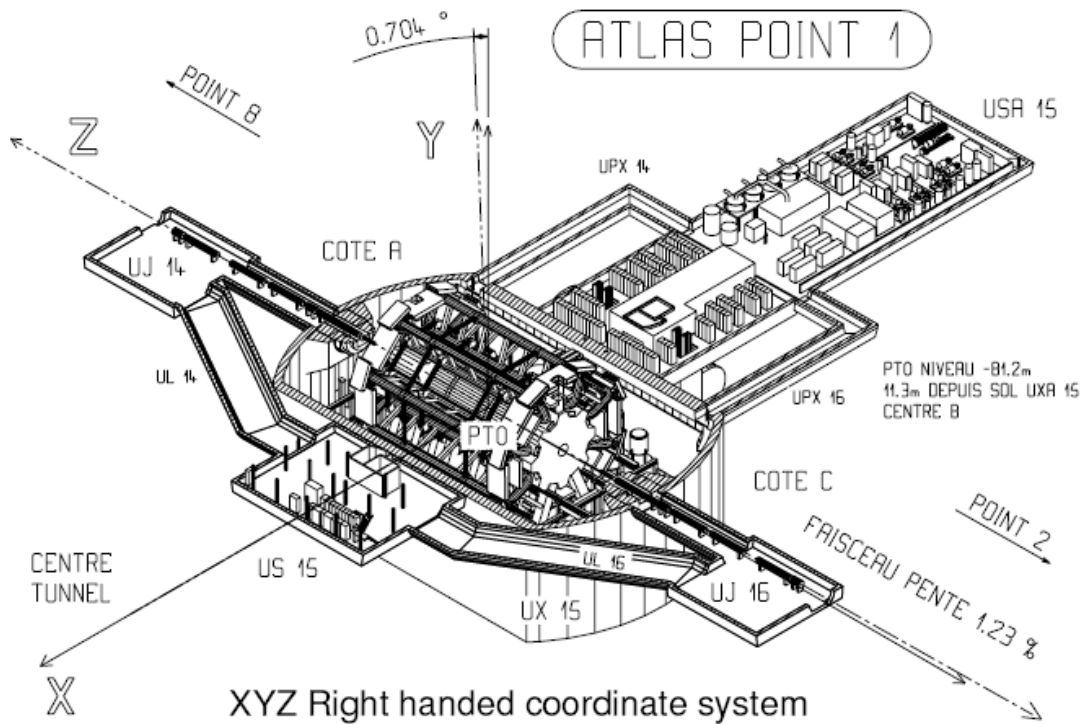


Figure 22: Schematic view of the ATLAS detector oriented in the global coordinate system.

The following describes the global coordinates in Cartesian and Cylindrical representations:

- X : Cartesian X, pointing to center of LHC ring, perpendicular to beam.
- Y : Cartesian Y , pointing up, perpendicular to beam.
- Z : Cartesian Z, pointing along beam, as defined by X and Y for RH system.
- R : Cylindrical radius, pointing out radially from beam.
- ϕ : Cylindrical azimuthal angle, as defined by R and Z for RH system around the beam axis, where $\tan \phi = p_y / p_x$.
- θ : Polar angle, defined in the ZY plane from the beam axis, where $\tan \theta = p_y / p_z$.
- η : Pseudorapidity. Defined as $\eta = -\ln \tan(\theta/2)$.

5.3.2 Local Coordinate Frame

The intrinsic frame on detector elements or surfaces will be called local frame. Positions given with respect to the local frames will be called local positions. Each sub-detector module (i.e. CSC strip layer or MDT tubes layer) is defined as a surface on which all local hit positions are defined. In general local positions are two dimensional as the third free parameter is determined by the constraint of the position to be on the surface. The

surface contains the current geometry parameters from the full ATLAS detector description, including a global position in ATLAS and alignment corrections. Hence, all transformations from local to global frame are done on this surface, as seen in Figure 23.

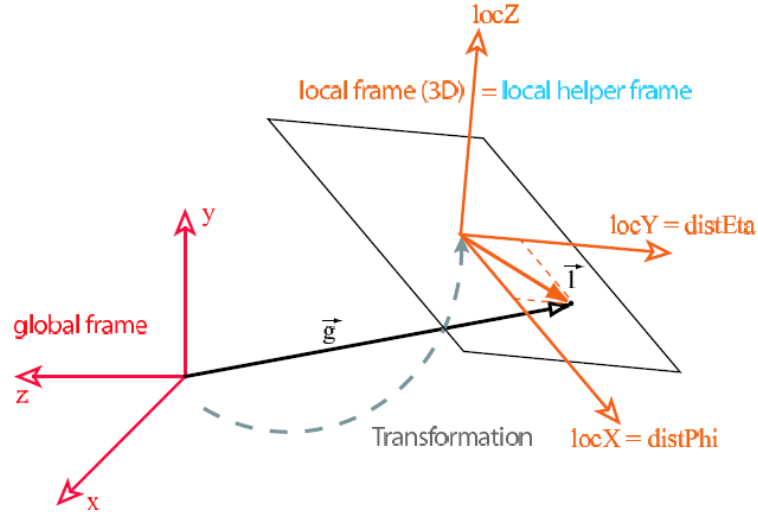


Figure 23: Illustration of the local to global transformations.

The local EDM coordinate frame for the CSC chambers is defined as a 2D frame on a single CSC surface (layer), where:

- **X** : The X position on a surface represents the main surface coordinate (i.e., for a η surface, X would be along the roughly along the η coordinate and for ϕ surface, X would be along the ϕ coordinate).
- **Y** : The Y position on a surface represents the secondary surface coordinate (i.e., for a η surface, Y would be along the roughly along the ϕ coordinate and for ϕ surface, Y would be along the η coordinate).
- **Direction**: A 2D vector perpendicular to the surface.

Each CSC layer is represented by a surface whose origin is the layer center. By combining all four 2D coordinates from each surface a 3D track is formed.

5.3.3 Chamber Coordinate Frame

For convenience purposes and backward compatibility with the testbeam data, the local coordinate system chosen for this CSC tracking algorithm (dubbed *chamber coordinate frame*) differ from that of the EDM local coordinate system. It is defined as follows:

- **X** : The position along the strips. Used for both the precision strips (η strips) and the transverse non-precision strips (ϕ strips).
- **Y** : The layer position.

- θ : Polar angle, defined in the XY plane. Where $\tan \theta = Y / X$.

In effect there are two transverse 2D coordinate frames – one for the η and one for the ϕ strips – and a combination of the two gives a 3D frame. The origin of either frame is the first CSC strip in the first CSC layer for each chamber. Figure 25 illustrates the coordinate frame.

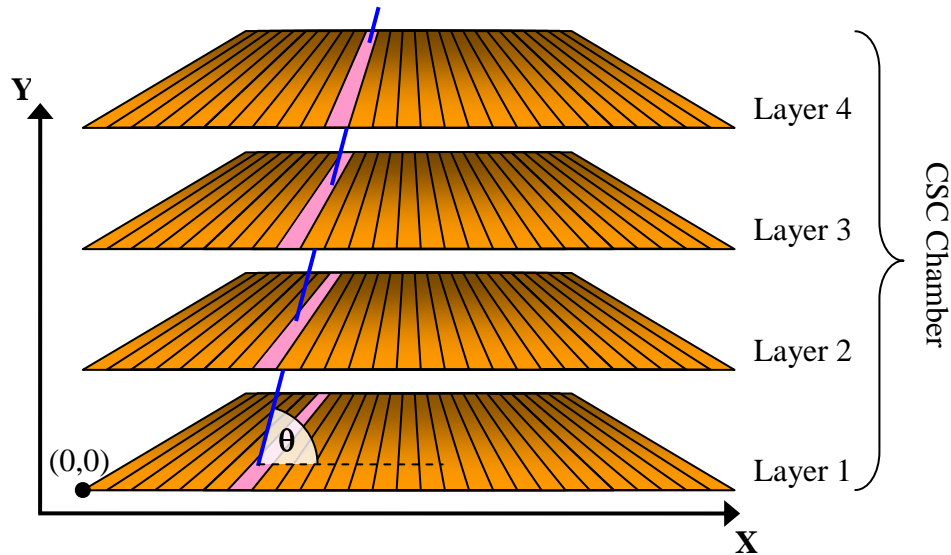


Figure 24: An illustration of the chamber coordinate frame.
 The orange strips are the CSC strips (either the η or ϕ strips). The blue line is a muon track, hitting the chamber in each of the layers. The bright magenta strips represent the hit cluster.

6 CSC Tracking

6.1 Hough Transform

Short track segment identification within a particle detector can be translated to line identification in a noisy image [22]. The detection of straight line-segments in images is a problem that often occurs in image processing. One method for detection of collinear points is related to the Hough transform (HT) [23, 24], in which points in the image are transformed into lines in a Hough parameter space. Those parameter-space lines which correspond to collinear points will cross each other at one point, as seen in Figure 25. This point defines the spatial parameters of the line through the collinear points.

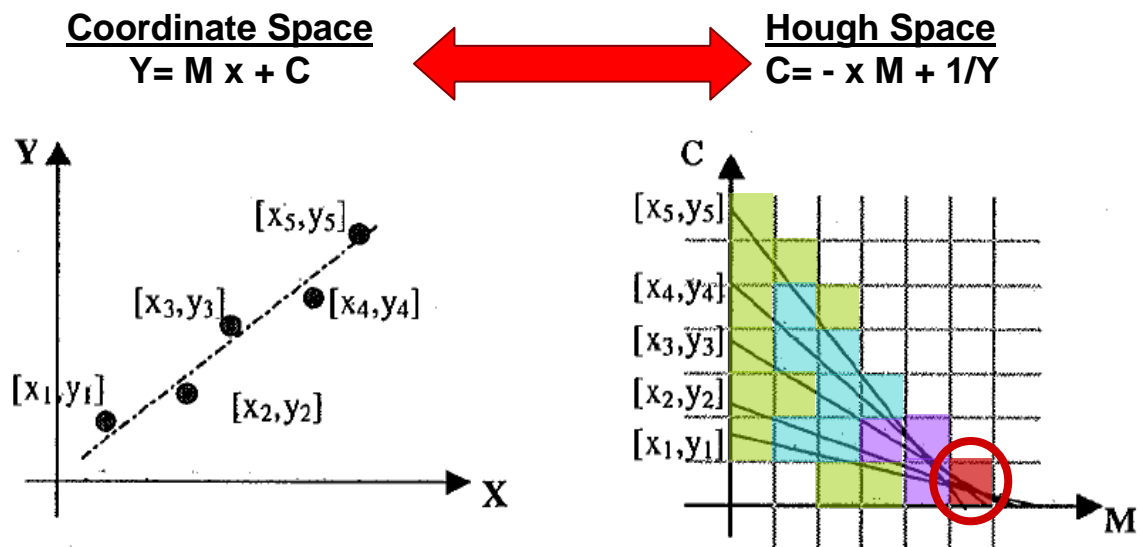


Figure 25: Illustration of the discrete Hough Transform. Points in the coordinate space (left) are transformed into lines in the Hough parameter space (right). Lines in the parameter space corresponding to collinear points will cross each other at one point.

In practice, the parameter space is divided into an array of discrete cells. When a point in the image space is transformed into a line in the parameter space, all cells crossed by the line are incremented. If n points are approximately collinear, the line parameters in the image space correspond to a local maximum in the parameter space, produced by the approximate intersection of n lines. Thus, the Hough transform reduces searching for collinear points in the image to looking for cells in the parameter space which are local maximum. This trait is especially useful in a noisy environment such as the CSCs, where many hit clusters are expected, performing much faster than traditional combinatorial approaches.

6.2 Approach

While the efficiency of the CSC detection of a minimum ionizing particle – such as muon – is close to 100% in each layer, in the presence of a noisy background, the number of recorded hits can be larger than the number of the muon hits. In the background environment of the LHC, one can expect to get number of hits, which is an order of magnitude larger than the number of the interesting muon hits. This obviously leads to major difficulties in reconstruction and tracking. The major reason for these problems is the high background hits found in forward region of ATLAS.

In the CSC chambers it manifests itself in two ways:

- a. High number of hit clusters in each of the CSC layers.
- b. High overlap of background strip signals caused mainly photons and neutrons, with the muon strip signals.

In such an environment track-finding becomes computationally-intensive due to the large number of cluster combinations that result in a track. For that reasons it was chosen to use a procedure of detect-before-estimate that detects the muon tracks (four aligned potential clusters) without applying complex calculations to find the hit clusters. These complex calculations should only be applied as a second phase, only for the muon hit clusters that are clearly associated with a possible track. This approach has high potential to reduce the overall algorithmic complexity.

The main phases of the detect-before-estimate approach are:

1. Activity Detection: Tag hit strips as either *real muon* signal, *masked muon* signal or *background* signal.
2. Crude Track-Finding: Use Hough transform on the real muon hits to find collinear hits in several CSC layers.
3. Clusterization: Find all the hit clusters in a window around the crude tracks and calculate the muon hit position for each layer.
4. Fine Track-fitting: Use a weighted-least-square algorithm to fit the clusters' hit position to a fine track representing the muon track.

Each of these stages is explained in full in the next section.

6.3 The Tracking Algorithm

The CSC tracking algorithm was designed to comply with the ATHENA guidelines in which a clear separation between algorithm and data exists. It was designed to handle data from both ATHENA EDM objects and X5 testbeam objects, perform a Hough-based reconstruction

procedure and provide output in the form of either EDM objects or ROOT Ntuples. A visualization package enables quick debugging of individual events.

A detailed discussion of the algorithm structure will be given in this section, followed by a more technical description of the package structure and usage. A flowchart of the algorithm is depicted in Figure 26.

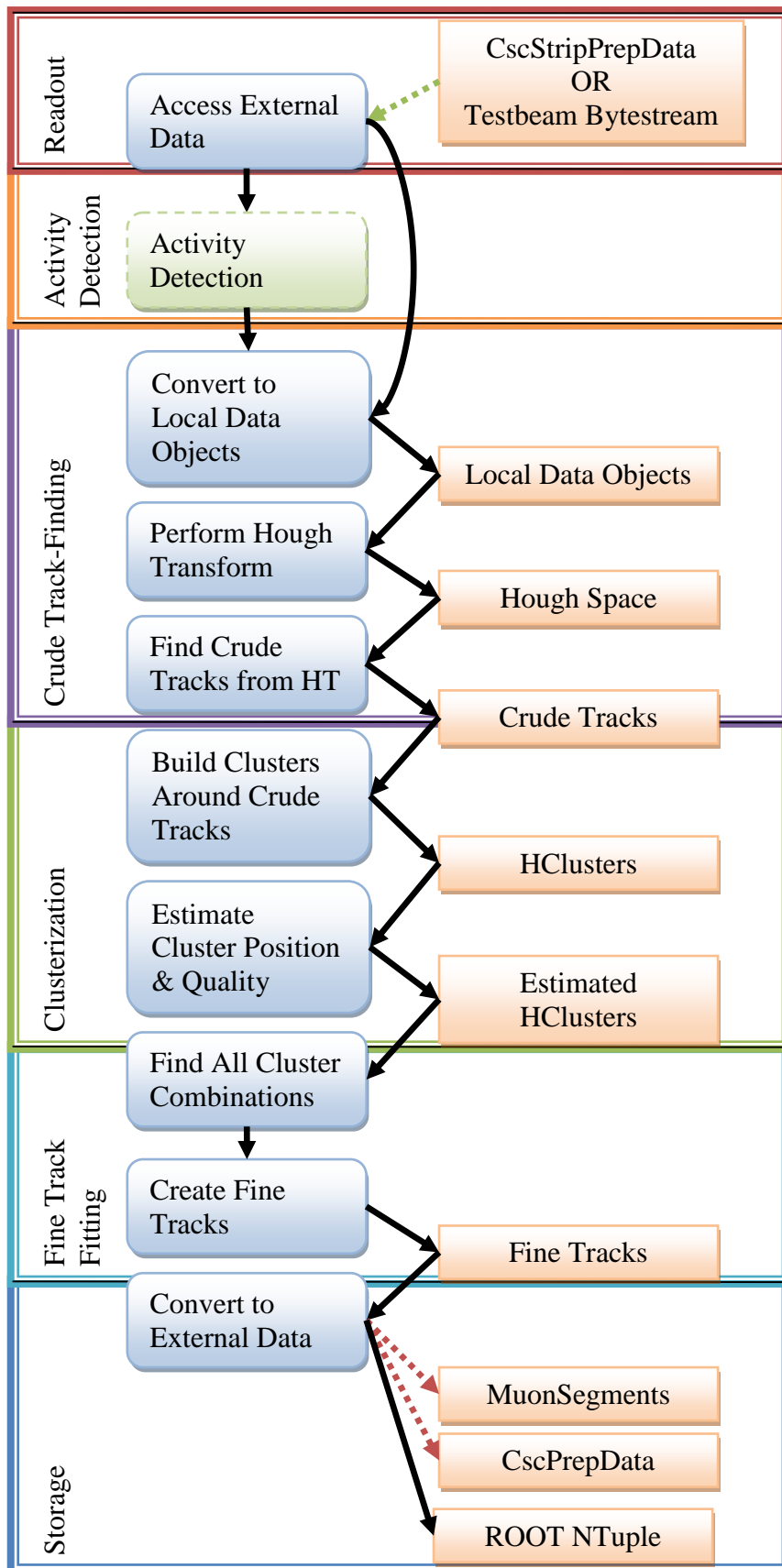


Figure 26: Flowchart of the CSC reconstruction algorithm. Blue rounded squares represent algorithmic operations and orange squares represent data objects. The green dashed rounded square is an operation done only for X5 testbeam data. Green and red dashed arrows respectively represent reading and writing to StoreGate.

6.3.1 Readout

The first stage of the algorithm is accessing data objects. Depending on whether an analysis of the testbeam or simulation data is required, the algorithm accesses either X5 testbeam bytestream data or the ATHENA Simulation `CscStripPrepData` collections.

In the case of the testbeam analysis, the algorithm continues to the Activity Detection (6.3.2) stage, while in the simulation analysis this stage is skipped. For the simulation analysis, a coordinate-frame conversion from the EDM's *local coordinate frame* (5.3.2) to the *chamber coordinate frame* (5.3.3) is performed. Both methods then convert the data to local data objects for further analysis.

6.3.2 Activity Detection

Since the hit time of photons and neutrons background is uncorrelated with the muon hit time, as explained in section 3.4.4, it is usually possible to distinguish between a real muon, masked muon and background muon hit by checking if the strip's bipolar signal peak is within the expected time-window for a muon. This stage tags hit strips as either *real muon* signal, *masked muon* signal or *background* signal by checking the maxima of the 25 time-samples of the bipolar signal.

A *real muon* is any signal whose maximum is above the given threshold and is inside the time-window set for a real muon signal. A *masked muon* is a possible muon signal being masked by a background signal, and is taken as any signal whose maximum is above a given threshold and inside the time-window set for a masked muon (typically any signal coming prior to an expected muon). All the rest are marked *background* signals. An illustration of these types of signals is given in Figure 27.

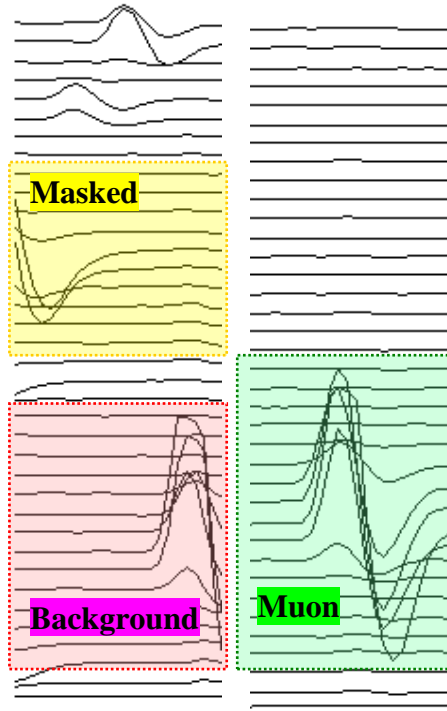


Figure 27: Testbeam bipolar signals of CSC strips. X-axis is the time and Y-axis is the sampled charge. Examples of a muon signal cluster (right), a masked muon cluster (left, top) and a background cluster (left, bottom) are marked in a dotted line.

In the Hough Transform phase *muon strips* and the *masked-muon strips* are used together. A track candidate must have either potential strip or masked strip in each layer.

The activity detection phase is only relevant for the X5 testbeam analysis since the data from all 25 time samples of a CSC strip is only available for testbeam runs. The input from simulation runs already filters out strips with background hits and only contains four time-samples around the bipolar maximum.

6.3.3 Crude Track-Finding

The Crude Track-Fitting stage is meant to be a fast and simple sweep over hits from the CSC chamber, filtering out all non-collinear hits. In this stage the position of each *real muon* or *masked muon* strip is transformed into the Hough Space using the two point form:

$$(2) \quad y - y_1 = \frac{y_2 - y_1}{x_2 - x_1} (x - x_1)$$

which represents a line connecting points (x_1, y_1) and (x_2, y_2) . For the CSC detector, y_1, y_2 are both known (the layers location), so it is possible to implement a transform from any point (x, y) of the track to a straight

line in the parameter space using (x_1, x_2) as the only two unknown parameters. This formulation allows putting constraints on the tracks (i.e. applying an angle constraint, in which the transform will be applied only for lines roughly projected to the interaction point).

The Hough space is a discrete form of the continuous transform, i.e. a matrix of Hough Cells. Each cell in the Hough parameter space has three values:

- The number of *muon strips* whose transform cross the cell.
- The number of *masked-muon strips* whose transform cross the cell.
- The strip charges associated with the cell.

When a line representing a point in image space crosses a cell in the Hough space, that cell's values are incremented according to the originating image point. That is to say, a muon strip (masked muon strip) hit crossing a cell will increment its *muon strips* value (*masked-muon strips* value) by 1 and its charge value by the charge of the strip.

The first and second values are used for finding the local maximum above a predefined threshold. Then, the third value is used for filtering the parameter space by selecting the lines with the maximum charge sum.

Figure 28 illustrates the transform of four collinear points in an image to the Hough parameter space. This is a plot of the *muon strips* value of each Hough Cell.

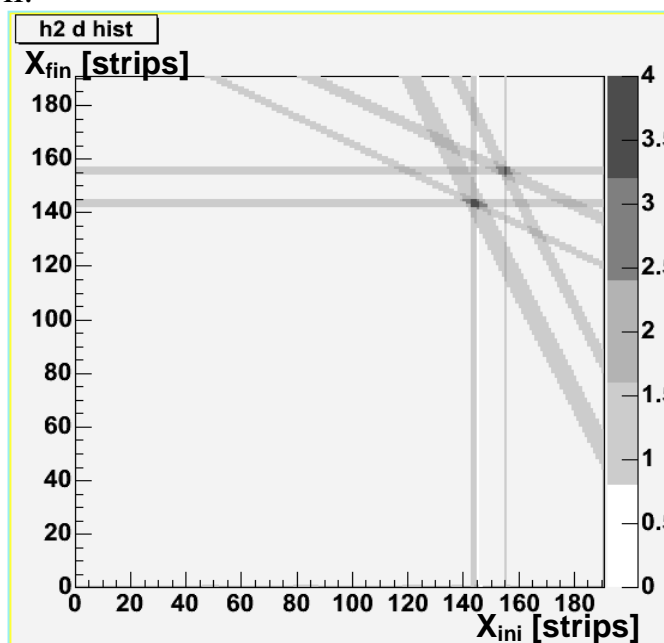


Figure 28: Hough parameter space for the four CSC layers, taken from the CSC_DHough program. “*Muon strips*” value is presented. Maximum areas correspond to collinear “real muon” hits in each of the four layers of the chamber. Axis are the coordinates of the hit position in the first and last CSC layers.

Applying a threshold algorithm to the first two values allows the track-selection of two, three, or four collinear points, depending on the user.

Since a muon hit in the CSC layers creates a cluster of active strips, there are number of possible collinear points in each of the four layers, and therefore a number of possible tracks. The track representing the real muon track is that connecting the maxima of these clusters. For that reason, the Crude Track is taken as that crossing the maximum number of collinear points (depending on user threshold) and the maximum cumulative charge (As seen in Figure 29).

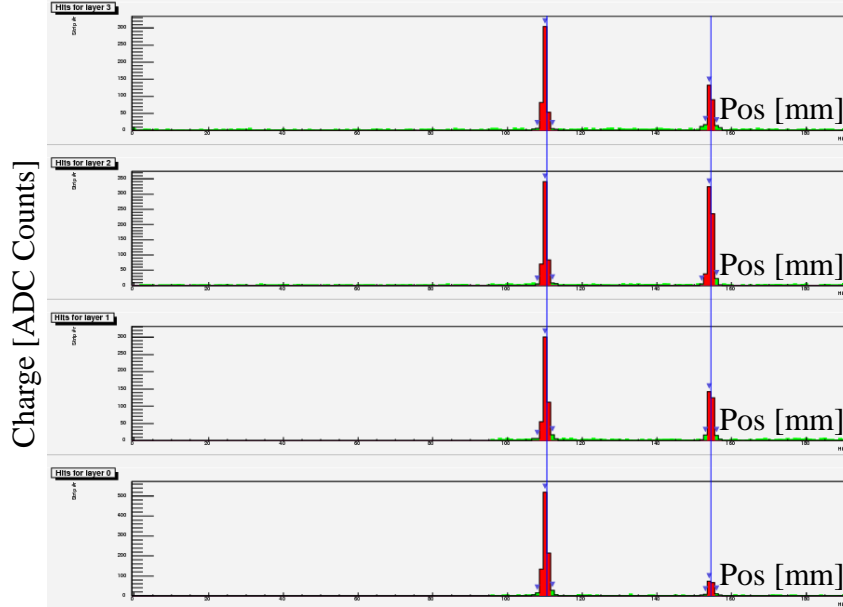


Figure 29: A view of the muon charge cluster in the four CSC layers.

The X-axis is the number of CSC strip and the Y-axis is the charge in units of charge counts. The blue lines represent two Crude Tracks crossing through the maximum cumulative charge.

6.3.4 Clusterization

During the clusterization stage, clusters are formed around the Crude Tracks and then evaluated. This phase is purposefully vague in order to allow different users a great deal of control over the cluster formation process, which is the heart of the CSC tracking algorithm.

The form of the charge distribution in a cluster is given by [25]:

$$(3) \quad C(x/d) = K_1 \frac{1 - \tanh^2(K_2 x/d)}{1 + K_3 \tanh^2(K_2 x/d)}$$

Where x is the precision coordinate (transversely to the strips) and d is the anode-cathode separation. K_2 and K_3 are related to the empirical formula:

$$(4) \quad K_2 = \frac{\pi}{2} \left(1 - \frac{1}{2} K_3^{1/2} \right)$$

Using the constraint that that the total charge induced on one cathode equals half the avalanche charge gives rise to this relation:

$$(5) \quad K_1 = \frac{K_2 K_3^{1/2}}{4 \tan^{-1} K_3^{1/2}}$$

Thus (3) can be reduced to a one-parameter expression, called the Mathieson distribution (See Figure 30).

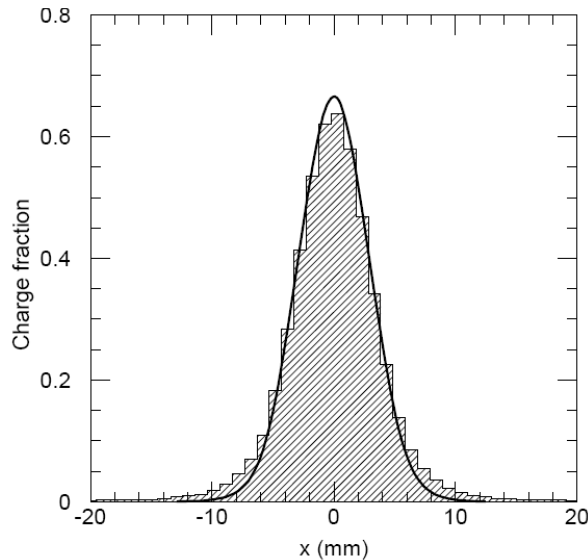


Figure 30: Mathieson Distribution.
Data from measured charge distribution in CSC prototypes, fit with the Mathieson Distribution.

Several problems needed to be characterized in order to allow the versatility required from the algorithm. Interaction of the muons with the detector material that may cause the creation of secondary particles, the inefficiency of strip channels, and other electronic phenomena such as overflow and crosstalk, can all contaminate the ideal structure of the charge distribution over the detector strips. The sum of these problems resulted in a different cluster distribution than the expected ideal Mathieson distribution. Thus, the cluster position error can no longer assumed to be as an error of an ideal cluster. Instead, clusters were sorted into “good” and “bad” clusters, corresponding to clear or contaminated cluster structure. *Good* and *bad* clusters will be treated differently in the Fine Track-Fitting stage.

Cluster building is done in several steps:

1. Space-Window:

A space-window opens around each of the crude tracks and active strips in this window are grouped together in each layer. These groups of strips are later scanned for possible clusters.

2. Peak Selection:

The individual groups of strips are scanned for local maxima, designating each as a peak. These peaks will be the seeds around which clusters will be built.

3. Cluster Formation:

Each peak found in the previous step serves as a seed around which a cluster is formed. Clusters with several peaks representing several hit points are possible, but are usually evaluated as bad clusters. A versatile set of cluster conditions was imposed to better characterize good and bad clusters:

- Clusters were defined to have a minimum number of *good strips* (*real muon strips* or *masked muon strips*), a maximum number of *bad strips* (*background strips*) and a maximum number of *empty strips* (ones in which no signal is present) in them. These conditions were added to allow a cluster in which one or more strips are noisy or not functioning.
- A parameter representing the maximum number of cluster strips allowed per cluster prevents clusters from being too wide.
- A charge-threshold on the cluster strip was added to prevent long cluster tails due to electronic noise.
- Two nearby muon hits will create a double Mathieson distribution which needs to be analyzed differently than a single-peaked distribution. A parameter for maximum peak-to-peak distance was added in order to take that into account. Clusters with peaks or larger distances will be broken into two clusters and treated as a single Mathieson distribution.

The cluster formation process starts from these peaks, trying to add nearby strips to them, as long as the restrictions are upheld. Once a restriction is broken, the cluster structure is set.

4. Cluster Position Estimation:

The cluster positioning estimation can be done in several ways; some are easy to implement but less accurate, and some are more accurate but are also computationally-intensive. The four algorithms that were considered are:

- The **Ratio algorithm** [26] estimates the hit position using the ratio between the charge difference between the maximum valued strip and its neighbors as describes in (6)-(8).

$$(6) \quad r = \begin{cases} (p - p_L)/(p - p_R) & p_L > p_R \\ (p - p_R)/(p - p_L) & p_L \leq p_R \end{cases}$$

$$(7) \quad x_f = \Theta(p_L - p_R) \cdot \begin{cases} A_0 + rA_1 + r^2A_2 + r^3A_3 & : r < 0.5 \\ B_0 + rB_1 + r^2B_2 + r^3B_3 & : r \geq 0.5 \end{cases}$$

$$\begin{array}{l}
\text{Where:} \\
A_0 = .501102 \quad B_0 = .247930 \\
A_1 = -.299817 \quad B_1 = .818883 \\
A_2 = -.0339802 \quad B_2 = -1.66454 \\
A_3 = -.172102 \quad B_3 = .597592
\end{array}$$

$$(8) \quad x_{pos} = w_s x_f + x_{peak}$$

p , p_L and p_R are the charge values of the peak strip and its neighbors to the left and right. r is the ratio between them and x_f is the peak offset in units of strip-width after applying non-linear corrections for the Mathieson distribution. A and B are vectors of parameters optimized for the ratio peak-finding of the Mathieson distribution. x_{peak} is the cluster's peak position and x_{pos} is the new ratio-evaluated cluster position.

- The **CoM algorithm** [9] is based on center-of-mass formula:

$$(9) \quad x_{pos} = \frac{\sum_{i=1}^{i=3} x_i p_i}{\sum_{i=1}^{i=3} p_i}$$

Where p_i is the strip charge and x_i is the strip position. Like the ratio algorithm, the CoM algorithm takes into account the maximum strip and its left and right neighbors.

- The **Parabola algorithm** uses a parabolic estimation to the three points around the cluster peak:

$$(10) \quad r = 0.5 \cdot (p_L - p_R) / (p_L + p_R - 2p)$$

$$(11) \quad x_f = A_0 \cdot \tan^{-1}(A_1 x) + x A_2$$

$$A_0 = 0.324610$$

$$\text{Where:} \quad A_1 = 8.19201$$

$$A_2 = 0.136329$$

p , p_L and p_R are the charge values of the peak strip and its neighbors to the left and right. r is the estimated parabolic fit x_f is the peak offset in units of strip-width after applying non-linear corrections for the Mathieson distribution. A is vectors of parameters optimized for the parabola peak-finding of the Mathieson distribution. x_{pos} , given by (8), is the new parabola-evaluated cluster position.

- The **Maximum Likelihood** algorithm estimates the cluster's hit position by performing a maximum-likelihood fit of cluster strips using the theoretical Mathieson distribution plus a fraction of random Gaussian background.

The total charge induced on strip n in the cluster is modeled as:

$$(12) \quad y(n) = aS(n; x_p) + r(n)$$

Where x_p is the hit position, a is amplitude variable and $S(n; x_p)$ is the normalized Mathieson distribution over the cluster strips. $r(n)$ is the random Gaussian background.

The two unknown parameters, a and x_p , of the Maximum Likelihood are derived in the following way:

$$(13) \quad (\hat{a}, \hat{x}_p) = \arg \min_{(a, x_p)} \sum_{n=0}^{n=N-1} |y(n) - aS(n-i; x_p)|^2$$

Where i is the offset between the signal from the n^{th} measured cluster strip $y(n)$ and the signal expected from the n^{th} Mathieson strip $S(n; x_p)$.

The solution for any x_p is:

$$(14) \quad \hat{a} = \frac{\sum_{n=0}^{n=N-1} y(n) \cdot S(n-i; x_p)}{\sum_{n=0}^{n=N-1} S(n-i; x_p)^2}$$

By substituting (14) into (13)

(13), the solution for x_p is obtained:

$$(15) \quad \hat{x}_p = \arg \max_{x_p} \frac{\left(\sum_{n=0}^{n=N-1} y(n) S(n-i; x_p) \right)^2}{\sum_{n=0}^{n=N-1} S(n-i; x_p)^2}$$

The algorithm calculates x_p by running over possible i for offset between signals and finding the one with the maximal value of (15).

5. Cluster Quality Estimation:

In the cluster quality estimation, each cluster is assigned a number representing its quality. Cluster quality will range from 1 for very *good* clusters and 0 for very *bad* clusters. In the fitting stage clusters can then be excluded based on their individual quality value.

There are two possible quality classification methods:

- The **Simple Quality** was designed to be a computationally-rapid method, which gives a quality of 1 for every cluster that follows these restrictions:
 - Single peak.
 - Number of cluster strips under a user-defined threshold (Mathieson distribution in a CSC cluster ideally results in five to six active strips).
 - Closest nearby cluster is further away than a user-defined threshold. The exact evaluation of the hit positions originating from two close clusters with a double Mathieson distribution is computationally-intensive and often not as accurate as well-separated clusters.All other clusters get a quality of 0.4.
- The **Maximum Likelihood Quality** uses the inverse of the sum of the residuals between the cluster's signals and the theoretical Mathieson distribution for this cluster as calculated in (15). The more similar the cluster signal is to the expected Mathieson signal, the smaller the sum of residuals would be and the higher the quality would be.

6.3.5 Fine Track-Fitting

Track fitting that uses the traditional least squares (LS) method takes the same ideal position error for all clusters, and thus it loses its optimal properties on contaminated cluster data. A suggested track fitting technique, which calculates the cluster quality and classifies the clusters into “*good*” and “*bad*” clusters, was examined. It was shown [27] that fitting algorithm that uses only the “clean” clusters, resulted in better performance than other methods such as traditional least squares (LS), weighted least squares (WLS), and robust fitting.

However, to allow versatility, a combination of “*good*” and “*bad*” clusters is also allowed for track fitting purposes. Each cluster is assigned a quality parameter which is used in the track-fitting stage to either filter

the cluster out (when its quality is too low) or include the cluster in the fit if the cumulative cluster quality of all clusters in track is above a user-defined threshold. By changing the cumulative cluster quality parameter it is possible to control the number of “good” and “bad” clusters which are considered for the fitting. All valid combinations are taken and fitted into a straight line using the LS method. In the case of several nearby tracks, the one with the least χ^2/DoF was taken.

6.3.6 Storage

The final stage of the track reconstruction algorithm is persistifying the tracking data in the form of EDM objects, as well as keeping important results in an additional ROOT NTuple. The EDM objects used for tracking are the `MuonSegment` (See 5.2) used for track segments and `CscPrepData` used for CSC clusters.

Conversion from the local tracking objects to the EDM objects requires an anti-transform of the measurement frame from *chamber coordinate frame* to the *local coordinate frame*. Once this is complete, a `MuonSegment` object is formed, containing a list of the `RIO_onTrack` objects which were used to create it. A list of `CscPrepData` objects representing the unfitted clusters is saved separately for the purpose of other track-fitting algorithms (such as `MuonBoy`).

Both these objects are collected into groups and recorded into the Transient Data Store, `StoreGate`, for further use.

6.4 Software Structure

6.4.1 The converters

The core algorithms do not depend on any other ATHENA packages in the ATLAS CMT environment. The algorithms can be used with several data input from different sources. The dependency on the data input structure exists only in the interfaces classes; thus, the core algorithms are independent of the data structure. Figure 31 describes the idea of the separation between the core algorithms and the data input. There are currently three types of data input: data from the X5 testbeam, data from ATHENA simulation, and data from a ROOT file. The converter classes are used to convert the data into internal data objects which are used by the track finding and fitting algorithms. In order to run the algorithms faster with ATHENA data input, it is possible to run ATHENA once and dump all event information into an external ROOT file, `Events.root`. This ROOT file can be used outside of the ATHENA environment in order to debug the algorithms faster. For debug purposes, special event displays

(based on ROOT classes) were implemented. In the ATHENA environment, other converters are used to convert the internal data objects into ATHENA objects (such as MuonSegment, RIO_OnTrack, see sections 6.3.1 and 6.3.6).

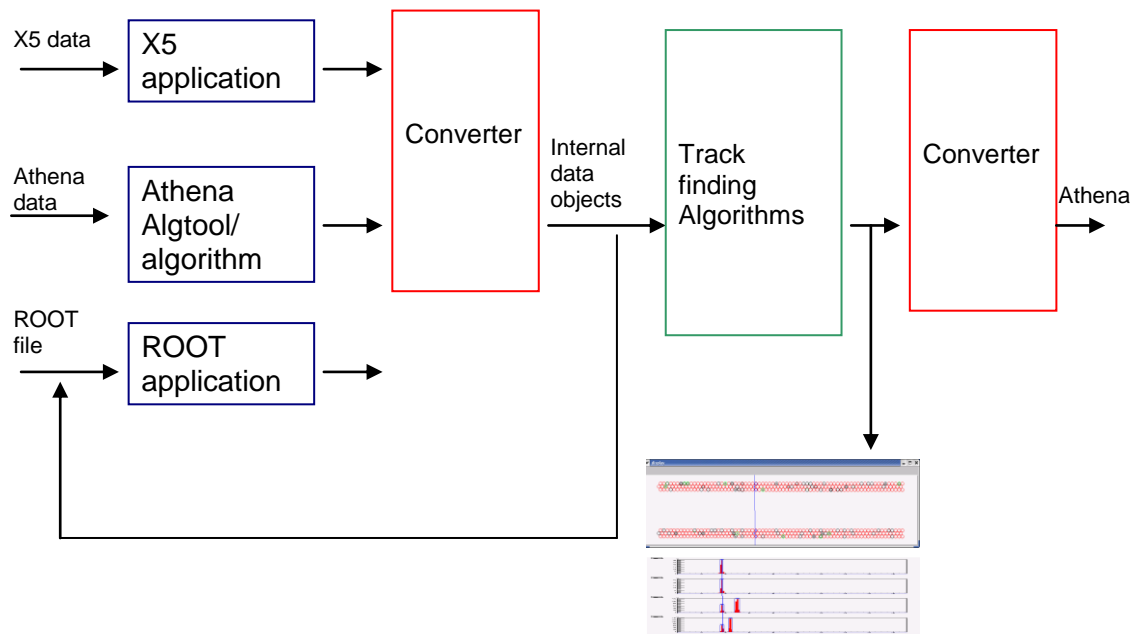


Figure 31: Data converters usage in the track finding algorithm.

6.4.2 Packages structure in ATHENA

All packages are located in the ATLAS offline CVS. The core algorithms are located under

`MuonSpectrometer/MuonReconstruction/MuonRecUtils`. The ATHENA tools and algorithms (which are the ATHENA interfaces to the core algorithms) are located under

`MuonSpectrometer/MuonReconstruction/MuonSegmentMakers/MuonSegmentMakerAlgs/` for ATHENA algorithms, and

`MuonSpectrometer/MuonReconstruction/MuonSegmentMakers/MuonSegmentMakerTools/` for ATHENA tools. The core packages are divided into a common non-specific package for both CSC and MDT detectors, and specific CSC and MDT packages, as described in Figure 32:

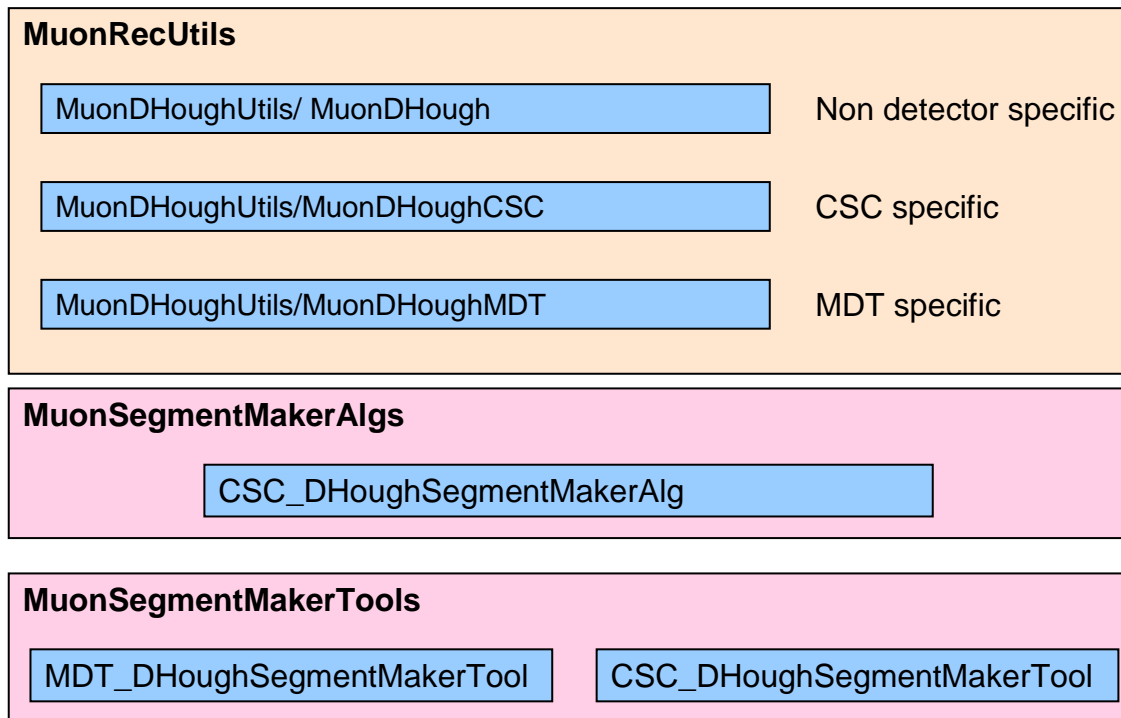


Figure 32: CSC and MDT Reconstruction packages structure in the ATHENA framework.

6.4.3 Packages description

6.4.3.1 Common packages

There is only one common package for CSC and MDT detectors. It holds the core Hough transform algorithm, and interfaces for specific CSC and MDT classes.

MuonDHough

This package contains the core Hough transform algorithms, including the Hough space data structure, the Hough filling classes and the basic algorithms for the Hough space (such as local maxima finding, comparison to threshold, etc).

6.4.3.2 CSC packages

CSC_DHough

This is the main package containing the bulk functionality for the CSC data analysis. It is being called to by either CSC_DHoughRoot if the input is a ROOT file, from CSC_DHoughTestBeam if the input is testbeam data, or from CSC_DHoughSegmentMakerAlg if the input is simulation CscStripPrepData data from StoreGate.

It uses the MuonDHough package to perform a single Hough Transform on the CSC hits and finds crude tracks in the CSC chambers. All the clusters in the vicinity of those crude tracks are then found and registered

and their position and quality are calculated using various methods. Using the cluster position and quality a fine track is found using LS fitting. The crude tracks, clusters and fine tracks data are all available as output, which is passed on to the calling algorithm.

CSC_DHoughClusterML

A package that performs Minimum Likelihood position and quality estimation for a single cluster.

CSC_DHoughRoot

ROOT analysis of CSC data using the ROOT file as input. It translates the Events.root file into the internal data structure, and calls the core reconstruction algorithms. A ROOT event viewer may be called to visualize the CSC layers, hits, clusters, crude tracks and fine tracks. The purpose of this package is to provide the same environment for testbeam and simulation data, as well as faster standalone analysis for debug purposes.

CSC_DHoughDisplay

A ROOT display package taking data from CSC_DHoughRoot and writing it into a ROOT canvas.

CSC_DHoughTestBeam

Test-beam data analysis of CSC data using test-beam output files as input. The output is then saved to a ROOT file for future display.

CSC_DHoughSegmentMakerAlg

The main algorithm for the ATHENA framework. It converts a collection of CscStripPrepData to internal objects, performs the CSC analysis and saves the resulted tracks into StoreGate as MuonSegment collection.

CSC_DHoughSegmentMakerTool

A supplementary tool that takes a road as input and returns all CSC MuonSegments lying inside this road.

6.5 Analysis

The ATLAS reconstruction environment is under continuously extensive development, with frequent changes in interfaces, package functionality and EDM structure. The CSC software package, as part of it, is also under development. As better simulation data becomes available, it is important to test the reconstruction against it and tweak it accordingly.

For this reason, a great effort was invested in designing the package to be versatile and allow a wide range of operation without considerable change to the code. The package contains a long list of batch parameters (detailed in Appendix 1.1.1) in the form of jobOption file. Many studies have been done on different simulation data using various parameter schemes. These studies are a part of a continuous effort to improve the algorithm results, and are in not yet final. A selected number of examples will be given in this section.

The first stage of the algorithm utilizes the Hough transform to accept only collinear hits above a user-defined threshold. Due to geometric differences, the acceptance parameters (detailed in appendix 0) are different for the η and ϕ strips, and the rejection is done independently. Figure 33 shows the percentages of accepted and rejected strips out of all CSC strips over 2000 single-muon events. Since the left plot corresponds to hits in the η plane one sees an asymmetry caused by excess of hits in higher η region.

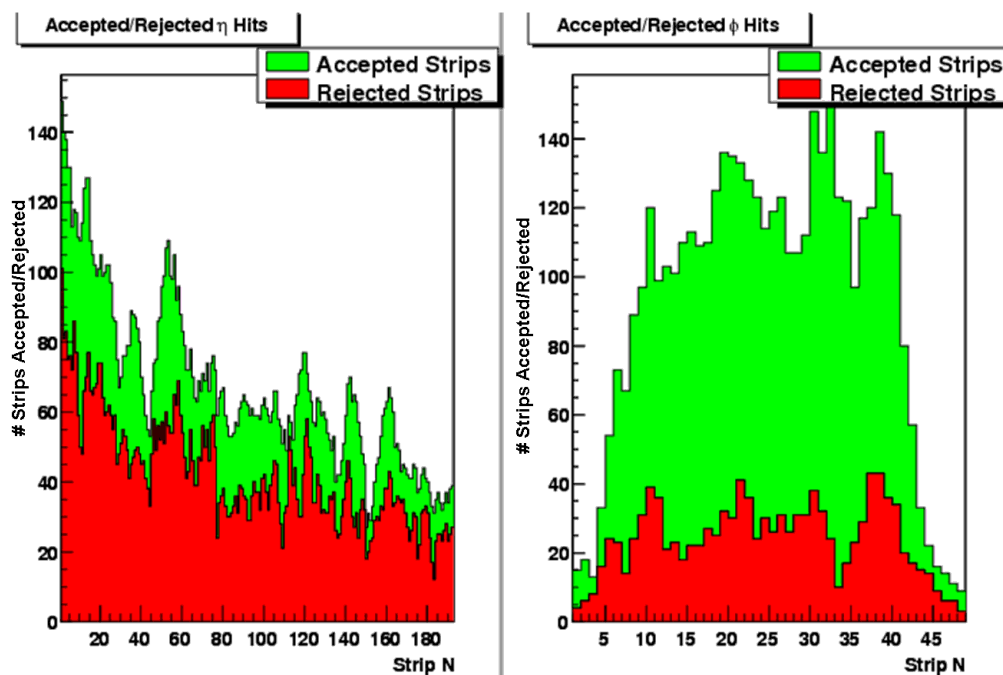


Figure 33: Accepted and rejected strips for the η and ϕ planes of the CSC chambers. Rejection cuts on ϕ are much looser than on η .

The normalized form of this plot, showing the acceptance percentages of hits in η and ϕ planes is shown in Figure 34. It is clear from this plot that the rejection cuts employed by the Hough-Transform are uniform. The ϕ -strips acceptance rate is considerably higher than the η -strips one due to the poorer resolution of the ϕ strips. While η clusters show a good four to six active strips per hit, their ϕ counterparts have a one or two.

This makes it more difficult for the Hough Transform to distinguish between real tracks that have multiple collinear strip clusters to a fake track passing through three or four fired strips. The poor resolution of the ϕ strips is the reason for not including them in the current form of the tracking algorithm.

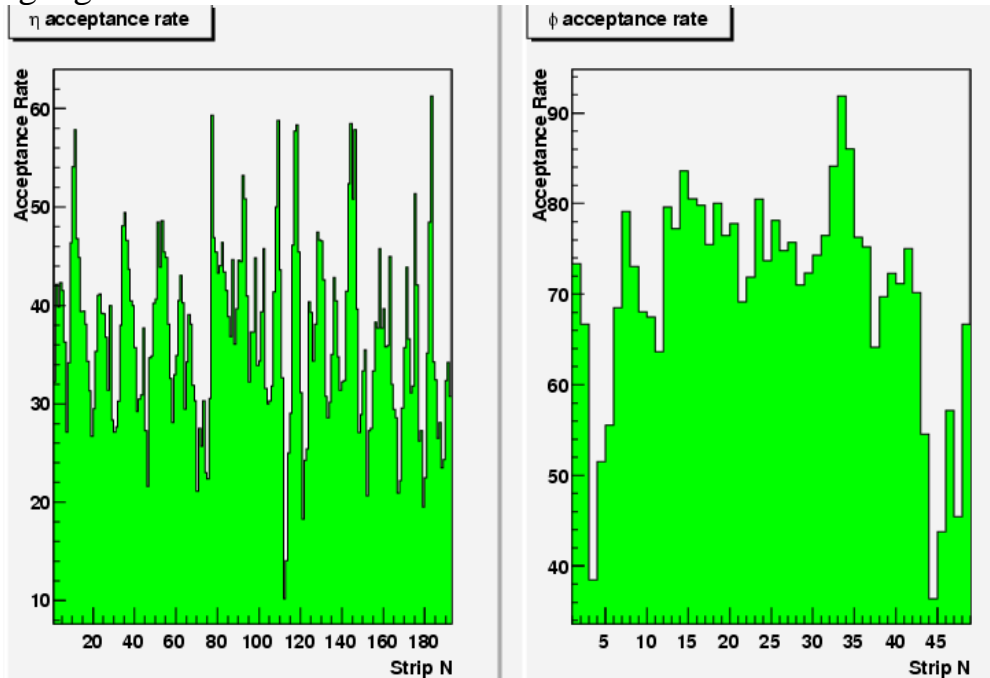


Figure 34: Hit acceptance rates after the Hough-Transform, averaged over strip number.

The Hough Transform algorithm contains numerous parameters that make it very versatile. To optimize the behavior of the algorithm, selected number of parameters were changed to measure their effect. These parameters are:

1. **Hough Space Size:** Number of cells in the Hough space. A transform with large cells will produce imprecise track, while that with very small cells will create a large overhead and possibly miss the track altogether (If the coordinate-space points are not precisely collinear. See Figure 25). [Range tested: 48x48-384x384 cells for η strips. 12x12-96x96 cells for ϕ strips].
2. **Hough Line Resolution:** In a discrete Hough transform, a line in the Hough space is sampled in discrete points and the values of these points' corresponding cells are incremented. Line resolution represents the number of points sampled in a distance that corresponds to one cell length. [Range tested: 1-10].
3. **Hough Value Threshold:** The Hough transform algorithm is designed to find collinear points. The value threshold is the minimum number of collinear points to look for. Since the CSC chambers consist of four layers, we are interested in any track originating from three or four collinear hits. [Range tested: 3-4].

4. **Hough Local Max Region:** Since there are a few active strips per each cluster, it is possible to get a range of collinear strips in a region of Hough space. To get the optimal track parameters, only the Hough cell with the maximal cumulative charge is regarded as the real track. This parameter represents the region of this local charge maxima search. Smaller regions will allow closer track-finding but will also create a large number of fake tracks. [Range tested: 1x1-6x6 cells, corresponding to 1-6 adjacent strips].
5. **Hough Transform Max Angle:** The muons originating from the interaction point hitting the CSC chambers are expected to be relatively orthogonal to the CSC plane. To prevent a high number of fake tracks the search tracks with large angles was limited with this parameter. [Range tested: ± 0.001 to $\pm \pi/2$ radians].

The acceptance rates, averaged over all strips, of ϕ and η planes were examined in various cuts shown in Figure 35.

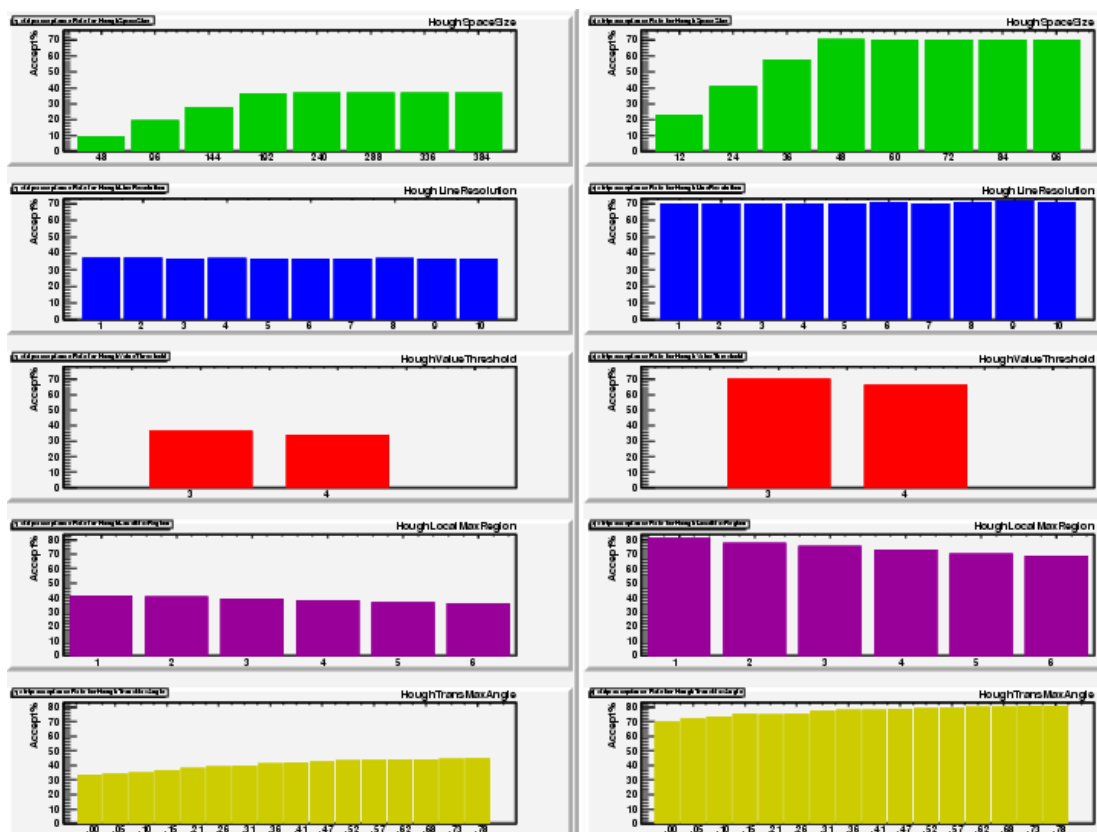


Figure 35: Change in acceptance rates for η (right) and ϕ (left) planes, using various parameter cuts.

The first plot (green) shows the acceptance rates for various Hough Space sizes. The second plot (blue) shows the acceptance rates for difference resolutions of the Hough Space. The third plot (red) shows the acceptance rates for collinear strips in 3 and 4 layers. The fourth (purple) plot shows the acceptance rates for increasing size of the Hough block in which local maxima are found. The fifth (yellow) plot shows the acceptance rates for different cuts on the Hough track angle. This is used for optimizing the parameter space of the Hough CSC algorithm. Note that the graphs axis are on top.

The first plot makes it clear that using Hough spaces smaller than 192x192 for η and 48x48 for ϕ strips (the sizes of the CSC layer) results in under-sampling and poorer performance. The second plot shows a relatively uniform response of the number of samples-per-cell chosen to represent a line in Hough space, so only one sample can be used. The results from the third plot show that there are only a few ($\sim 3\%$) more three-layer collinear hits than four-layer collinear hits. The fourth plot shows that there are $\sim 5\%$ η tracks and $\sim 12\%$ for ϕ tracks that fall inside the 1-6 strips range of another track. The difference is probably due to the different size of η and ϕ strips. The fifth plot shows the expected decline in acceptance rate as the opening angle gets smaller. It is also shown that 76% of the found η tracks and 86% of the found ϕ tracks are completely orthogonal to the CSC plane, so a tight cut on track angle is justified.

A second method to quantize the effect of the Hough parameters employed the use of efficiency versus fake-rates plots. A selection of 500 of the previous events was examined and the number of real tracks in each event was evaluated. This was compared to the number of tracks found by the Hough transform to calculate the efficiency:

$$(16) \quad E = \min\left(1, \frac{\# \text{ measured tracks}}{\# \text{ real tracks}}\right)$$

and fake rate:

$$(17) \quad F = \max\left(0, \frac{\# \text{ measured tracks} - \# \text{ real tracks}}{\# \text{ real tracks}}\right)$$

per event. The averaged result on all events is given in Figure 36.

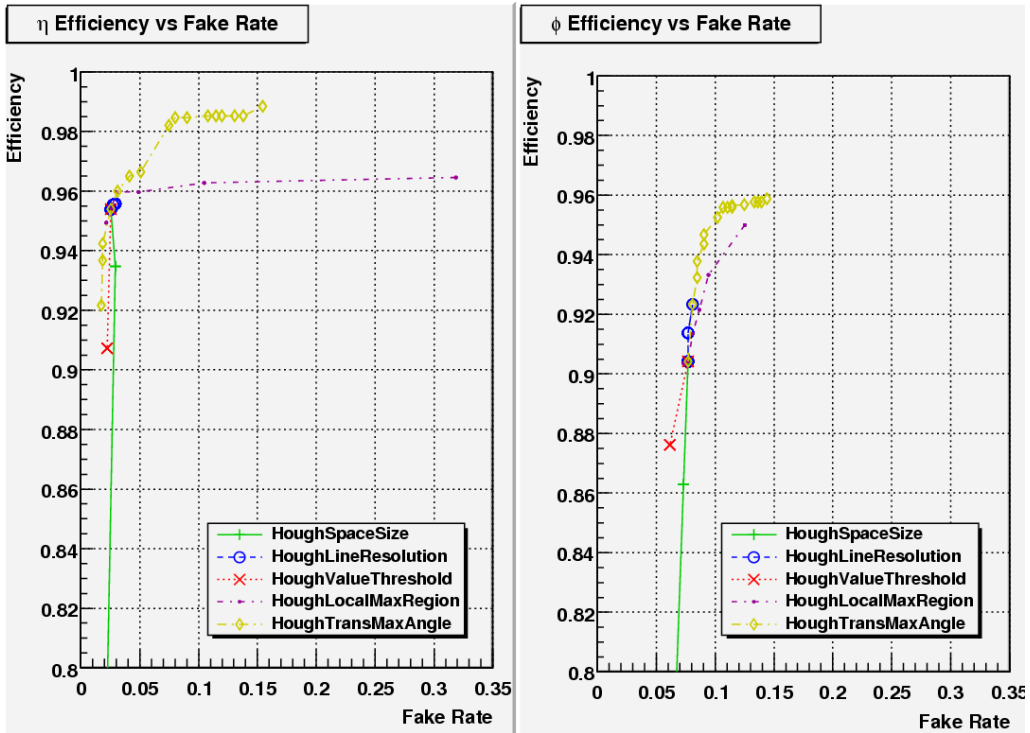


Figure 36: Efficiency as a function of fake rate for η (right) and ϕ (left) planes. The various parameter cuts are given in the legend.

These plots again show that small Hough space size causes under-sampling of the coordinate space and lower efficiencies, but gives no great advantage to larger space sizes, so the optimal sizes equal to the number of strips in either plane can be safely employed. Similar to the previous plot, the Hough line resolution is shown to have little effect. Choosing only four collinear hits produces poorer results than allowing three collinear-hits tracks. Choosing a very small Hough local maximum region produces results with slightly better efficiency, but considerably higher percentage of fake tracks. Since muon clusters are expected to be four to six strips wide in the η plane, a similar number can safely be chosen for the Hough local maximum region. The track angle cuts show a significant rise in efficiency up to 0.15 radians, where any further rise in efficiency is accompanied by a rise in fake tracks. Together with the numbers from the previous plots a range of 0.15 to 0.21 radians can be chosen.

It is important to note that while in the η plane the tracks were clear and obvious to the naked eye, it was exceedingly difficult to locate them in the ϕ plane – a fact that is noticeable in these plots as well.

After the Hough Transform stage, the accepted strips (and possibly some of the rejected strips, depending on clusterization parameters) are grouped into clusters for the next stages of the algorithm. A cluster

representing a muon hit will have the expected form of the Mathieson distribution (3). In Figure 37 the maximum strip in a cluster charge is shown to be following the expected Landau distribution of a muon traversing matter. Since the peak of the Mathieson distribution is relative to the total charge in the cluster, it is possible to use the maximum hit in lieu of the cluster charge.

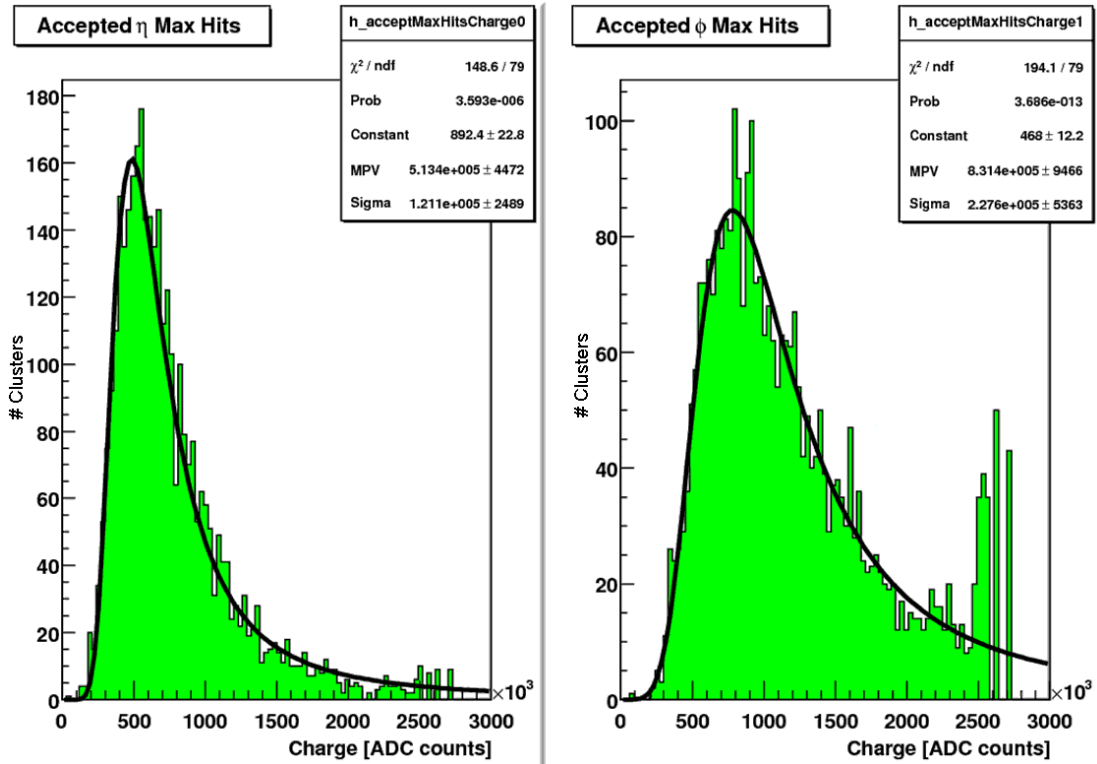


Figure 37: Distribution of the maximum charge in a cluster.

A sample of a reconstructed simulated event in the CSC is shown in Figure 38.

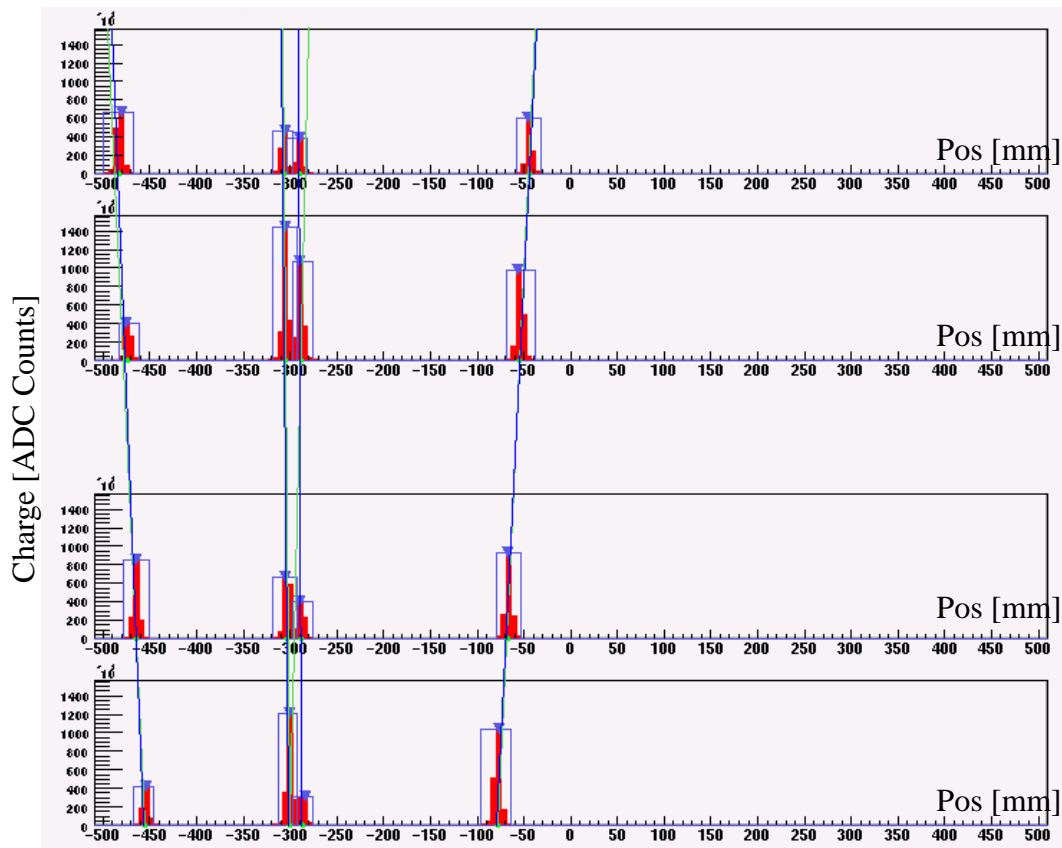


Figure 38: ROOT Event Viewer of a CSC chamber. Four simulated muons traversing the CSC chamber, creating tracks. The dark blue lines are the reconstructed fine tracks and the light green tracks are the Hough crude tracks. The blue frames around each hit cluster represent the reconstructed cluster and the blue arrows represent the reconstructed hit position for each cluster. The hit position in each of the four layers is shown. X axis is the η -strip position in mm and the Y axis is the strip charge-count.

A long study of the strip-positioning data was conducted, due to problematic residuals in the reconstructed tracks. After the reconstruction process was done, residuals of the fine tracks were taken for each of the four CSC layers (shown in Figure 39). The residuals were plotted separately and showed a very large distribution width in comparison to the expected $60\mu\text{m}$ CSC resolution. Additionally, an asymmetry between the two middle layers was apparent, corresponding with a position-shift.

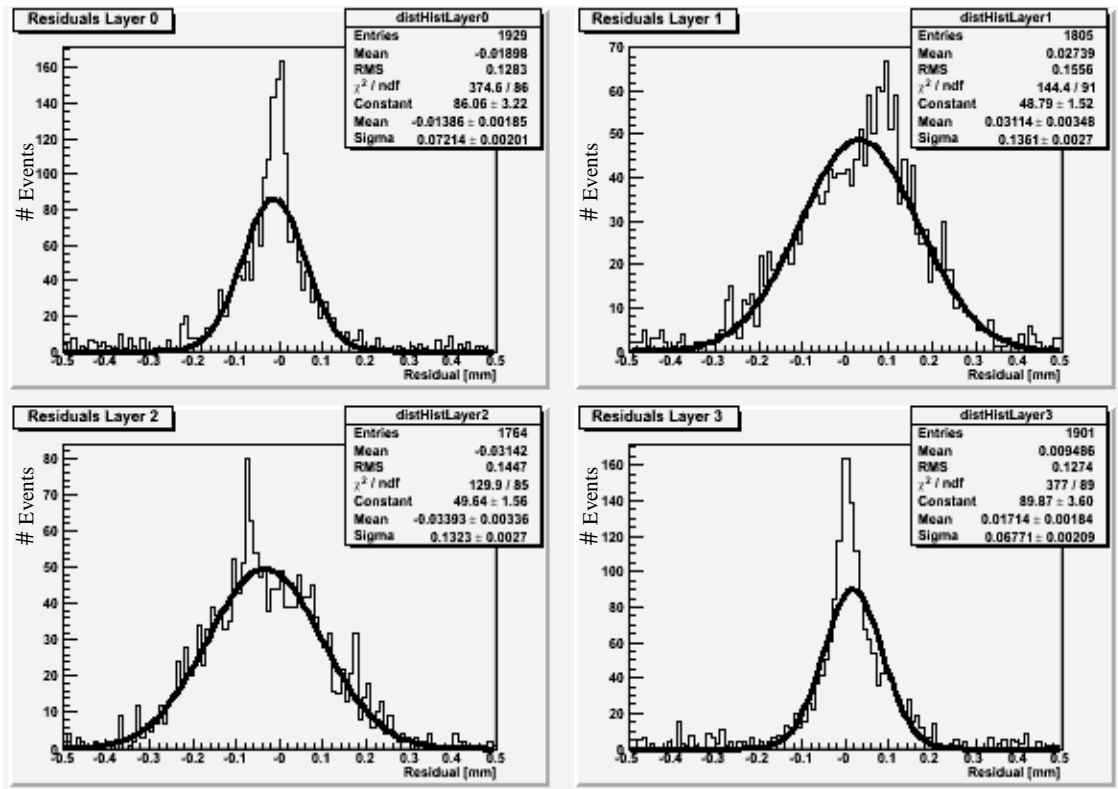


Figure 39: Fit residuals in each of the CSC layers. Asymmetry in 1st and 2nd layer and large σ is caused by faulty simulation position data.

Further validation showed indeed a shift of $\pm 2.5\text{mm}$ in the strip position of the EDM data object, which was fixed in later versions.

The importance of cluster classification to good and bad clusters, as discussed in section 6.3.4, is best demonstrated by residual analysis of reconstructed fine tracks. *Bad* clusters are likely to have skewed reconstructed position which will result in a skewed fine track reconstruction. *Good* clusters are likely to have a good hit-position reconstruction and will better describe the muon track. Two cases were considered – tracks reconstructed from any combination of good or bad clusters and those reconstructed from groups of three to four good clusters only. Results of this study are shown in Figure 40, where the shape of the residual is a result of the precise good cluster reconstruction and noise from bad cluster reconstruction. The prominent Gaussian represents the good track reconstruction and the wider Gaussian represents contamination from bad clusters. Figure 40(a) shows a width of $45\mu\text{m}$ for the signal and $162\mu\text{m}$ for the noisy tracks, while Figure 40(b) shows a smaller $32\mu\text{m}$ signal width and a $107\mu\text{m}$ width from residual misclassified bad cluster reconstruction. The expected position resolution for the CSCs is around $70\mu\text{m}$.

From these results it is clear that misclassification of clusters plays a great role in the reconstruction algorithm.

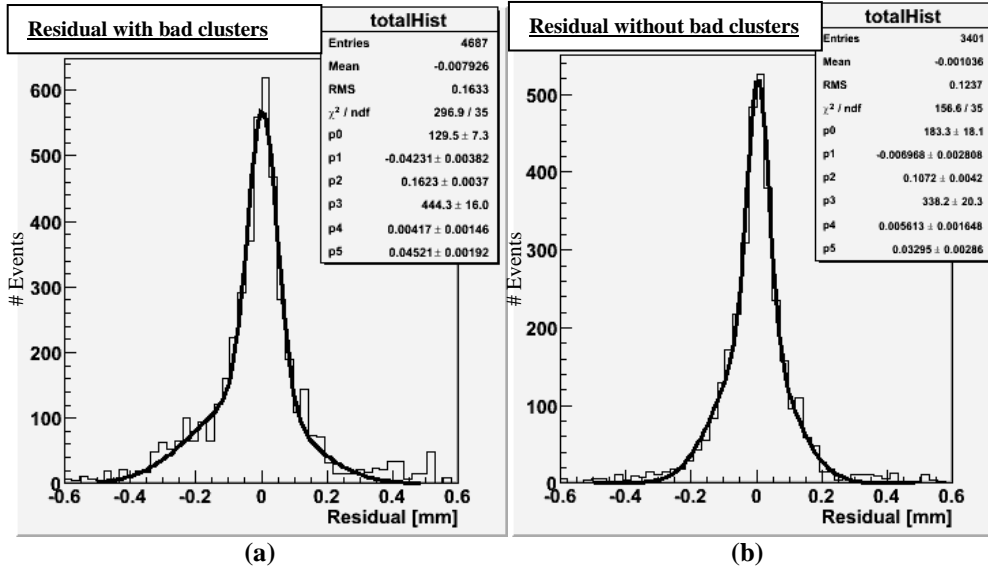


Figure 40: Total residuals of the reconstructed CSC tracks.
 (a) Bad clusters included in reconstruction.
 (b) Bad clusters excluded from reconstruction.

The results from the CSC reconstruction algorithm have not yet been integrated with the global reconstruction frameworks. Integration trials with MuGirl resulted in good agreement of the CSC muon segments with the extrapolated muon segments from the Inner Detector, shown in Figure 41.

An analysis was done on 100GeV single-muon simulated events, comparing only those events with $|\eta|$ between 2.0 and 2.7. Track segments originating from either the CSC or the ID were extrapolated to the middle MDT station. Their extrapolated hit positions were compared with the measured hit positions from the MDT hits and a $\Delta\eta$ was calculated. The measured efficiency out of 272 muon tracks in the ID was 75% for the CSC_DHoughSemgnetMakerAlg and 72% for the Csc4dSegmentMaker.

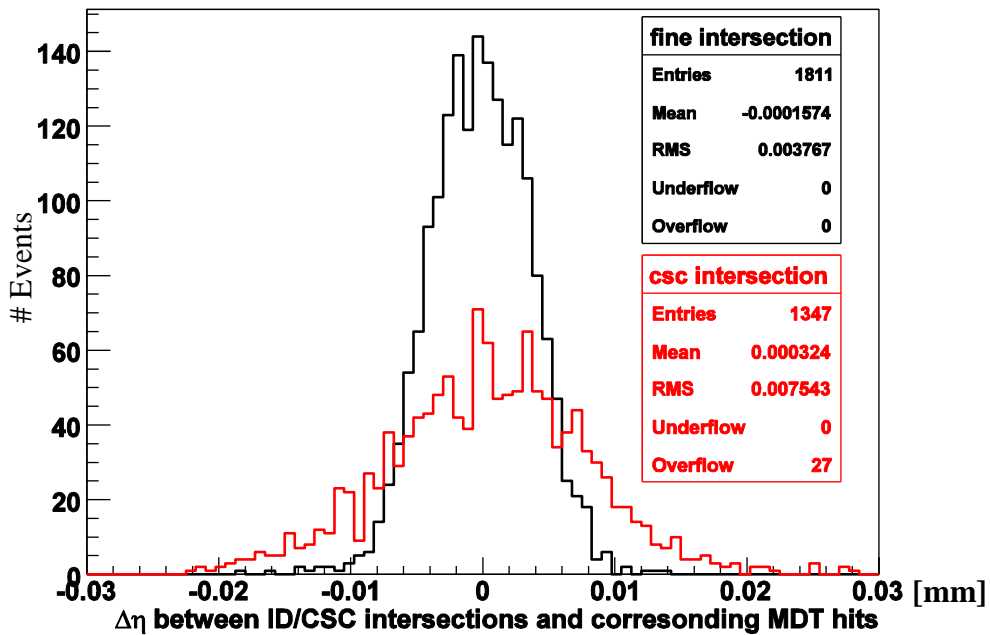
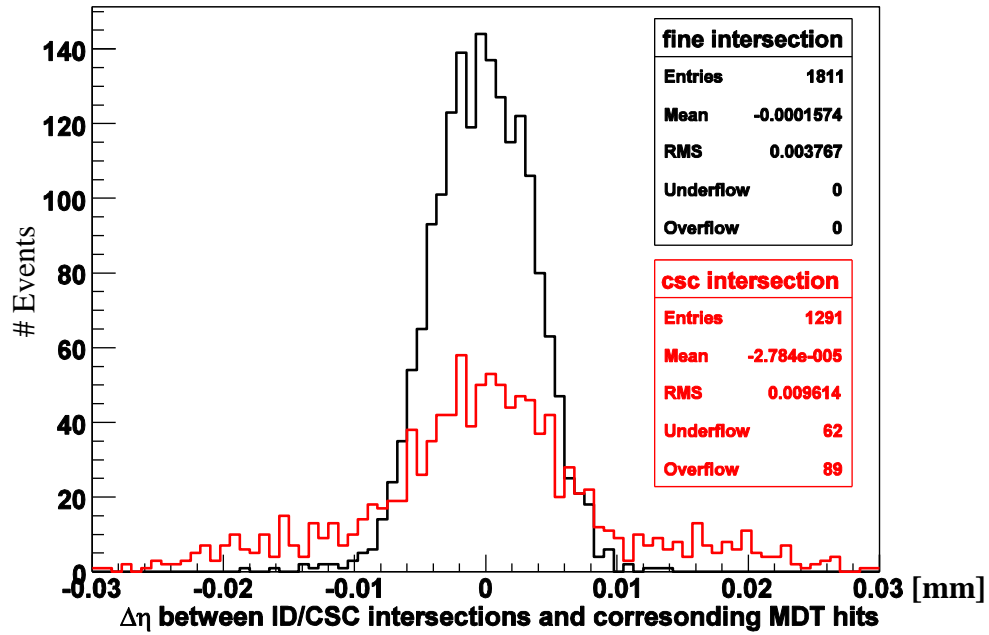


Figure 41: Extrapolated CSC and ID tracks to the MDT middle station. The $\Delta\eta$ calculated is between the extrapolated tracks and the real MDT hits found in the station. Top – Inner detector tracks checked against CSC tracks from the CSC_DHoughSegmentMakerAlg
Bottom - Inner detector tracks checked against CSC tracks from the Csc4dSegmentMaker.

MuGirl trials have compared two competing algorithms – the CSC_DHoughSegmentMakerAlg and the Csc4dSegmentMaker. The results (in red) show a great deal of agreement with each other and the difference of efficiency and variance between the two demonstrates the strength of the Hough Transform analysis.

7 Summary

This thesis is a culmination of work encompassing various fields in the ATLAS Muon Spectrometer. It documents in great detail the pattern recognition algorithms used to allow efficient track reconstruction of muons in the ATLAS as whole and the Cathode Strip Chambers specifically. The algorithms are primarily based on the detect-before-estimate approach, chosen to produce low fake rates of tracks and be less computationally intensive than the combinatorial approach. This enables the algorithm to do a quick rough scan of the parameter space, select the probable track candidates, and only then committing considerable computing resources analyzing and fitting these candidates.

The Hough algorithm was coded into the ATHENA software environment in various modular packages, enabling users to add or drop parts of the tracking chain as they see fit. Owing to the nature of ATHENA, the Hough packages are regularly updated, improving its functionality as more data becomes available. This is an ongoing process and requires considerable amount of work.

As data validation software becomes available for the ATLAS simulated data, thorough analysis and validation runs will be issued, comparing this algorithm's results the simulated data and the results of other algorithms. It is a challenging task which will prove essential to the further development of the CSC Hough tracking algorithm.

Additionally, description of the work done for the purpose of TGC quality testing is found in the next section. It includes further pattern-recognition algorithm devised for hole-tracking in the TGCs as well as a miniature standalone hardware and software system for preliminary tests of channel signals in the TGCs.

8 Appendices

The following appendices shed light on some of my contribution in the course of writing this thesis. The first deals with the CSC tracking software implementation. The second has nothing to do with the CSC and is related to a work I did for the TGC project.

8.1 CSCs

The following appendix describes in some more details the structure of the software used for CSCs tracking, which was developed as part of this thesis work.

8.1.1 Software Flow

The general software flow is given by Figure 42-Figure 46 below:

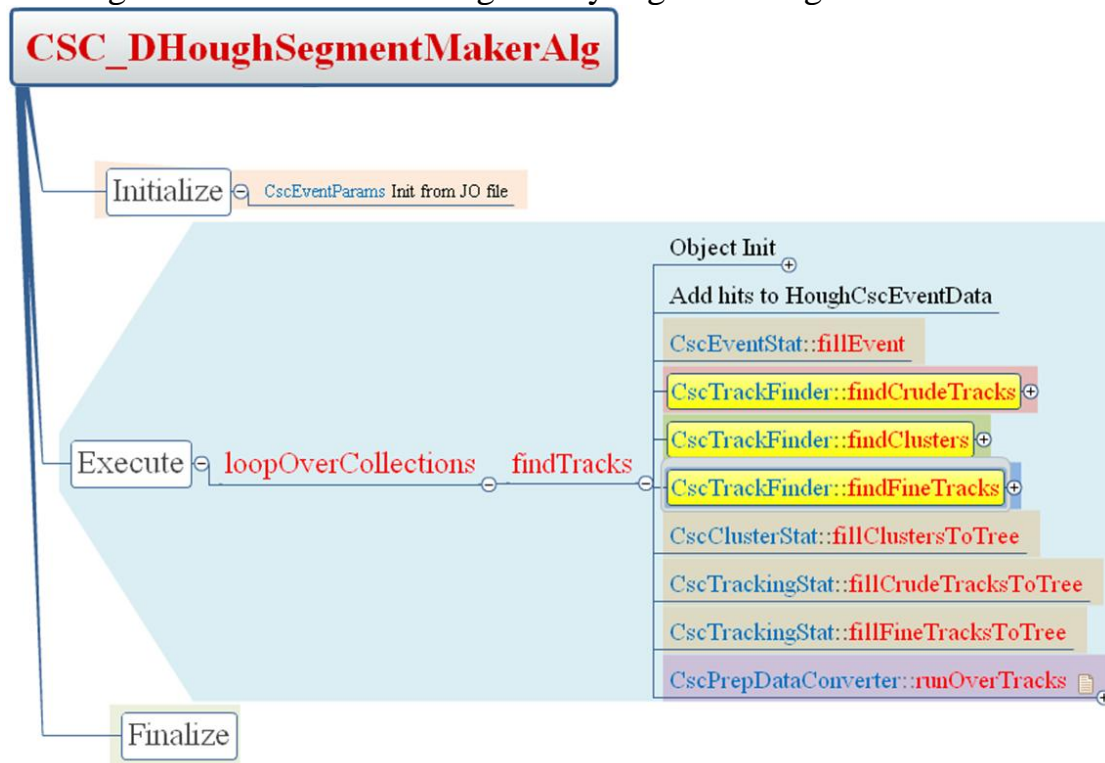


Figure 42: Global CSC Reconstruction software flow.

The three main algorithmic phases are shown, Initialize, Execute and Finalize. The yellow squares represent the three main stages of the algorithm – Crude Track Finding, Clusterization and Fine Track Fitting. Red text represents the function names; blue text represents the classes to which these functions belong to.

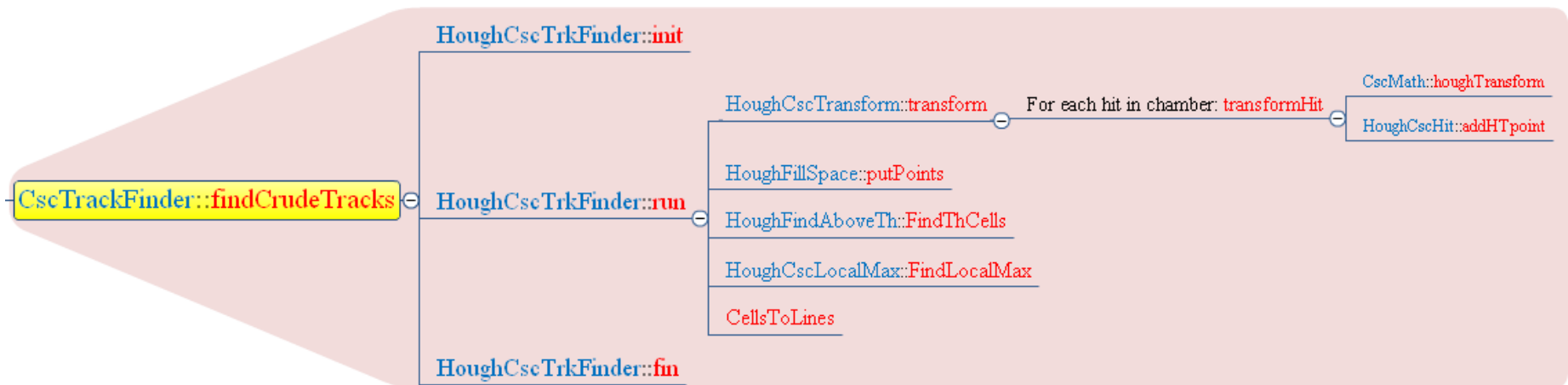


Figure 43: Workflow of the Crude Track Finding stage.
 Red text represents the function names; blue text represents the classes to which these functions belong to.

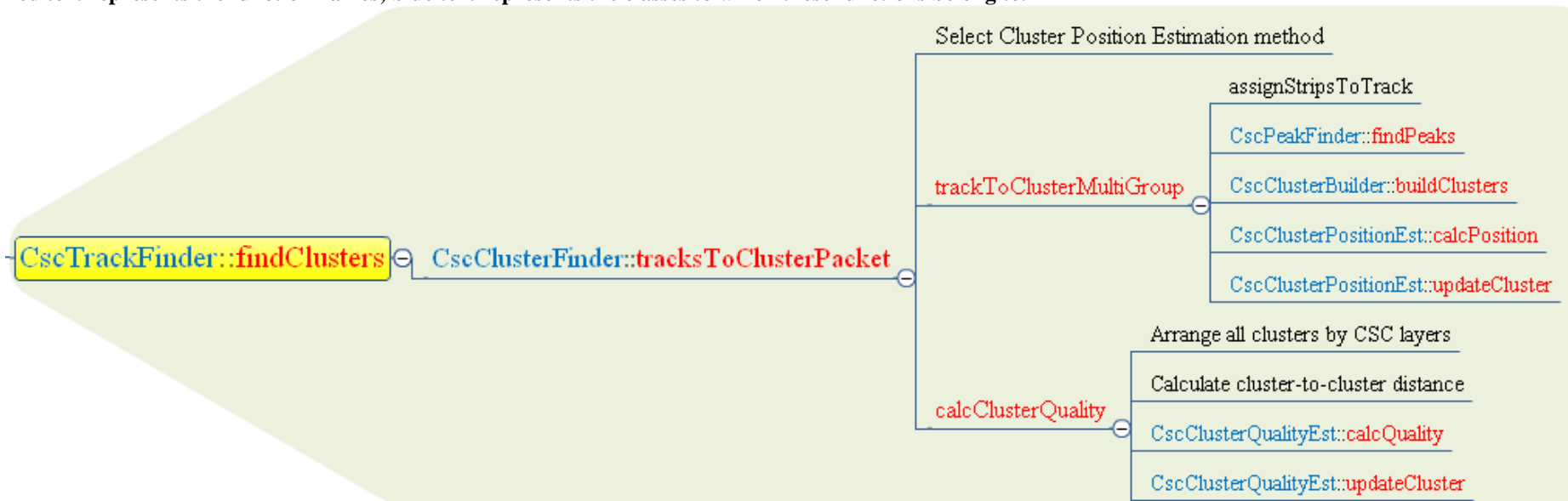


Figure 44: Workflow of the Clusterization stage.
 Red text represents the function names; blue text represents the classes to which these functions belong to.

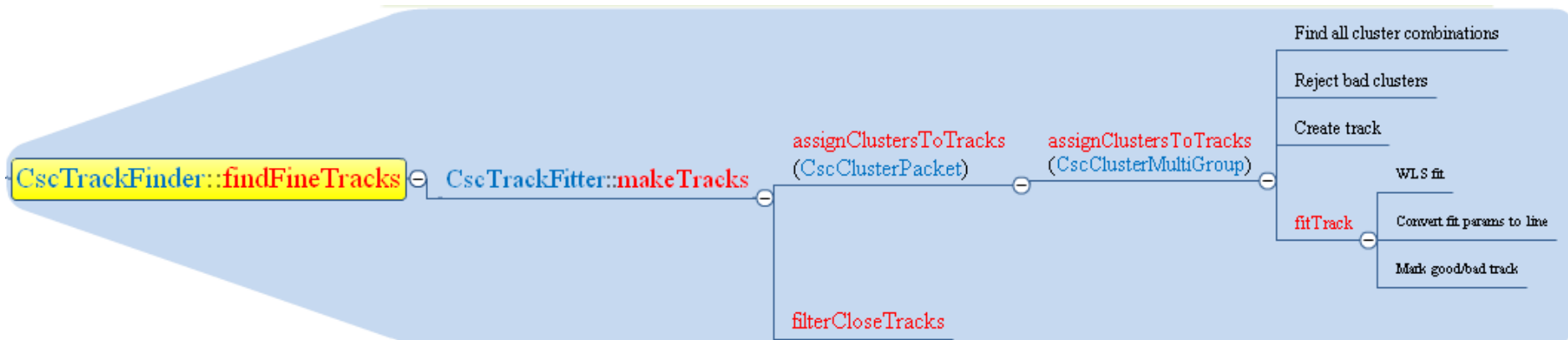


Figure 45: Workflow of the Fine Track Fitting stage.

Red text represents the function names; blue text represents the classes to which these functions belong to.



Figure 46: Workflow of the Storage stage.

Red text represents the function names; blue text represents the classes to which these functions belong to.

8.1.2 Software Usage

The CSC track finding method uses two main ATHENA packages: an ATHENA algorithm, *CSC_DHoughSegmentMakerAlg*, and an AlgTool, *CSC_DHoughSegmentMakerTool*. The first finds all possible segments in an event and the second filters out all events that are not in a given road.

8.1.2.1 CSC_DHoughSegmentMakerAlg

8.1.2.1.1 Algorithm input and output

The algorithm inherits from Algorithm class and implements the three main methods. The input is a container of vectors of `CscStripPrepDataCollection` objects, each representing a CSC Chamber and is a collection of `CscStripPrepData` objects which represents one CSC strip.

The output is:

- a. `SegmentCollection` – a collection of `MuonSegments` that represent the muon tracks traversing through the CSC chambers.
- b. `SegmentCombinationCollection` – a collection of `MuonSegments` organized in a segments-per-station hierarchy to better resolve track ambiguities.
- c. `CscStripPrepDataContainer` – A collection of `CscStripPrepData` containing all the clusters belonging to the discovered segments.
- d. `Events.root` file – an ntuple of the event data found. [Optional]
- e. `Clusters.root` file – an ntuple of the cluster data found. [Optional]
- f. `Tracks.root` file – an ntuple of the crude tracks and fine tracks found. [Optional]

8.1.2.1.2 Using the tool

In order to use the *CSC_DHoughSegmentMakerAlg* tool one should:

- a. Include the algorithm's `jobOptions` file in the main `jobOptions`.
- b. Retrieve the found segments with the key "`CscHoughSegments`" from `StoreGate`.
OR
Retrieve the found segments in `SegmentCombination` with `StoreGate` key "`CscHoughSegmentsCC`".
- c. Retrieve the found clusters with the key "`CscHoughClusters`" from `StoreGate`.

8.1.2.1.3 Using the tool job option

The algorithm has many options to run. Using the Job Option one can adjust the algorithm for his needs. The options are divided into a few categories:

(Default values in square parentheses)

a. General:

OutputLevel – The algorithm’s output level. [INFO]

DumpAllInput – Dumps all the CscPrepData information of all existing hits. [TRUE]

DumpSelectedInput – Dumps selected CscPrepData information of all existing hits. [FALSE]

b. Naming:

InputClusterCollectionName – The StoreGate key for the input data. Contrary to the name, this collection does not contain clusters, simply a long list of strips per chamber. [CSC_PREPDATA_NEW]

OutputClusterCollectionName – The StoreGate key for the output clusters. [CscHoughClusters]

SegmentCollectionName – The StoreGate key for the output SegmentCollection. [CscHoughSegments]

SegmentCombinationCollectionName – The StoreGate key for the output SegmentCombinationCollection. [CscHoughSegmentsCC]

c. Printing:

PlotCscClusterContainer – Plots the CscClusterContainer that was found by the algorithm and recorded into StoreGate. [FALSE]

PlotCscSegmentCollection – Plots the CscSegmentCollection that was found by the algorithm and recorded into StoreGate. [FALSE]

PlotCscSegmentCombCollection – Plots the CscSegmentCombinationCollection that was found by the algorithm and recorded into StoreGate. [FALSE]

d. ClustEstParams: Methods of Cluster position and quality

PositionEstMethod – Method for cluster position estimation. [“Ratio”]

Possible methods are:

- “Simple” – Cluster hit position is the cluster peak.
- “CoM” – Center of mass fit.
- “Parabola” – Parabola fit for Mathieson distribution.
- “Ratio” – Ratio fit for the Mathieson distribution.
- “ML” – Maximum Likelihood fit to the Mathieson distribution.

QualityEstMethod – Method for cluster quality estimation. The possible methods are:

- “Simple” – All clusters with number of peaks smaller than `maxPeaks`, with no neighboring clusters closer than `minClustDist` and with number of strips higher than `goodClustMaxStrips` are good. All the rest are bad.
 - “ML” – The ML method calculates the quality according to distance of the measured cluster charge distribution from the expected Mathieson distribution.
Not implemented yet.
- e. ClustMaxParams: Options for cluster rejection
- MaxBadStrips* – Maximum allowed bad strips in a cluster. Bad strip = mask or noise strip. [2]
 - MaxContinuousEmptyStrips* – Maximum continuous empty strips allowed inside a cluster [0]
 - MaxPeakDist* – In case a cluster is allowed to have two peaks, this is the max distance between those two cluster peaks in units of strips (if distance >maxDist two clusters are created). [0]
 - MaxStrips* – Max strips in a cluster. [8]
 - MaxWindowSize* – Max window size, in units of strips, around a crudeTrack for the search of clusters [20]
 - MinGoodStrips* – Minimum good strips required to create a cluster. [2]
 - ClustStripThreshold* – Threshold of strip charge allowed into a cluster. [10000]
- f. ClustQualParams: Parameters according to which the cluster quality is calculated
- MaxPeaks* – Max number of peaks allowed in a cluster. [1]
 - GoodClustMaxStrips* – Max Strips for a "good" cluster. Anything above that will be considered a bad cluster. [10]
 - MinClustDist* – Minimum cluster-to-cluster distance for a "good" cluster, in units of strips, calculated from the edges of neighboring clusters. Anything below that will result in bad clusters. [3]
- g. PeakParams: Options for finding peaks inside a group of hits.
- TreatMaskAsNoise* – Treat hits marked as masked muon hits as noise hits (cannot be peaks). [TRUE]
- h. StatParams: Options for writing a ROOT statistics files.
- TakeClustStat* – Write cluster statistics. [TRUE]
 - TakeEventStat* – Write complete event statistics (all data from hits). [TRUE]
 - TakeTrackStat* – Write track statistics [TRUE]
 - MaxTracksToStat* – Do not write track statistics for events with more tracks than this. [100]
- i. HoughParams: Parameters for the Hough Transform.
- MaxAngle* – Maximum angle allowed for a crude track (Hough

- track). All tracks with angles between $\text{PI}/2$ to MaxAngle will be allowed. [PI/3]
- Resolution* – Resolution of the Hough transform. To prevent digitization errors, the Hough Transform function gives back a number of values equal to the resolution. The first of these values that is inside the range of the Hough cell is used. [10]
- HoughChargeThreshold* – Charge threshold for the strips that get into the Hough transform. Strips under it will not be included in the transform. [10000]
- HoughValueThreshold* – Threshold value to create a crude track from the Hough transform. Each Hough cell is incremented by a value according to the hits that contribute to it. Muon hits have a value of 1, masked muons 0.4 and noise 0.01. Hough cell values above this threshold will create a crude track. [3.5]
- HoughMaxRegion* – Region around a local Hough maximum around which no other maximum can be found; meant to reduce multiple crude tracks for the same muon track. [3]
- j. TrackTakingParams: General parameters for fine track rejection.
- AngleConstraint* – Fine track angle restriction. Symmetric around trajectory orthogonal to the CSC plane. [90]
- ExcludeClustByLayer* – Exclude one CSC layer from the track fitting process. For debugging purposes. [FALSE]
- ExcludeClustLayer* – Excluded CSC layer. 1= first layer, 4= last layer. [4]
- FilterCloseTracks* – Filter close tracks and picks the best track out of their group. Best tracks are tracks minimal $\chi^2/\text{Degrees of Freedom}$. [TRUE]
- MaxChi_2_OverDeg* – Maximal accepted fine track $\chi^2/\text{degrees of freedom}$. Above that, fine tracks are tagged as bad fine tracks. [5]
- MinPointsInFineTrack* – Minimum number of points to create a fine track. [3]
- OnlyConvertToRoot* – Only read the CSC PRDs and convert them to ROOT file, without performing track reconstruction. [FALSE]
- MinTotClusterQuality* – Minimum summed quality required to create a good track. [2.8]
- MinClusterQuality* – Minimum quality required to create a cluster. All clusters below this are rejected. [0.3]

8.1.2.2 CSC_DHoughSegmentMakerTool

8.1.2.2.1 Algorithm input and output

The tool inherits from `IMuonSegmentMaker` class and implements a single interface.

The input is:

- a. `SegmentCombinationCollection` – a collection of `MuonSegments` organized in a segments-per-station hierarchy to better resolve track ambiguities.
- b. `TrackRoad` – a road that can be used by the algorithm to limit returned `MuonSegment` tracks.

The output is a `SegmentCollection` with only the segments lying inside the given `TrackRoad`.

8.1.2.2.2 Using the tool

In order to use the tool one should:

- a. Make sure the `CSC_DHoughSegmentMakerAlg` algorithm runs before this tool.
- b. Include the tool's `jobOptions` in the main `jobOptions`.
- c. Load the tool in the initialization of the algorithm.
- d. Call the `Find` function of the tool with the following interface:

```
std::vector<const Muon::MuonSegment*>* find( const Trk::TrackRoad&
road, const std::vector< std::vector< const
Muon::MdtDriftCircleOnTrack* > >& mdt, const std::vector<
std::vector< const Muon::MuonClusterOnTrack* > >& clusters, bool
hasPhiMeasurements);
```

Where the only relevant parameter is the `TrackRoad` (all the rest are not used and can be left blank).

An example of the use of the tool can be found in the `MuGirl` package (<http://atlas-sw.cern.ch/cgi-bin/viewcvs-atlas.cgi/offline/Reconstruction/MuonIdentification/MuGirl/src/CSC.cxx>)

8.1.2.2.3 Using the tool job option

The tool has many options to run. Using the `Job Option` one can adjust the tool for his needs:

(Default values in square parentheses)

- a. `OutputLevel` – The algorithm's output level. [INFO]
- b. `SegmentCollectionName` – The `SegmentCollection` key in `StoreGate`. This collection is not read and is only left behind for legacy purposes. ["CscHoughSegments"]
- c. `SegmentCombinationCollectionName` – The input `SegmentCollection` key in `StoreGate`. ["CscHoughSegmentsCombCol"]

- d. *isExR/ad* – Use internal (from jobOption) or external (from the interface) road parameters. If the road is given as an input from the algorithm that calls the tool, isExRoad should be True. [FALSE]
- e. *Road_dEta* – Internal Road’s dEta opening. [0.02]
- f. *Road_dPhi* – Internal Road’s dPhi opening. [0.01]
- g. *Road_dEtaMult* – Multiplier for the Eta opening. [1]
- h. *Road_dPhiMult* – Multiplier for the Phi opening. [1]

8.1.2.3 CSC_DHoughRoot

8.1.2.3.1 Input and output

The input is:

- a. A ROOT file, Events.root, containing the event data of each event.
- b. Analysis parameters hardcoded in `CscRootData::initEventParams`.

The output is:

- a. A visualization of the CSC layers, hough tracks, clusters, estimated hit points and fine tracks found in each event.
- b. Clusters.root file – an ntuple of the cluster data found. [Optional]
- c. Tracks.root file – an ntuple of the crude tracks and fine tracks found. [Optional]

8.1.2.3.2 Usage

When compiled, the package creates a run.exe executable. Add a link to the event data ROOT file called Events.root in the executable’s directory, then run the executable.

8.1.2.3.3 Using the parameters

To change analysis parameters, change the `eventParams` object in `CscRootData::initEventParams`. The parameters are similar to the ones detailed in section 4.2.3. Parameters specific to this package are:

- a. ColorParams: [`eventParams->colorParams`]
 - fineTrack* – The colour of the fine track. [Blue]
 - badFineTrack* – The colour of the bad fine track. [Cyan]
 - crudeTrack* – The colour of the crude track. [Green]
 - defaultTrackLine* – The colour of a default track (not marked as crude or fine). [Light Cyan]
 - realHist* – The colour of the real hits histogram. [Red]
 - maskHist* – The colour of the masked hits histogram. [Blue]
 - noiseHist* – The colour of the noise hits histogram. [Green]

clusterHist – The colour of the frame around a cluster. [Black]

clusterPeaks – The colour of the cluster peaks marker. [Black]

clusterHitPoints – The colour of the cluster hit point marker.

[Black]

markedStrips – Not used.

b. DrawParams: [eventParams->drawParams]

drawAllEvents – If true, draws all the events. [True]

drawSelectEvents – If true, and unless drawAllEvents is true, draws only selected events. [False]

printEventsToFile – If true, creates a .ps file containing the EventViewer plot of an event. [False]

drawFineTracks – If true, draws the fine tracks in the EventViewer. [True]

drawBadFineTracks – If true, draws the bad fine tracks in the EventViewer. [False]

drawCrudeTracks – If true, draws the crude tracks in the EventViewer. [False]

drawClusters – If true, draws the clusters in the EventViewer. [True]

drawClustPeaks – Not used.

drawHoughTransform – If true, creates HoughSpace.root and HoughPoints.root with containing a detailed plot of the event's Hough transform.

8.1.2.4 CSC_DHoughTestBeam

8.1.2.4.1 Input and output

The input is:

- a. The testbeam events file.
- b. The testbeam pedestal file.
- c. x5files.txt – a file listing the location of testbeam event file, pedestal file and number of initial events to skip (in some runs the first events are for calibration).

The output is an Events.root file – an ntuple of the event data found.

8.1.2.4.2 Usage

When compiled, the package creates a run.exe executable. Create an x5files.txt file with a list of input files. Each line should have the path and name of the testbeam event file, the path and name of the pedestal file

pedestal file and the number of initial events to skip for each file.
Syntax: run.exe number_of_events.

8.1.3 Class Descriptions

8.1.3.1 MuonDHough

HoughSpace

This class contains the main data structure of the Hough transform. The space consists of an array of *HCell* objects; each represents a cell in the Hough space. The cell contains the total value of the cell, vectors of values for each layer (total and values for separate multilayers of MDTs), charge vector for the CSC and index to HitCont vector, that holds the pointers to the hits. This class has some useful functions for initializing the space, copy space to space, print the space content, and get and set functions.

HoughFillSpace

This class gets the Hough hits and fill the Hough space. It has several functions that take care that the cell will be updated only once per hit. It also has a function that fills only one value per layer in a Hough cell (it takes the best value).

HoughHit

This class is the data structure of a hit. It contains a vector of points in the parameter space, the layer and multilayer of the hit.

HoughCircleTransform

This class is responsible to the transform of a circle to the Hough parameter space. It includes all the mathematics needed for several scenarios: transform of a circle, transform of a ring, transform of inner and outer rings (used for mask hits).

HoughDisplay

This class is used to create root files of the Hough space. It should only be used for debugging. There are two main functions: *drawHoughHits* which creates an equivalent Hough space and store all the cell values in ROOT files. Note that it is a basic space which does not handle the case of several hits from the same layer that contributes to the same cell. The *drawHoughSpace* stores all the hits from the original space that are above the threshold in the ROOT files. Thus, not all the cell values are presented.

HoughFindAboveTh

This class goes over all cells and finds if the cell value is bigger than the threshold. The threshold is given as another object (inherited from *HoughCellTH*) which can be different from detector to detector.

HoughDefaultCellTh

This class is an example (default class), inherited from *HoughCellTH* that has a threshold and a logical conditions, for comparing to cell values.

HoughFindLocalMax

This class goes over all cells above threshold and try to find the local maximum. The search is done in a local area whose dimensions are given as an input. It gets a pointer to *HoughLocalMax* function that can be changed according to the application.

HoughDefaultlocalMax

This class is an example (default class), inherited from *HoughLocalMax* that implement an example of finding a local maximum for a given cell.

HoughCombineLines

This class goes over all cells that are local maxima, sort them, makes a loop over all cells and looks for close cells. The comparison of the cells is done by calling the *HoughCombine* object.

HoughDefaultCombineLine

This class is an example (default class); inherited from *HoughCombine* that makes the comparison between the cells. It calculates the distance between the cells, and if it is smaller than a certain threshold and takes the cell with the best value.

8.1.3.2 CSC_DHough

ClusterConvertor

Converts from *CscCluster* object to *CSC_DHoughClusterML* internal objects and vice-versa.

CscCluster

Derived of *CscHitDataGroup*. A group of close strips banded together with additional cluster parameters such as: number of good, masked and bad strips, cluster window size and number of strips in the cluster. Additionally, it has a vector of peaks (may be one or several) and vectors of *CscLocalHitPoints* and *CscIndexHitPoints*.

- *CscClusterGroup* is a vector of clusters in one layer belonging to one track.

- `CscClusterMultiGroup` is a vector of `CscClusterGroups` in various layers assigned to the whole track.
- `CscClusterPacket` is a vector of `CscClusterMultiGroup` of all the various tracks belonging to the same chamber.
 - a. `CscIndexHitPoint`
After taking into account the shape of the cluster, an extrapolated hit point around the peak is calculated, which represents the reconstructed muon hit point. This vector is the fractional index of the reconstructed hit point.
 - b. `CscLocalHitPoint`
After taking into account the shape of the cluster, an extrapolated hit point around the peak is calculated, which represents the reconstructed muon hit point. This vector is the local position of the reconstructed hit point, in local chamber coordinates.

CscClusterBuilder

This class is responsible for building `CscClusters`. It takes as input a list of all chamber hits, a list of all chamber peaks and the max parameters for a single cluster and builds clusters around these peaks. It then goes over all hits in the vicinity of the peak and adds them to the cluster, making sure the set cluster-limitations aren't broken. In case the limitations are broken, it tries to form two clusters out of the hits. The output is a `CscClusterPacket` which is a list of clusters in one chamber.

CscClusterFinder

This class is responsible for finding all clusters surrounding a crude track. It takes as input a vector of `CscCrudeTracks`, `HoughCscEventData` and the event parameters. It first assigns a list of hits in a window around a crude track to the crude track and then uses the `CscPeakFinder` to find all peaks in those hits. It continues by using the `CscClusterBuilder` to make clusters out of these peaks and uses the chosen cluster-positioning-estimation and cluster-quality-estimation techniques to find the cluster position and quality. The output is a `CscClusterPacket` which is a list of clusters in one chamber.

CscClusterPositionEst

This is an abstract class whose derived classes are responsible for calculating a precise cluster hit position. Each derived class must implement the `calcPosition` method and output results into a `CscLocalHitPoints` vector. Then the `updateCluster` method is

called to update the cluster parameters with these results.

Derived classes are:

- a. `CscClusterPositionEstSimple`
A simple position takes the position of the peak strip as the position of the entire cluster.
- b. `CscClusterPositionEstML`
Calculating cluster position according to the Maximum Likelihood algorithm, detailed in section 6.3.4.
- c. `CscClusterPositionEstParabole`
Calculating cluster position according to the parabola algorithm, according to which 2 points on either side of the peak are substituted in the formulas (8), (10)-(11).
- d. `CscClusterPositionEstRatio`
Calculating cluster position according to the ratio algorithm, according to which 2 values on either side of the peak are substituted in the formulas (6)-(8).
- e. `CscClusterPositionEstCoM`
Calculates the cluster position by doing a center-of-mass of the peak and its two adjacent strips using formula (9).

CscClusterQualityEst

This is an abstract class which its derived classes are responsible for calculating a cluster quality. Each derived class must implement the `calcQuality` method and returns a double. Then the `updateCluster` method is called to update the cluster parameters with this result.

Derived classes are:

- a. `CscClusterQualityEstSimple`
Calculates a simple quality according to these conditions:
 - Cluster peaks \leq maximum number of allowed peaks. (*MaxPeaks*)
 - Closest cluster-distance $>$ Minimum allowed cluster-distance. (*MinClustDist*)
 - Number of strips in cluster \leq maximum allowed good cluster strips (*GoodClustMaxStrips*)

If all these are true, the cluster quality will be 1. Otherwise, it will be 0.4. Later on a more sophisticated method may be applied.

In track fitting stage, tracks will be created only for clusters with cumulative quality that is higher than *MinTotClusterQuality*.

This will essentially be able to filter out various combinations of bad and good clusters. The default value for this parameter accepts combinations of 2 bad clusters and 2 good clusters

(quality=2.8), 3 good clusters and no fourth cluster (quality=3), but rejects a combination of 1 good cluster and 3 bad clusters (quality=2.2).

- b. `CscClusterQualityEstML`
Calculates quality according to the Maximum Likelihood algorithm, according to details in section 6.3.4. Yet to be implemented.

CscCrudeTrack

This class is the result of the first stage of reconstruction – the Hough transform. The crude track contains the track's initial hit position in the first CSC-layer plane and the tracks direction, both in local CSC coordinates. It also contains the list of strips that created the crude track, sorted by the different CSC layers.

This object is created by `HoughCscTrkFinder` (in ATHENA) or `CscX5DataPrepper` (in X5 testbeam analysis). It is then used by `CscClusterFinder` to find all clusters near the crude track.

CscFineTrack

This class is the result of the second stage of reconstruction – the fine tracking. The fine track is a fine-tuning of the crude track using a weighted-least-squares fit of the precise cluster positions. It contains the list of the clusters associated with it, the number of strips in all these clusters. In case clusters have more than one hit points, the used hit-point is stored in `m_hitPointIndices`. Similarly to the crude track, the fine track also contains the track's initial position and direction, as well as the fit parameters used to create it and a parameter to indicate whether it is a good track. Good tracks are ones in which the total cluster quality is greater than `MinTotClusterQuality`, as discussed in `CscClusterQualityEst`.

CscStatistics

This group of classes is used for statistics purposes. It is a dump of each of the main data objects to root ntuple. The group consists of:

- a. `CscEventStat`
Dumps the event data into a root file. Event data is the full strip information and the geometry parameters.
- b. `CscClusterStat`
Dumps the cluster data into a root file.
- c. `CscTrackingStat`
Dumps both crude track and fine track into a root file. Contains a method to dump the residuals as well.

CscHitData

This group of classes is the CSC strips raw data taken from StoreGate and rearranged into local data objects. The group consists of:

- a. *CscHitDataStrip*
A single CSC strip, including the original *CscStripPrepData* position, charge and time of the hit. Each strip contains a list of charge samples – the sampled charge around the peak of the induced charge on the strip. The timing of this peak is used for activity detection. Following the activity-detection stage, strips are divided into three types – real, masked and noise, which is called the hit type. The strip type is either Y type (along the non-precision phi coordinate) or X type (along the precision eta coordinate).
- b. *CscHitDataGroup*
A group of *CscHitDataStrips* in the same CSC layer that share a common characteristic. The *CscCluster* is derived from this object.
- c. *CscHitDataPacket*
A group of *CscHitDataGroup*, one from each CSC layer, representing the entire CSC chamber.

CscHitGeometry

This group of classes represents the CSC geometry, which is a separate entity than *CscHitData* since sometimes a position of a strip that didn't fire (and therefore does not exist in *CscHitData*) is needed. The group consists of:

- a. *CscHitGeometryStrip*
Represents the position of one CSC strip according to *GeoModel*.
- b. *CscHitGeometryGroup*
A group of *CscHitGeometryStrip*. Represents the position of all the strip in one layer of a CSC chamber.
- c. *CscHitGeometryPacket*
A group of *CscHitGeometryGroup*. Represents the position of all strips in all layers of a CSC chamber.

HoughCscEventData

A collection of objects that represent all the data in a single event. It contains the *CscHitDataPacket* and *CscHitGeometryPacket* objects, as well as a *CscGeometry* object, which provides general non-strip-specific data of the CSC geometry.

It is the main data object passed along from the beginning of the algorithm to most of the subalgorithms.

CscMath

This class is a collection of all the purely mathematical calculations in the algorithm.

CscPeakFinder

This object finds peaks in a given group of strips. It takes as input either a `CscHitDataPacket` or `CscHitDataGroup` and returns either vector of `CscHitDataGroup` iterators pointing to the peak strips in the group or a vector of those vectors, representing the peaks in all the groups.

CscTrackFinder

This is a container class for the 3 tracking stages – crude track finding, cluster finding and fine track finding, represented by the three functions implemented in it – `findCrudeTracks`, `findClusters`, `findFineTracks`. It is meant to be used seamlessly by both the simulation tracking package and the test-beam tracking package.

CscTrackFitter

This class is responsible for fine track creation. Its `makeTracks` function takes a list of `CscClusterPacket` in a CSC chamber and a list of fitting parameters in `CscEventParams` and returns a list of the fitted tracks in `CscFineTracks`. The class uses weighted-least-squares fit to fit all possible cluster combinations to a track that adheres to the constraints given in `CscTrackTakingParams`. In case there are multiple close tracks, it can filter them out or keep them, depending on the value of `FilterCloseTracks`. The filter takes only the track with the lowest χ^2/DOF . The χ^2/DOF of every fitted track must be below `MaxChi_2_OverDeg` or the track is rejected.

HoughCscCellTh

This class is derived from the abstract `HoughCellTh` and implements the threshold function for the specific needs of the CSC. It uses the parameters from initialization phase and checks if a weighted sum of the multi values of the Hough cell is above a certain threshold.

HoughCscHit

This class is derived from `HoughHit` and is the data structure of a Hough CSC hit. It contains the Hough transform of the hit and a pointer to the `CscHitDataStrips` that contributed to it.

HoughCscLocalMax

This class is derived from `HoughLocalMax` and implements the Hough local-max function for the specific needs of the CSC. It uses the parameters from the initialization phase to check if the Hough cell is a local maximum.

HoughCscTransform

This class is responsible to convert the `CscHitData` into Hough points in the Hough parameter space. It first initializes all parameters, including the road constraints of the transform; then it loops over all hits, transforming each hit point into a Hough hit using a linear transform. Lastly, it creates a pointer to the `CscHitDataStrip` from which it was generated. These Hough hit points will later be put into `HoughSpace`.

HoughCscTrkFinder

This class is responsible for calling all the Hough-tracking classes and methods in the right order. The initialize phase creates an empty Hough space, into which Hough hits will be filled later. The run phase uses performs the Hough transform using `HoughCscTransform`, fills the results into `HoughSpace`, finds maximum regions in this space and finally transforms these into crude tracks.

CscParams

This class contains most of the common parameters used for the `HoughCSC_DHoughAlg` as entered by the `jobOptions` file parameters explained in section 4.2.3.

8.1.3.3 CSC_DHoughClusterML

This package performs cluster position and quality estimation based on the Maximum Likelihood algorithm. It takes as input a list of `CscDetClusters` and compares each of them to the theoretical Mathieson distribution expected of a cluster. The results are given both by the best estimated cluster position and an estimated cluster quality.

8.1.3.4 CSC_DHoughRoot

CscRootData

This class is a modified ROOT generated class that represents the event data-structure in the ROOT ntuple. It is essentially the ROOT analogy to the `HoughCscEventData` object. Its `Loop` function is the core of the algorithm, listing the methods to follow per ROOT entry.

CscRootDataConverter

This package converts ROOT event data to HoughCscEventData. It takes a CscRootData object and converts it to HoughCscEventData.

CscRootDataReader

This is a container class for the CSC ROOT analysis package, responsible for initializing all the data structures and running the converter.

8.1.3.5 CSC_DHoughDisplay

CscDisplay

This class is the EventViewer for the CSC hough code. It is called by CSC_DHoughRoot and draws the CSC event data, clusters and tracks on the fly. Its processData function is given data to process into a ROOT structure while the draw function draws this data into a ROOT canvas.

8.1.3.6 CSC_DHoughTestBeam

CscX5Data

This class is the X5 testbeam data object.

CscX5DataPrepper

This class is responsible for converting data from the CscX5Data object to the HoughCscEventData object which will later be written into ROOT.

CscX5DataReader

This is a container class for the CSC Testbeam package, responsible for initializing all the data structures opening the testbeam data files and reading each event, converting it to a HoughCscEventData event and writing it into a ROOT file.

8.1.3.7 CSC_DHoughSegmentMakerAlg

CSC_DHoughSegmentMakerAlg

This class is the core of the algorithm. It initializes all services and job-options variable, then for each event converts the input data (CscPrepData) to internal object structure, calls the sub-algorithms for finding the crude tracks, clusters and fine tracks, and saves results in external ROOT files (optional). It then calls the CscPrepDataConverter to convert the results back to EDM objects and writes them into StoreGate.

CscHoughNtuple

The class is the Ntuple structure of the algorithm. (Under construction)

CscPrdClusterStat

This class is the temporary ROOT ntuple for the `CscPrepData` clusters in the algorithm. It will eventually be replaced by `CscHoughNtuple`.

CscPrdSegmentStat

This class is the temporary ROOT ntuple for the `MuonSegments` in the algorithm. It will eventually be replaced by `CscHoughNtuple`.

CscPrepDataConverter

This class is responsible for converting the internal object structure back to EDM objects. It takes a list of fine tracks as input and converts them to a list of `MuonSegmentCombinationCollection` (SG key: *SegmentCombinationCollectionName*) in which there is only one `MuonSegmentCombination` comprised of a list of segments found in each of the CSC stations. For legacy purposes, it also saves a list of `Trk::SegmentCollection` (SG key: *SegmentCollectionName*) with all segments found in the entire event. Each segment contains a list of `CscClusterOnTrack` and each of those contains a `CscPrepData` cluster containing all the strips that create it. In addition to the segments, the `CscPrepData` clusters are saved separately into `StoreGate` (SG key: *OutputClusterCollectionName*).

8.1.3.8 CSC_DHoughSegmentMakerTool

HoughCSCTrkFinderTool

This is the core class of this tool, responsible for extracting a group of segments in a certain road from `StoreGate`. Only one of its find functions is implemented, taking a `TrackRoad` as input, reading the event's segments from `StoreGate` and giving back a vector of `MuonSegments` representing all the segments in which the segment extrapolations to the first CSC layer and the last CSC layer are both inside the given road.

8.2 Thin Gap Chambers

8.2.1 Testbench

The Tel Aviv University TGC testbench uses cosmic muons to measure the efficiency of the TGCs [28]. The tested TGCs are sandwiched between two precision chambers (PRC). The PRC measure the impact point of the cosmic muon that crosses them. From this information, the track of the cosmic muon is reconstructed, and the crossing point through each plane within the stack of the tested TGCs can be calculated. The number of times a signal was registered in a TGC, that was crossed by a muon, relative to the total number of crossing muons, defines the efficiency.

The information that a muon crossed the PRC, and therefore the tested stack, is provided by two scintillator planes, one above the upper PRC and one below the lower PRC. This is depicted in Figure 47.

A coincidence between a hit in the upper scintillator plane and lower scintillator plane serves to trigger the data taking.

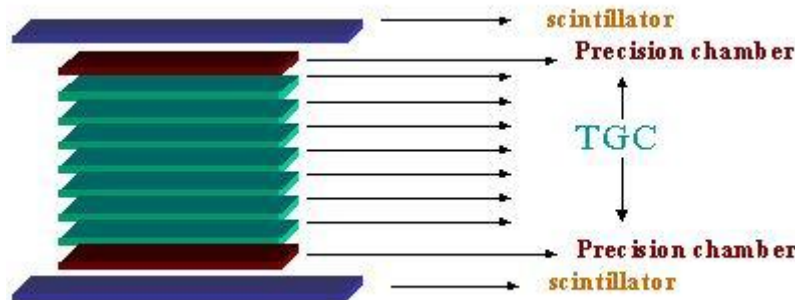


Figure 47: The testbench schematic structure.

A combination of online and offline readout software reads the data from the PRCs and TGCs to form an efficiency map of each TGC chamber (See Figure 12). This map should be examined for holes (i.e. inefficient areas) in the surface of the TGC and give a total percentage of the chamber's efficiency.

8.2.2 Hole-tracking software

The goals of the tesbenches are to see that the chambers are operating well under High voltage, and are efficiently detect the passing muons in most of the detection area. The test should provide an efficiency map of each detector. The regions with lower efficiency detected by the testbench software must be evaluated and logged in the offline database in order to allow better optimal positioning of the TGC chambers in the ATLAS pit.

A detailed mapping of the chambers efficiency at this stage will assist in better systematic understanding of the triggering system at the stages of data taking and analysis. TGC chambers with overlapping inefficient areas must not be placed in front of each other in the Muon Spectrometer wheels. Additionally, problematic strips and wires in a TGC must also be marked for repair, or marked as such for better understanding of the real detector structure. In the following we use the term hole referring to contiguous regions with efficiency lower than the 90%. The hole-tracking software we have developed is meant to find these holes, their size and position with respect to the center of a chamber as well as missing strips and wires. These will be registered in the offline database for further use.

As a criterion for the quality of a chamber it was decided to set a limit on the total area of inefficient regions. An inefficient area is defined as a contiguous area of more than 25 cm^2 that extends by more than 5 cm in each direction (x & y), with every point in that area having an efficiency of less than 95%.

The quality criterion is based on the percentage of the integrated size of the surface of all the inefficient areas in the detector. The integrated inefficient area should not exceed **5%** of the total active area of the detector. The calculation of the inefficient area is done including all the known support lines and buttons.

The algorithm that calculates the integrated inefficient area is the following:

1. Search for all the inefficient regions. By searching for $5 \times 5 \text{ cm}^2$ elements, the search is not sensitive to the support lines and the support buttons.
2. Calculate the total area covered by these inefficient regions.

The technique is illustrated in Figure 48, showing a TGC chamber after the search for inefficient regions is applied. The inactive areas around the supports disappear, while the inefficient regions in the left low corner are clearly seen. Small inefficient regions and edge effects are naturally ignored.

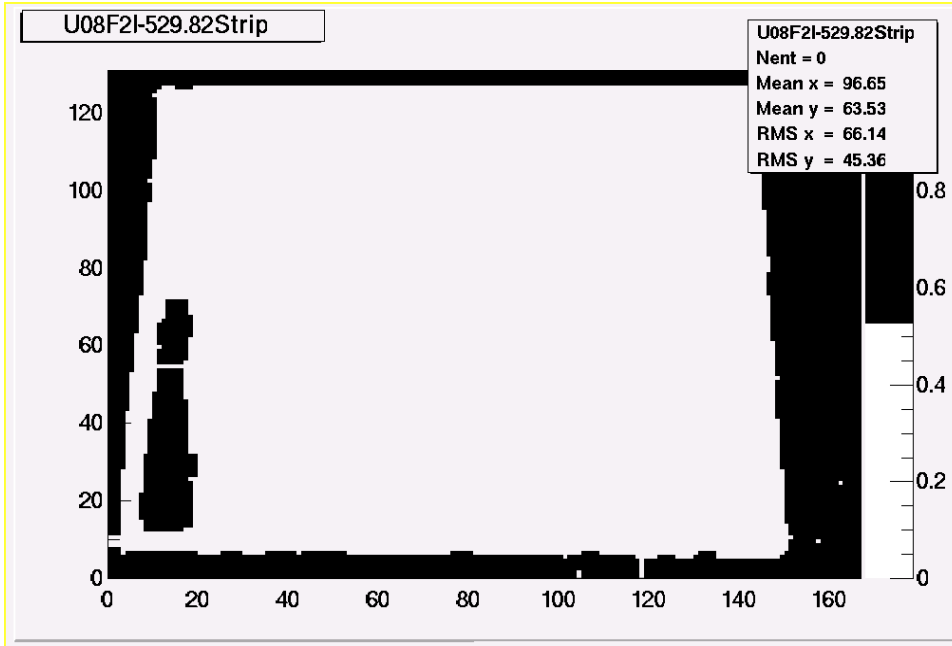


Figure 48: An efficiency mask of a TGC chamber.
White areas represent efficient areas while black areas represent inefficient areas.

Hole tracking was performed on this mask by applying several filters to this histogram. Each point of the histogram was sampled then recursively compared to all the points in its vicinity. This method essentially classifies bounded areas in the histogram and color-coded them appropriately. Each color value represented a different bounded area, as seen in Figure 49.

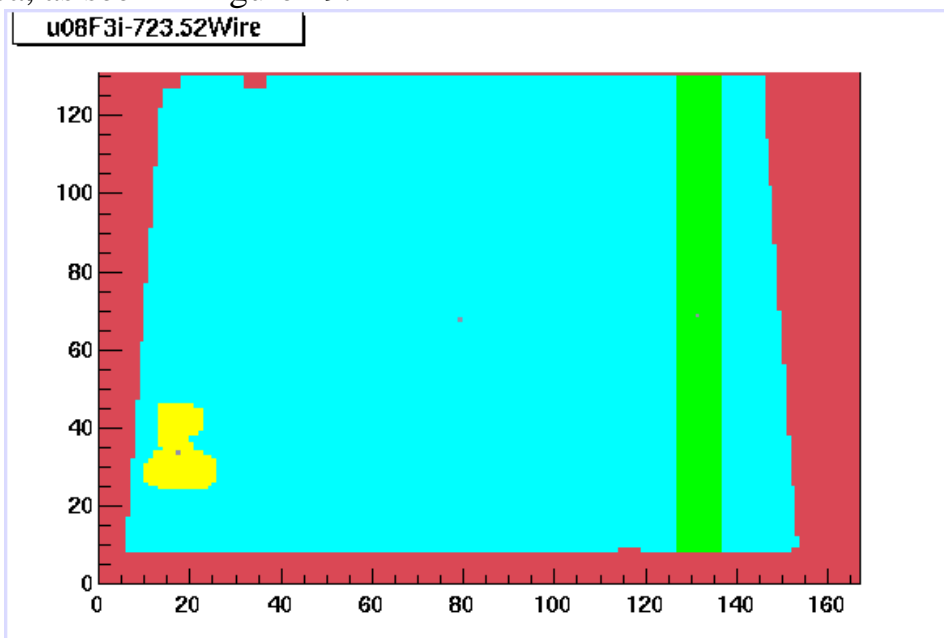


Figure 49: A hole classification plot of a TGC chamber.
Each bounded area is given a different value and its center (marked by a grey dot) is calculated using Center of Mass algorithm. The yellow amorphous area represents an inefficient hole and the green rectangular area represents an inefficient group of strips.

The hole-tracking procedure scanned all the valid bounded areas (all those that are bounded by other areas and are not the *background*.

Background is the only unbounded area) and performed the following:

1. Find the chamber surface: The case of two separate large areas constrained by *background* was identified as two parts of the same chamber separated by an area of non-functional wire group. Those two areas are marked with the same value. The edges of these areas were linearly extrapolated (As seen by the green area in Figure 49) to form an additional bounded area designated as missing strip/wire group.
2. Find the chamber center: The center of the bounded surface representing a chamber is similar to the center of the unbounded surface representing *background*. Thus a Center of Mass algorithm (See section 6.3.4) was performed on the background to find the chamber center. All subsequent hole-positions will be given relative to it.
3. Find holes centers: Each of the bounded surfaces representing holes is scanned to find its center of mass and its area.
4. Reposition: The histogram is repositioned so that its origin is the center of the chamber.
5. Total Efficiency: The total efficiency calculated is the fraction of the efficient chamber area found out of the effective chamber area expected for a specific TGC type.

The results of this algorithm, including the efficiency of each chamber, the position and area of holes and missing wire/strip groups and an efficiency plot of the chamber are all saved in an SQL offline database for further use.

8.2.3 TightTGC

TightTGC is handheld readout device designed for TGC signal test. It is based on the TGC-Lite system produced at Weizmann Institute for the TGC final certification tests at CERN and a LabView software and hardware package meant to check TGC chambers during the final certification tests at CERN. It is used in order to test TGC connectors for abnormal behavior such as noise, dead channels and gas and HV problems. All the TGCs which arrived to CERN from Japan, China and Israel went through these tests before they were mounted on the TGC wheels [29]. The TightTGC is a handheld version which was produced in Tel Aviv for fast tests of a single unit.

The hardware package is comprised of a power source connected to a hardened box of three TGC-Lite Readout cards produced at the Weizmann Institute. Each of these has long data cables to connect to the onboard TGC connectors.

The software package was written in the LabView environment to control the readout of each of these three cards and plots results in various graphs. It was built to be a fast and easy system to operate.

In this section the usage of this package will be specified.

8.2.3.1 TightTGC layout

The TightTGC system (Figure 50) is comprised of two modules – one is the power-supply and the other is the TightTGC readout box. The power supply power-jacks are color-coded to match the readout box.

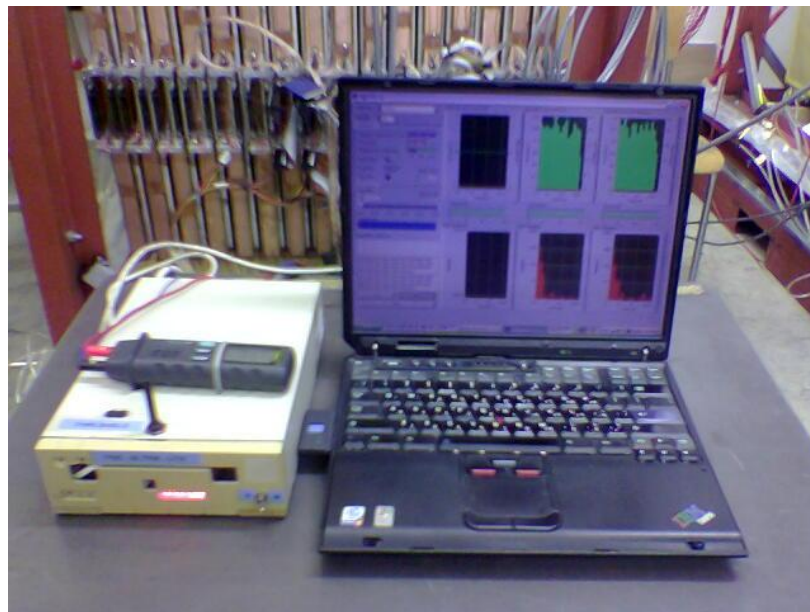


Figure 50: TightTGC system.

The system includes power supply, readout cards and computer for readout the TGC output signals.

The readout box is built as follows:

1. Color-coded power sockets. These are connected to the power-supply box.
2. USB port on the back. This is connected to the computer.
3. Power switch on the back.
4. Strip/Wire switch on the front.
5. Two 40pin flat-cables marked J11 and J12. These are connected to the TGC chamber.
6. One long red cable for ground. This is connected to the frame of the TGC chamber.
7. Two red-blue threshold cables to check the threshold value. These are connected to the voltage-meter of your choice.
8. A threshold knob on the top of the box to change threshold values. Keep in mind that Strip threshold should be around 80mV and Wires should be around -80mV.

The following operation steps are taken from the TightTGC operation Manual [30].

8.2.3.2 TGC-Lite Readout card layout

The readout box in the TightTGC system is built around a single TGC-Lite readout card. The readout card is connected on one side to the TGC ASD cards via the J11 and J12 cables and on the other to a computer via a serial to USB adapter. The detailed layout of the readout card is given in Figure 51.

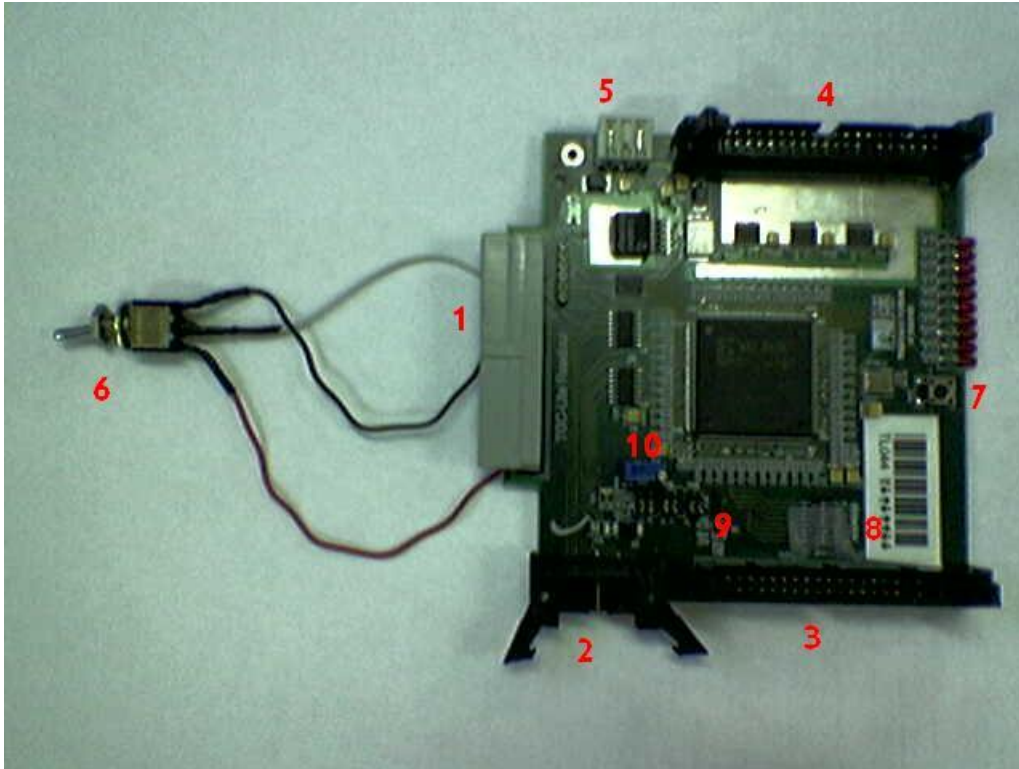


Figure 51: TGC-Lite readout card.
Different components of the card are represented by numbers and explained below.

Some of the card's key components, marked by red numbers above, are:

1. TGC-Lite readout connector: This port receives the data from the TGC chamber (ASD0). Not used.
2. Serial read out: This is a RS485 serial port, through which the TGC-Lite readout card is connected to a computer via an RS485 to USB adapter.
3. A second TGC-Lite Readout connector (ASD1). Connected to J11 cable.
4. A third TGC-Lite Readout connector (ASD2). Connected to J12 cable.
5. Voltage connector: This port you is a +/-5V DC low voltage connector for the power supply.
6. Wires/Strips selector switch: This switch chooses positive or negative threshold voltages. Negative is used for wires-mode and positive is used for strips-mode. See also entry 8).
7. Manual reset button: Pressing this button resets the ASD readout card and clears its memory buffer, overriding the controlling program. A reset is also possible via the TightTGC program.
8. Status operation switch system: This 8 up/down switch system includes one switch (rightmost) which is the wires/strips selector, and seven switches which control the card's serial Board ID.

Switch no. 1 was rerouted to the external switch (Component no. 6) for convenience. The seven switches represent a binary number which is the serial Board ID whose most significant bit (MSB) is second from the right and least significant bit (LSB) is leftmost (See Figure 52).

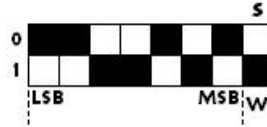


Figure 52: TGC Lite switch system.
The rightmost switch is the Wire/Strips selector. Seven leftmost switches represent the Board ID. In the current case the settings are for using the wires system on a serial address of 0x65.

9. Local/External Threshold setting jumper system: By changing the jumper location, a user can choose whether the threshold would be set locally by the potentiometer or externally. This is set to local by default.
10. Potentiometer: This sets the threshold voltage to the TGC-Lite card. For convenience and better control this was connected to an external knob (See Figure 50).

8.2.3.3 Data layout of the TGC-Lite readout card

The TGC-Lite read in connection receives from the TGC chamber an array of 35 cells; each one includes a 16-Bit word, representing the cell's data. The meaning of each cell in the array is given in Table 2.

The first 16 words each represent the hit count of a channel, which is how many events were read from the channel. The next 16 words each represent the multiplicity, which quantifies the cross-talk between channels. It is the number of times signals were received simultaneous in more than one channel. The next two words represent the time that passed from the start of the measurements until the time of reading the data (represented in two words because of its large value). The last 16-Bit word is the status word, which describes the binary status of 16 parameters of the TGC-Lite card (detailed in Table 3).

TGC-Lite readout (16-Bit words)			
Word	Meaning	Word	Meaning
0	Hit count channel 1	18	Count of multiplicity=2
1	Hit count channel 2	19	Count of multiplicity=3
2	Hit count channel 3	20	Count of multiplicity=4
3	Hit count channel 4	21	Count of multiplicity=5
4	Hit count channel 5	22	Count of multiplicity=6
5	Hit count channel 6	23	Count of multiplicity=7
6	Hit count channel 7	24	Count of multiplicity=8
7	Hit count channel 8	25	Count of multiplicity=9
8	Hit count channel 9	26	Count of multiplicity=10

9	Hit count channel 10	27	Count of multiplicity=11
10	Hit count channel 11	28	Count of multiplicity=12
11	Hit count channel 12	29	Count of multiplicity=13
12	Hit count channel 13	30	Count of multiplicity=14
13	Hit count channel 14	31	Count of multiplicity=15
14	Hit count channel 15	32	Timer high
15	Hit count channel 16	33	Timer low
16	Count of multiplicity=16	34	Status
17	Count of multiplicity=1		

Table 2: The TGC-Lite readout data format.

Hit count is the number of hits per channel, count of multiplicity is the number of events with signals hitting 1,2,3... 16 channels simultaneously. Timer hi and low represent the time duration from the beginning of the run. Status represent the status of the TGC-Lite readout card.

Status Register (16 Bits)		
Bit	meaning	Example of status
0	threshold 1 control	Threshold 1 is local
1	threshold 2 control	Threshold 2 is local
2	threshold 3 control	Threshold 3 is local
3	-3V status	-3V is OK
4	+3V status	+3V is OK
5	ASD0 enable	1=enable
6	ASD1 enable	1=enable
7	ASD2 enable	0=disable
8	Wire/strips selection	1=wire, 0=strip
9	Run/ hold status	1=run
15	Bad command(error)	1=bad

Table 3: The TGC-Lite status register

Communication with the TGC-Lite readout card is done using a list of commands. Prior to each command, an address needs to be sent, representing the board IDs of the card need to perform the command. To do that the “DEADFACE” string followed by the card’s board ID is sent. The card then returns an 8-bit signal containing its board ID, signifying that it received the address and is ready to perform the command. After which the Hex code of the command can readily be sent and the return data read back. A detailed list of commands is given in Table 4.

The “ASD enable mask” command determines which of the three available ASD ports will be on/off. The command is of the form $0x2m$, when m is a three-bit binary number. Each bit represents whether its corresponding ASD port is on or off.

TGC-Lite readout (16-Bit words)				
Command	Explanation	Code	Comments	Returns
Reset	Resets card	0x30	If address==0, all units respond	
Acquire	Start storing data in the memory buffer (according to the pre-fixed ASD enable mask)	0x31	If address==0, all units respond	
Do test pulse	Send a test pulse	0x32	If address==0, all units respond	Status
Stop	Stop storing data in the memory buffer	0x33	If address==0, all units respond	Status
Read ASD0	Release all the data in the memory buffer for reading	0x80		The 35 cell array
Read ASD1	Release all the data in the memory buffer for reading	0x81		The 35 cell array
Read ASD2	Release all the data in the memory buffer for reading	0x82		The 35 cell array
Read status	Read card status	0x83		Status
Set for strips	Set the card to strips mode	0x40		Status
Set for wires	Set the card to wires mode	0x41		Status
Clear memory buffer	Clears card's memory buffer	0x42		Status
ASD enable mask	Turns each of the ASD ports on/off depending on the value of m	0x2m	ASD0 port corresponds to the 1 st bit. ASD1 port corresponds to the 2 nd bit. ASD2 port corresponds to the 3 rd bit.	Status

Table 4: List of commands for the TGC-Lite readout card.

8.2.3.4 Software Installation

1. Extract the enclosed "TightTGC Installer.exe" file into any directory.
2. In that directory run the setup.exe file and follow its instructions.
3. Make sure to install the RS232-RS485 Adapter drivers and the USB Serial Port drivers (you will be prompted to do so when the device is first connected).

8.2.3.5 TightTGC Software

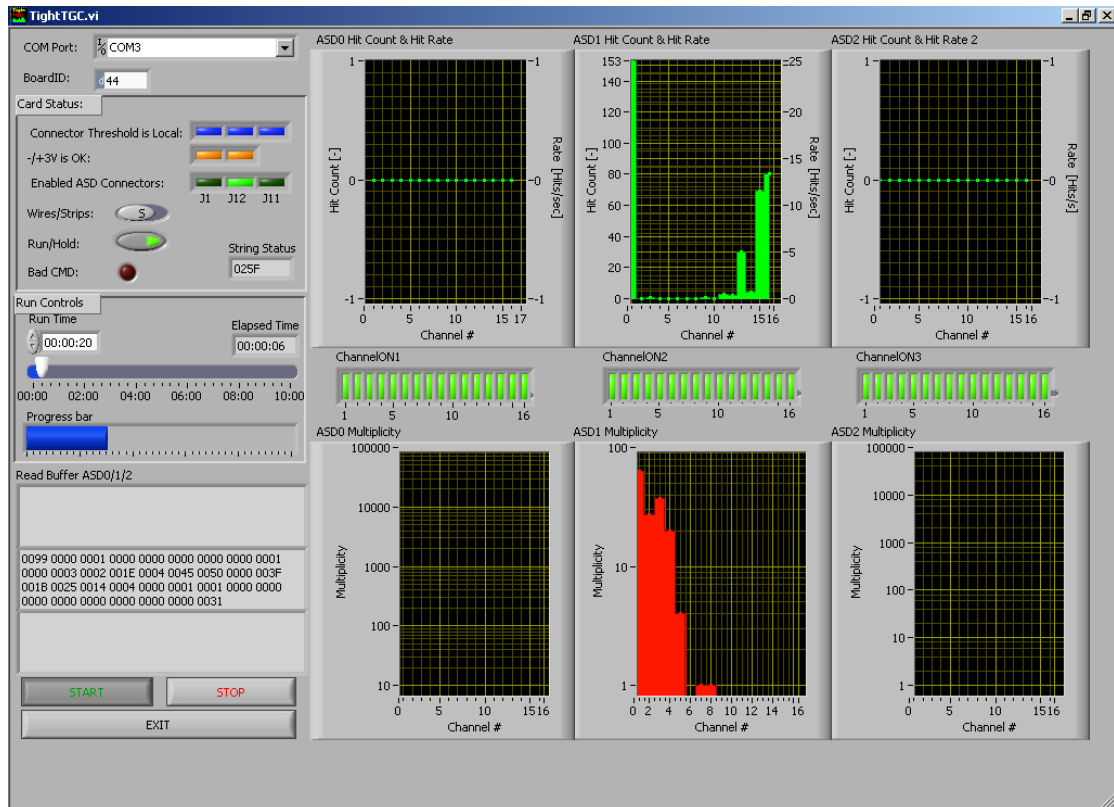


Figure 53: TightTGC main screen

1. COM Port: User must specify the COM port to which the device is connected to. See Section 5 on how to locate your COM port number.
2. BoardID: The Board Id of the TGC Lite card. Program automatically scans for it.
3. Connector Threshold is local: Indicator to whether the threshold is set locally or not. All 3 LEDs must always be ON.
4. -/+ 3V is OK: Indicator to whether the TGC Lite card gets proper voltage. Two LEDs must always be ON.
5. Enabled ASD Connectors: Check each LED to activate the respective connector.
6. Wires/Strips: User must specify whether the system is connected to wires or strips. Note that the switch on the front panel of the TightTGC box must also be set accordingly.

7. Run/Hold: Indicator to whether the TGC Lite card is running. Must always be ON.
8. Bad CMD: Indicator of whether a bad command was sent to the TGC Lite card. Must always be OFF.
9. Run Time: User must specify how much time the system should run. Default value is 20 seconds.
10. Elapsed Time: The elapsed time since the START button was hit.
11. Read Buffer ASD0/1/2: The raw data read from the three ASD connectors.
12. ASD Hit Count & Hit Rate: A graph specifying the number of events on the left and the events rate on the right read from the TGC chamber.
13. Multiplicity: A graph specifying the how many channels were hit simultaneously on n^{th} connected chamber.
14. ChannelON 1-3: A control that allows the user to disconnected specific channels from the ASD readout. Note that the channels are *not physically disconnected*, but rather that their value is ignored (hence there will be no change to the multiplicity count when a channel is disconnected).

8.2.3.6 System set-up

1. Connect the TightTGC box to a computer via USB.
2. Plug in the power jacks from the power-supply box.
3. Connect the J11 and/or J12 cables to the TGC chamber.
4. Switch on the power box and the TightTGC box.
5. In the TightTGC program, set the desired run time.
6. Hit START.

8.2.3.7 Checking COM Port

1. Plug the TightTGC USB cable into the computer and make sure all the drivers specified in Section 2 are installed.
2. Click Start→Settings→Control Panel.
3. Double-click System.
4. In the opened dialog box click the Hardware tab.
5. Click the Device Manager button.
6. In the opened windows, double-click the “Ports (COM & LPT)” entry.
7. The COM port written as USB Serial Port is the port the system is connected to.

8.2.3.8 Identifying problems with a TGC chamber

Using the TightTGC software makes it easy to locate problems in the TGC chamber readout. Three possible scenarios will be shown below

Threshold problem

This problem occurs whenever the voltage threshold for the TGC-Lite card is set too low and the readout becomes saturated. The effect of which is shown in Figure 54.

In the case of the good threshold (Figure 54b), the hit counts graph show even response with a rate of about 25 hits per second and very low multiplicity. In the case of the low threshold (Figure 54a), the hit count is saturated and the multiplicity graph show as many as 13 channels giving simultaneous signals. This may also be the case if the entire TGC chamber is too noisy.

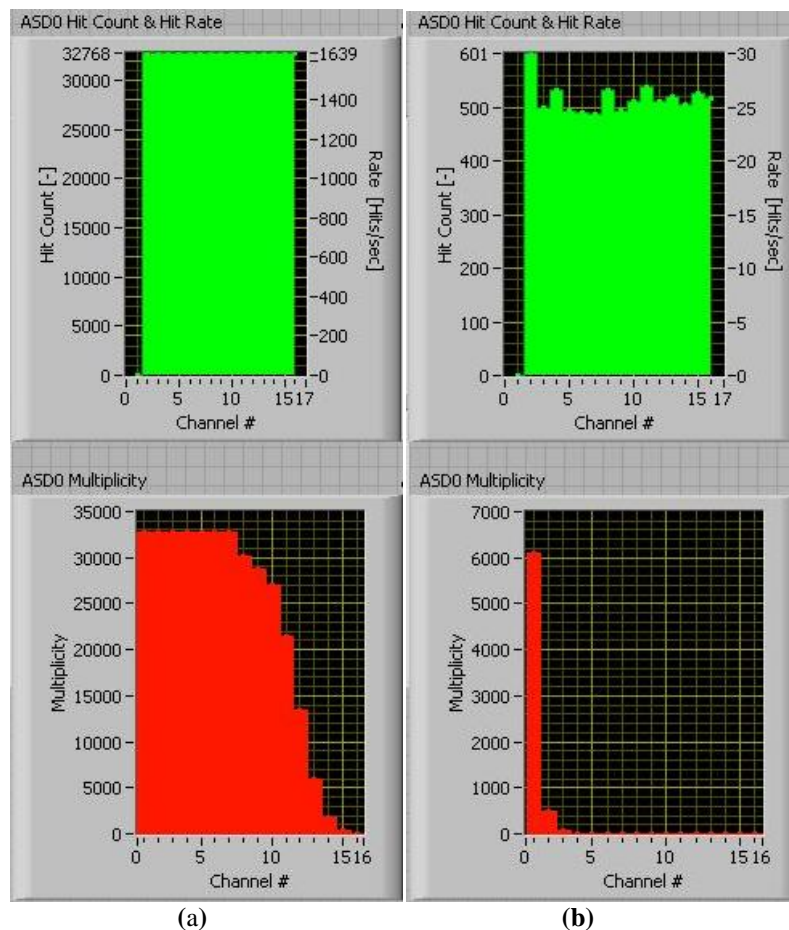


Figure 54: TGC-Lite hit count and multiplicity in two threshold settings.
(a) Threshold set too low, or the TGC chamber as whole is too noisy.
(b) Correct threshold.

Noisy/Missing channels

A noisy TGC channel is easy to spot using TightTGC. It will be characterized by a peak in the hit count graph with a count rate considerably higher than the expected 25 hits per second of the cosmic flux. The multiplicity count will also be affected as the noisy channel often sends signals in tandem with other channels. The effects of this can be seen in Figure 55a. In such a case it is possible to tell TightTGC to ignore this channel to get a better picture of the rest of the channels. This is done by selecting the Channel-ON control (See section 8.2.3.5).

A missing TGC channel is demonstrated in Figure 55b. It can be caused by a faulty wire or an intentional disconnection of a noisy wire in the TGC chamber.

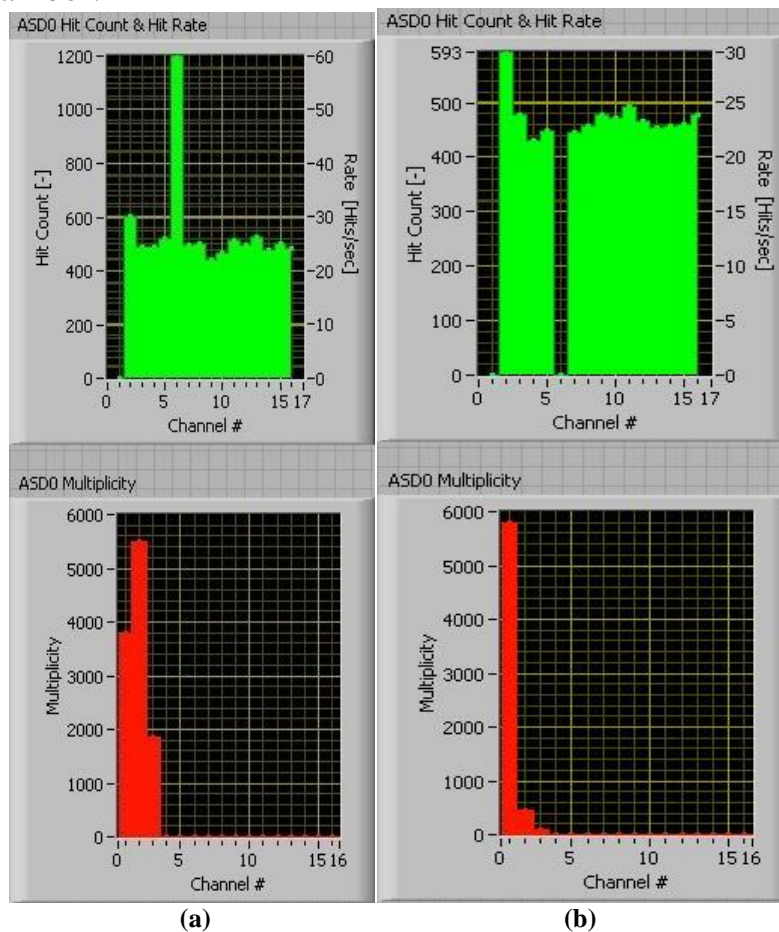


Figure 55: TightTGC hit count and multiplicity for noisy/missing channels.

- (a) Channel #6 is noisy.**
- (b) Channel #6 is missing.**

Channel Cross-talk

Cross-talk between channels occurs when one channel induced a signal on another nearby channel. This can happen due to irregular spacing between the channels or faulty insulation which causes electric discharge

of one channel on the other. Cross-talk causes the two (or more) channels to act similarly.

Cross-talk can clearly be identified in the TightTGC system by closely inspecting the multiplicity graph. In any case where the multiplicity is not dominated by a single channel it is likely caused due to cross-talk (See Figure 56). High cross-talk rate is usually measured in channels that are close to a chamber's power-supply due to faulty insulation.

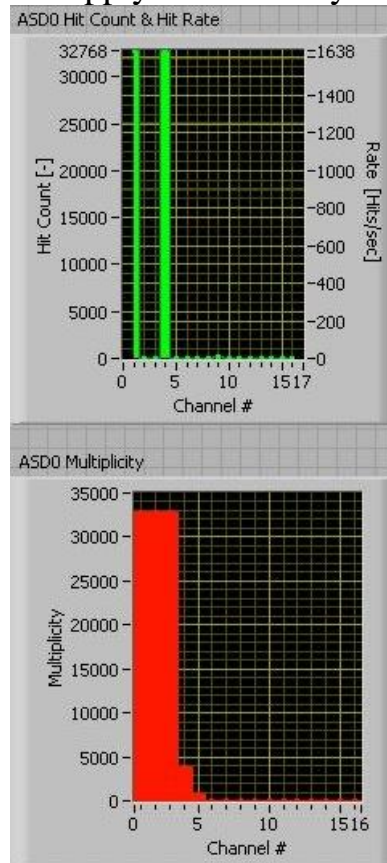


Figure 56: TightTGC channel hit count and multiplicity for cross-talk channels.

8.2.3.9 TightTGC summary

The TightTGC provided a handheld testing device which enabled to test and track problems in TGC detectors in a fast and easy methodology.

10References

- ¹ ATLAS Technical Proposal; CERN / LHCC 94-43; LHCC / P2; December 1994
- ² S. Glashow, Nucl. Phys. 22 (1961) 579;
S. Weinberg, Phys. Rev. Lett. 19 (1967) 1264;
A. Salam, "Elementary Particle Physics", W. Svartholm, ed., Almquist and Wikesel, Stockholm, 1968.
- ³ P.W. Higgs, Phys. Rev. Lett. 12 (1964) 132 and Phys. Rev. 145 (1966) 1156 ; F. Englert and R. Brout, Phys. Rev. Lett. 13 (1964) 321.
- ⁴ E.Richter-Was et al., "Minimal Supersymmetric Standard Model Higgs rates and backgrounds in ATLAS", ATLAS Internal Note ATL- PHYS-96-074 (1996) , published in Int. J. Mod. Phys. A13 (1998) 1371
- ⁵ ATLAS Detector and Physics Performance Technical Design Report; ATLAS TDR 14; CERN/LHCC 99-14; May 1999.
- ⁶ ATLAS Inner Detector Technical Design Report; ATLAS TDR 4; CERN/LHCC 97-16; April 1997.
- ⁷ Calorimeter Performance Technical Design Report; CERN/LHCC 96-40; January 1997.
- ⁸ Tile Calorimeter Technical Design Report; CERN/LHCC 96-42; December 1996.
- ⁹ ATLAS Muon Spectrometer Technical Design Report; CERN/LHCC 97-22; June 1997.
- ¹⁰ N. Drego, D. Hawkins, A. J. Lankford, Y. Li, M. Medve, S. Pier, M. Schernau, D. Stoker, "Off-Detector Electronics for a High-Rate CSC Detector"; Proceedings of 6th Workshop on Electronics for LHC Experiments; Krakow, Poland; 2000.
- ¹¹ ATLAS Computing Technical Design Report; ATLAS TDR--017, CERN-LHCC-2005-022; June 2005.
- ¹² <https://twiki.cern.ch/twiki/bin/view/Atlas/IPatRec>.
- ¹³ <https://twiki.cern.ch/twiki/bin/view/Atlas/InDetXKAlman>.
- ¹⁴ Th. Lagouri, D. Adams, K. Assamagan, M. Biglietti, G. Carlino, G. Cataldi, F. Conventi, A. Farilla, Y. Fisyak, S. Goldfarb, E. Gorini, K. Mair, L. Merola, A. Nairz, A. Poppleton, M. Primavera, S. Rosati, J. Shank, S. Spagnolo, L. Spogli, G. Stavropoulos, M. Verducci, T. Wenaus; "A Muon Identification and Combined Reconstruction Procedure for the ATLAS Detector at the LHC at CERN"; 2004.
- ¹⁵ Tarem S. et al. "Muon Identification in ATLAS from the inside out".
- ¹⁶ D. Primor; "Local Tracking in the ATLAS Muon Spectrometer"; PhD Thesis, Tel Aviv University, 2007.
- ¹⁷ Athena Developer Guide;
<http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/General/Documentation/AthenaDeveloperGuide.pdf>; May 2002.
- ¹⁸ S. Giani et al., Geant4: "Object Oriented Toolkit for Simulation in HEP", LCB status report CERN/LHCC/98-44, November 1998, S. Agostinelli et al , GEANT4 a simulation toolkit, Nucl.Instrum.Meth.A506:250-303,2003.
- ¹⁹ S. Agostinelli et al., "GEANT4 a simulation toolkit", Nucl.Instrum.Meth.A506:250-303,2003.
- ²⁰ V. Boisvert et al., "Final Report of the ATLAS RTF", ATLAS note ATL-SOFT-2003-010.

-
- ²¹ F. Akesson, T. Atkinson, M.J. Costa, M. Elsing, S. Fleischmann, A. Gaponenko, W. Liebig, E. Moyses, A. Salzburger, M. Siebel; “ATLAS Tracking Event Data Model”; July 2006.
- ²² R. Fruhwirth et al., “Data Analysis Techniques for High-Energy Physics”, Cambridge monographs on particle physics, nuclear physics and cosmology, 2000.
- ²³ Illingworth J., Kittler J., “A Survey of the Hough Transform, Computer vision, graphics and image processing”, 44(10):87-116, 1988.
- ²⁴ Duda, R.O and Hart, P.E. Use of the Hough Transform to Detect Lines and Curves in Pictures, *Commun. Ass. Comput.*, 15:11-15, 1972.
- ²⁵ E. Mathieson and J.S. Gordon, “Cathode charge distributions in multiwire chambers II. Approximate and empirical formulae”, *Nucl. Instrum. Methods A227* (1984) 277.
- ²⁶ Gordeev, A. et al., “CSC performance at High Background Rates”, ATL-MUON-2000-005; Geneva : CERN, 1999.
- ²⁷ D. Primor, G. Mikenberg, N. Amram, E. Etzion, H. Messer; “The use of cluster quality for track fitting in the CSC detector”; Proceedings of 2006 IEEE NSS; San Diego, California, USA; November 2006.
- ²⁸ E. Etzion, H. Abramowicz, N. Amram, Y. Benhammou, M. Ben-Moshe, G. Bella, J. Ginzburg, Y. Gernitzky, A. Harel, H. Landsman, N. Panikashvili, Y. Rozen, S. Tarem, E. Warszawski, J. Wasilewski, L. Levinson “The Cosmic Ray Hodoscopes for Testing Thin Gap Chambers at the Technion and Tel Aviv University”, *Trans. on Nucl. Sci.*, volume 51, number 5, pp 2091-2096 (2004)
- ²⁹ E. Etzion for the ATLAS TGC, “The Certification of ATLAS Thin Gap Chambers Produced in Israel and China”, IEEE Nuclear Science Symposium 2004; Rome, Italy; October 2004, [physics/0411136]
- ³⁰ N. Amram, M. Ben Moshe, E. Etzion, “Tight-TGC operation Manual” April 2005.