A NEW TRACKING ALGORITHM FOR ATLAS TRIGGER

by

Saime SARIKAYA

B.S., in Physics & Mathematics, Boğaziçi University, 2009

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Physics
Boğaziçi University

2012

A NEW TRACKING ALGORITHM FOR ATLAS TRIGGER

APPROVED BY:

Assoc. Prof. Erkcan Özcan (Thesis Supervisor)	.....
Assoc. Prof. Taylan Akdoğan	.....
Prof. Serkant Çetin	.....

DATE OF APPROVAL: 01.11.2012

This thesis is dedicated to my mom and dad,
Müzeyyen & Hikmet Sarıkaya

ACKNOWLEDGEMENTS

First of all, I would like to express my great appreciation to *Hans Drevermann*, who gave his years to physics and despite his age still continue his studies. I will always follow his lead in my life. This thesis is mainly constructed on his studies, and he tried to pass all his experiences to me. At this point, I also would like to express my great thanks to *Hans Grote*. Even we have not met him face to face yet, he helped me a lot. Many Thanks to *Pauline* for providing data sets.

Another person, who has a great afford on my thesis is *Erkcan Özcan*, my thesis supervisor. He not only helped me to maintain and finish my studies, but also with his friendly, understanding and tolerant manner, provided me moral support. From him, I learned how to be an experimental physicist.

Many thanks to everyone in the lab and at CERN, who always helped me, and made my life more tolerable. And I owe much to my professors at Boğaziçi University, especially professors in my thesis committee, *Taylan Akdoğan*, *Serkant Çetin*, *Metin Arık*.

Great thanks to *Mustafa*, who supported me for all of my life, especially, while writing my thesis. I wish his support and his love will always be with me.

I also appreciate my family. They grow me up and supported me at every stage of my life. I wish I will support my brother *Yusuf* for all of his life, as my family did to me. Thank you for believing in me.

ABSTRACT

A NEW TRACKING ALGORITHM FOR ATLAS TRIGGER

The Large Hadron Collider (LHC) is the world's highest energy accelerator with the highest luminosity, which aims to find answers to the most fundamental questions of particle physics. It leads us to the discoveries of new particles (e.g discovery of Higgs boson nowadays), as predicted by Standard Model and its extensions like supersymmetry. However, coping up with the data produced by the LHC, and finding the interesting events, is not an easy task. LHC produces 1TB of data per second and approximately only 1 event per million is from interesting physics. At this point, looking for an event in the whole data is like trying to find a needle in the haystack and to find the interesting events out of the entire data, trigger systems are used. One of the most important part of trigger is tracking, which one needs to identify the tracks made by particles, and to identify the physics objects in the event. In this thesis, we describe an algorithm, to run at one of the LHC detectors, ATLAS, in order to achieve fast tracking. The algorithm is tested, using very high luminosity simulation data both running on hits obtained from the entire detector and also in certain small regions around interesting signals. The performance is shown to be satisfactory both in terms of tracking efficiency and in terms of timing requirements.

ÖZET

ATLAS TETİKLEME İÇİN YENİ BİR İZ SÜRME ALGORİTMASI

Parçacık Fiziğinin temel sorularını yanıt arayan, Büyük Hadron Çarpıştıcısı (LHC) dünyanın en yüksek parlamasına sahip ve en yüksek enerjili parçacık hızlandıcısıdır. En güncel örneği Higgs bosonu olmak üzere, Standart Model ve onun uzantılarının öngördüğü bir çok yeni parçacığın keşfine önderlik yapmıştır. Fakat, LHC'nin ürettiği veriden, ilginç fizik olaylarını bulmak o kadar da kolay bir iş değildir. Saniyede 1TB veri üretilmesine karşın, bu veri içerisinden sadece milyonda bir olayda, ilginç fizik olayı tespit edilmektedir. Bütün veri içerisinden bu ilginç fizik olayını bulmak samanlık iğne aramaya benzer. LHC dedektörlerinde, ilginç fizik olaylarını bulmak için tetikleme sistemleri kullanılır. Ayrıca çarpışma sonucu oluşan parçacıkların tanımlanabilmesi için iz sürmeye ihtiyacımız vardır. Bu sebeple tetikleme sistemlerinin en önemli parçalarından biri de iz sürmedir. Bu tezde LHC dedektörlerinden biri olan ATLAS için yeni bir hızlı iz sürme algoritması anlatılmıştır. Algoritma hem yüksek parlaklıklı bütün dedektörü kaplayan simülasyon olayının noktaları ile hem de içerisinde ilginç sinyaller olan belirlenmiş küçük alanlardaki noktalar için test edilmiştir. Hem iz bulma verimliliği hem de zamanlama koşullarını yerine getirme açısından tatmin edici bir performansa sahiptir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv
ABSTRACT v
ÖZET vi
LIST OF FIGURES ix
LIST OF TABLES xii
LIST OF SYMBOLS xiii
LIST OF ACRONYMS/ABBREVIATIONS xv
1. INTRODUCTION 1
2. ATLAS Detector System 4
2.1. ATLAS Detector System Overview 4
2.1.1. Inner Detector 5
2.1.1.1. Pixel and SCT Detector 5
2.1.1.2. TRT Detector 6
2.1.2. Calorimeter 7
2.1.3. Muon spectrometer 8
2.1.4. Magnet System 9
2.2. ATLAS Trigger 9
2.2.1. Level 1 Trigger 10
2.2.2. Level 2 Trigger 11
2.2.3. Event Filter 12
3. TRACKING 13
3.1. ATLAS Level 2 Trigger Tracking Algorithms 13
3.1.1. SiTrack Algorithm 14
3.1.2. IDScan Algorithms 15
3.2. Helix Equations 17
3.3. Z-Finder Algorithm 19
3.4. V-Plot 25
3.5. Hit Filter Algorithm 32
4. ANALYSIS OF PERFORMANCE 40

4.1. Performance of the Z-Finder 40
4.2. Performance of the Hit Filter 42
4.3. Timing Performance 44
5. CONCLUSION 47
APPENDIX A: Z-Finder Algorithm in C programming language 48
APPENDIX B: Hit Filter algorithm in C programming language 52
APPENDIX C: Mathematical Proof of the shape V in V-plot 55
REFERENCES 57

LIST OF FIGURES

Figure 1.1. Overall view of LHC [1]. 1

Figure 1.2. Rates of Trigger levels and physical events [3]. 2

Figure 2.1. ATLAS Detector system [1]. 4

Figure 2.2. Inner Detector of ATLAS [1]. 5

Figure 2.3. Scheme of inner detector [1]. 6

Figure 2.4. ATLAS Trigger architecture and data flow [4]. 10

Figure 2.5. Example of region of interest (RoI) [5]. 11

Figure 3.1. A simulated $H \rightarrow ZZ \rightarrow 4\mu$ event with the associated large track density foreseen at high luminosity [1]. 13

Figure 3.2. $x - y$ projection of helix in cylindrical coordinates. 18

Figure 3.3. Histograms of z position of the primary vertex, after applying 2 or 3 nested loops, on 3 different data sets. 23

Figure 3.4. Histograms of z position of the pile-up events, after applying 3 nested loops. 24

Figure 3.5. Inner detector hits divided into bins [11]. 25

Figure 3.6. A three dimesional point is represented with a line in 2D. 26

Figure 3.7. Compression of points from diffirent kinds of detectors. 26

Figure 3.8. (a)Compression to identify kinks (b)Compression for vertical and oblique tracks (c)Compression in a noisy enviroment [12]. 27

Figure 3.9. ATLAS inner detector points (a) on $x-y$ plane (b) projection on $\phi-\rho$ plane (c)compression of $\phi-\rho$ projection [12]. 28

Figure 3.10. Construction of V-plot by using 2 high momentum tracks coming from the primary vertex [12]. 30

Figure 3.11. Examples of representing some tracks via V-plot [12]. 31

Figure 3.12. Principle of cleaning and grouping [13]. 34

Figure 3.13. An example to layer histogram. 35

Figure 3.14. Skew transformations applied to two tracks [13]. 37

Figure 3.15. Results of hit filter running on full detector data [13]. 38

Figure 3.16. Comparision of results of hit filter with the stand alone simulation results [13]. 39

Figure 4.1. Event display for the example event used in the optimization for full detector running. Projections are shown in: a)$x-y$ plane b)$\rho-z$ plane (barrel part) c)$\rho-z$ plane. Due to the large number of hits the detector module are clearly visible [13]. 41

Figure 4.2. Comparison of results of z-finder and the simulation, for primary (left) and secondary (right) vertices [11]. 42

Figure 4.3. Difference between the primary vertex positions from the z-finder and the simulation. 43

Figure 4.4. Lost hits from the SCT layers, because of shifts on SCT hits on V-plot [12]. 44

Figure 4.5. η resolution of hit filter. 45

Figure 4.6. ϕ resolution of hit filter. 46

LIST OF TABLES

Table 4.1. z-finder and hit filter timing results for full detector and RoI data. 46

LIST OF SYMBOLS

H	Higgs Boson
I_1	Number of entries of the cell of the histogram
I_2	Flag of the cell of the histogram
I_{sum}	Sum of entries of the cell and the neighbours of the histogram
k	V-plot parameter
l	Lepton
N_2	Flag of the hit
N_E	Minimum number of hits required to accept a cell
N_G	Group number of hits
P_{cut}	Minimum momentum of tracks, required to be selected
P_T	Momentum of tracks
R_0	Radius of the helix measured from the center of helix
R_{cut}	Minimum radius of helix, required to be selected
s	Skew parameter
x, y, z	Usual Cartesian coordinates
x_c, y_c, z_c	Coordinates of the center of the helix in Cartesian system
z	Distance of hits along the beamline from the middle of detector
Z	Z boson
z_0	Initial z position of track
z_i	z position of ith hit
z_v	z position of the interaction point
γ	Gamma particle
$\Delta\alpha$	Angular difference of hits relative to the center of helix
η	Pseudorapidity
$\eta'_{\pm}$	Pseudorapidity for the arms of V-plot
λ	Angle of the hit from the $x-y$ plane
λ_0	Initial angle of the track from the $x-y$ plane

ρ	Radial distance of hits from the z axis
ρ"	Maximum radius of inner detector for ATLAS setup
ρ_i	ρ value of ith hit
σ	Cross section
ϕ	Azimuthal angle of hits between the reference direction and the chosen plane
ϕ_0	Initial ϕ value of track, while going from the beamline to outer layers of detector
ϕ_i	ϕ value of ith hit

LIST OF ACRONYMS/ABBREVIATIONS

ATLAS	A Toroidal LHC ApparatuS
CERN	European organization for nuclear research
EF	Event Filter
EM	Electromagnetic
ID	Inner Detector
LAr	Liquid Argon
LHC	Large Hadron Collider
LVL1	ATLAS Level 1 Trigger System
LVL2	ATLAS Level 2 Trigger System
RoI	Region of Interest
SCT	Semiconductor Tracker
TRT	Transition Radiation Tracker

1. INTRODUCTION

Large Hadron Collider (LHC) is the world's largest and highest energy particle accelerator [1]. It has been built by European Organization for Nuclear Research (CERN) in a circular tunnel, which is about 1000 m under ground with a circumference of 27 km. It is designed to collide two counter rotating beams of protons at the center of mass energy of 14 TeV and luminosity of 10^{34} cm^{-2} s^{-1}, and heavy ions. These beam bunches are to pass through every each other 25ns, and each bunch crossing has 23 events on average, where most of the events produced are pile up events, not much interest for the search of new physics. Beams move around the LHC ring inside a continuous vacuum and are guided by magnets which are cooled by the cryogenics system at 1.9K. On the 10 of September 2008, LHC achieved first successful beam, and on 30th of March 2010 first collision data were taken. As of September 2012, it is running at 8TeV energy and around $5 \cdot 10^{33}$ cm^{-2} s^{-1} luminosity.

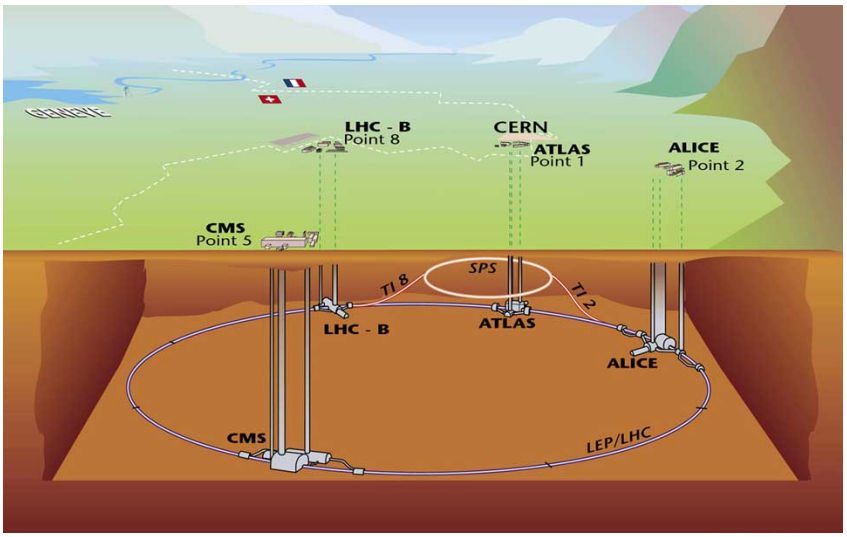

Figure 1.1. Overall view of LHC [1].

LHC is designed to answer fundamental questions like the existance of the presence and the nature of dark energy, dark matter, extra dimensions, Higgs and supersymmetry. To answer these questions and others, 6 experiments are held at the LHC (Figure 1.1):

- ALICE (A Large Ion Collider Experiment)
- ATLAS (A Toroidal LHC ApparatuS)
- CMS (Compact Muon Solenoid)
- LHCb (Large Hadron Collider beauty experiment)
- LHCf (Large Hadron Collider forward experiment)
- TOTEM (TOTal Elastic and diffractive cross section Measurement)

Among these experiments, ATLAS and CMS are multi-purpose detectors i.e they investigate the largest range of physics possible. The main physics goals of the ATLAS experiment include the understanding of the electroweak symmetry breaking, either by discoverin the Higgs boson or by measuring vector boson scattering, searching for new phenomena like super-symmetry, technicolor, leptoquark, extra dimensions etc.

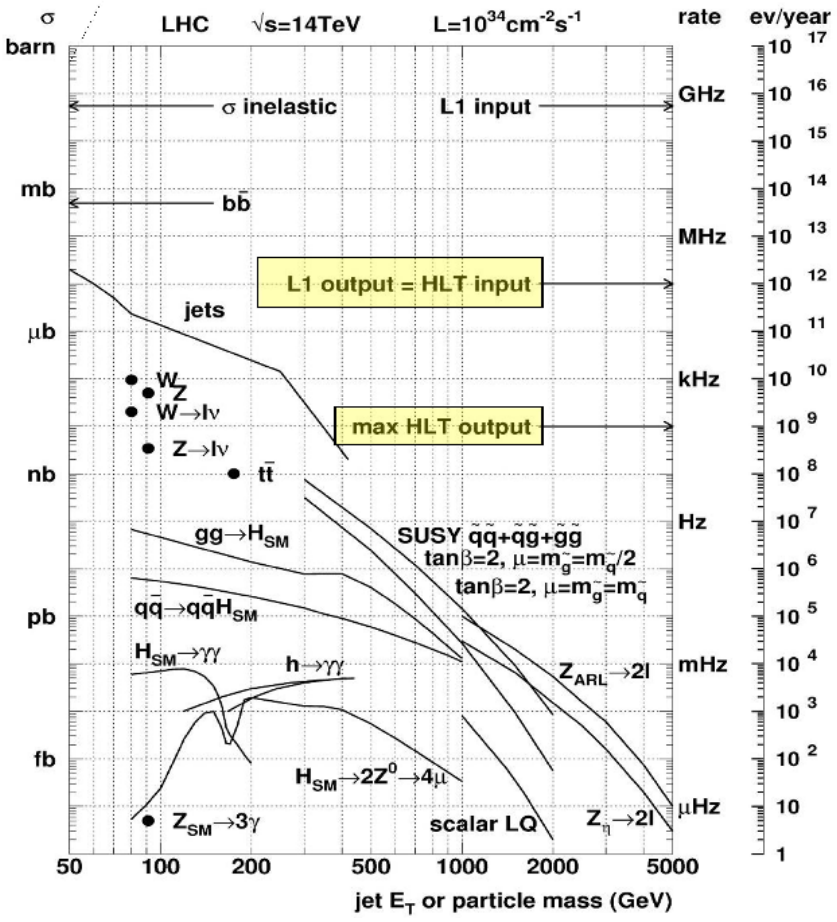

Figure 1.2. Rates of Trigger levels and physical events [3].

To select a very small single event out of a huge set of data is a tough job. For

example, while the total proton-proton cross section at 8TeV is around 110 mb, total Higgs boson cross section is roughly 50pb [2]. Only one out of 10^9 events produces a Higgs boson. Hence, for all these discoveries, one has to cope up with extremely high luminosity. It is impossible to record all the events. As a result, events should be analysed quickly and only a small fraction be selected. This is done by the trigger system.

One of the most of important parts of the trigger is tracking. For example, most favourable channels for the discovery of the Higgs boson rely on either photon or lepton identification, both of which needs high performance tracking. Whether the event is $H \to \gamma\gamma$ or $H \to ZZ \to 4l$, identification of a γ or l can not be achieved from raw detector data. To identify and find properties of these particles, the tracks should be known.

In this thesis, firstly the detector and the trigger system of ATLAS are described, followed by a general description of tracking and ATLAS tracking. A new modified tracking algorithm is then introduced and the performance of this algorithm is discussed, before we conclude.

2. ATLAS Detector System

2.1. ATLAS Detector System Overview

The Large Hadron Collider has been designed to produce proton and heavy ion collisions at 40 MHz bunch crossing rate [1]. The ATLAS detector is a general-purpose detector designed to record data from all kinds of collisions produced by the LHC. The detector system has 4 basic parts including; inner detector, calorimeter, muon spectrometer and magnet system. These can be seen in Figure 2.1. Inner detector measure the momenta of charged particles and it is used for tracking. Calorimeters measure the energy of particles. Muon spectrometer identifies muons and measures the momenta. Lastly magnet system is needed for momentum measurements of charged particles. ATLAS coordinate system is defined in usual cylindrical coordinates, where z axis is along the beam line, and the origin is placed in the middle of interaction region along z axis. x and y coordinates are defined from the beamline into the detector system ,so as ρ. Pseudorapidity, η, is used to define the angle of a hit relative to beam axis and P_T is used as transverse momentum of the particle.

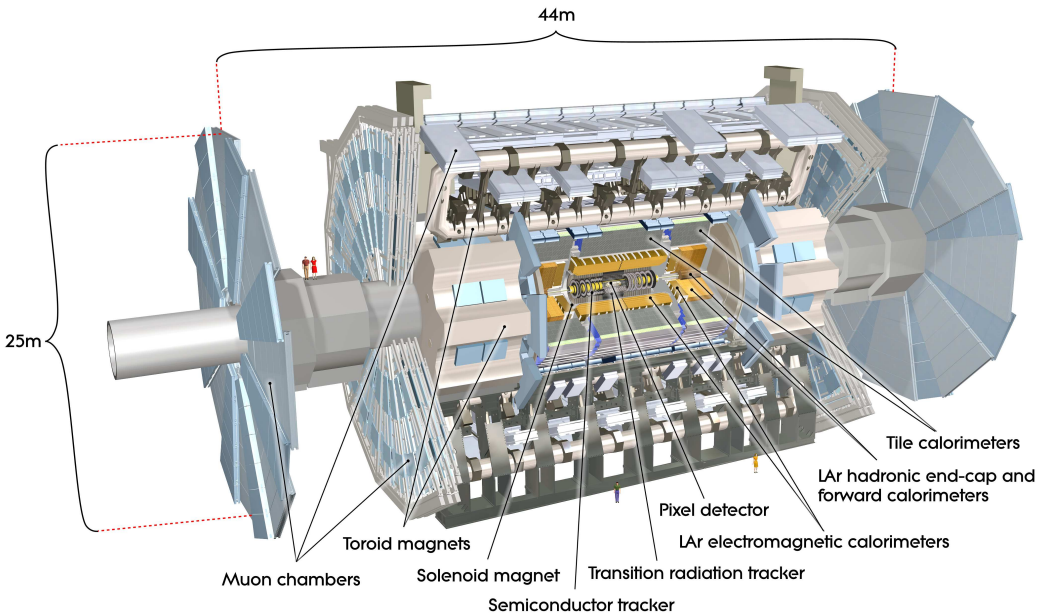

Figure 2.1. ATLAS Detector system [1].

2.1.1. Inner Detector

ATLAS Inner Detector (ID) is used for tracking and measuring the momenta of charged particles [1]. It is designed to work on a high luminosity and noisy environment. Vertexing is another function of the inner detector. The ID has 3 parts: Pixel Detector, Semiconductor Tracker (SCT) detector and Transition Radiation Tracker (TRT) detector. It is a combination of high resolution modules at the inner radii and continuous tracking elements at the higher radii. It has a total length of 7 m, and an overall of radius of 1.15m.

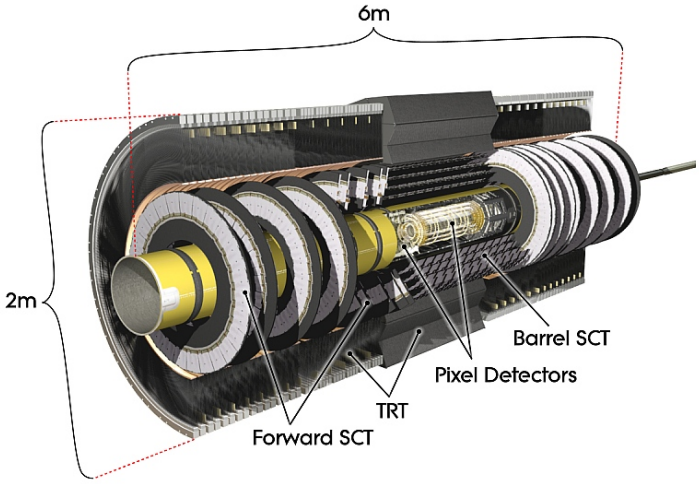

Figure 2.2. Inner Detector of ATLAS [1].

2.1.1.1. Pixel and SCT Detector. Pixel and SCT (Semiconductor Tracker) detectors provide high resolution pattern recognition using discrete space points [1]. There is a barrel part where detectors are placed as concentric cylinders surrounding the vertex of interaction, and end-cap parts where they are placed as disks. In the inner part of barrel, there are 3 layers of pixel detectors and in the outer part 4 layers of SCT (strip) detectors. On the end-cap part there are 3 pixel and 9 SCT detector layers on each side. Scheme of layers can be seen in Figure 2.3 where the z direction is defined along the beam line. Operating temperature is to be kept at below -7 C^o to maintain adequate noise performance after radiation damage, an evaporative fluorocarbon cooling system operating at -25 C^o is used.

In the pixel detectors, 250 μm thick silicon is used. Oxygenated n type wafers have readout pixels implanted in n^+ doping. Oxygenated n type wafers increase radiation tolerance to charged hadrons and n^+ implanted side provides good charge collection efficiency. Each pixel module is 62.4 mm long and 21.4 mm wide, There are 46080 pixel elements read out by 16 chips on each module. 1.7m^2 coverage area is provided by 80 million pixels of size 50μm$\times$400μm and the thickness of 280μm. Pixel detector occupies 50% of the total readout channels in the ATLAS detector. It provides extremely precise tracking very close to the interaction point. On pixel sensors, short-lived particles like B Hadrons can be identified as innermost layers are installed close to the beam line.

The difference of SCT detector from pixel detector is that the SCT has long narrow strips rather than small pixels and it covers a larger area. It is the basic element for tracking in the plane perpendicular to the beam axis with more sampled points and roughly equal accuracy. SCT detector has the thickness 285μm. Classical single sided p n technology is used in the sensors. Each silicon detector is 6.36$\times$6.40cm^2 with 780 readout strips of 80μm pitch. Layers in the barrel part of the SCT can provide precision points in the $\rho - \phi$ and z coordinates, using small angle stereo to obtain the z measurement.

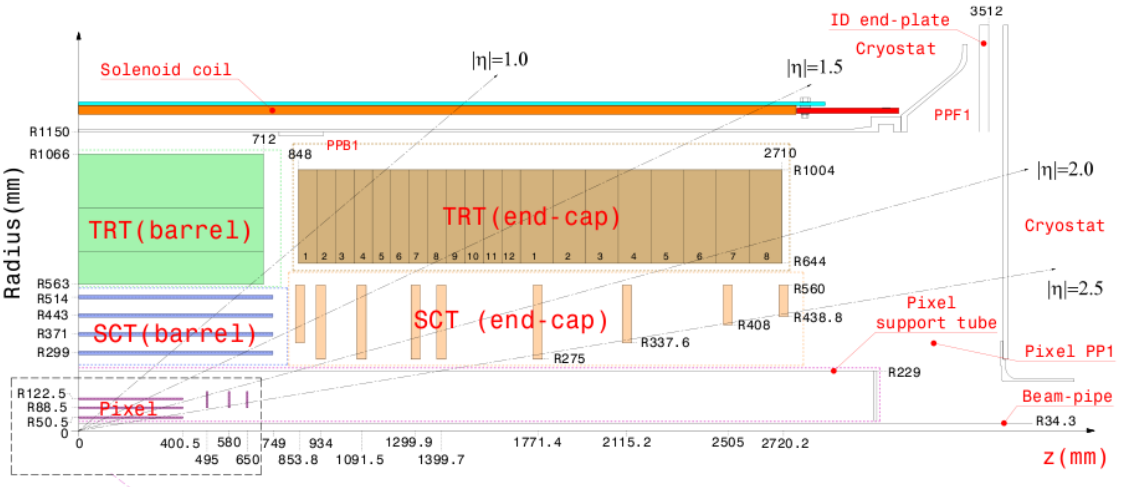

Figure 2.3. Scheme of inner detector [1].

2.1.1.2. TRT Detector. Outermost element of the inner detector is Transition Radiation Tracker (TRT) [1]. It detects particles via drift tubes, which have 4 mm diameter and length of up to 144 cm, with a position resolution of 200μm. The tubes are filled

with Xenon gas which is ionized when a charged particle passes through it. The signal is produced as a pattern of hit straws. TRT is present in both the barrel and the end-cap regions. In the barrel section, each straw is divided in two with readouts at both ends and the detector covers from 56cm to 107cm radially. In end-cap section, readouts are at the outer radius and the detector covers from 64 cm to 103 cm radially. The space between straws is filled with foam in order to generate transition radiation during the damage of charged particles. Particles with a particular energy have higher speed when they are lighter and for ultra-relativistic charged particles, amount of the transition radiation is high. So lightest charged particles, like electrons, can produce the strongest signals. Hence TRT assists in electron identification in addition to providing tracking information.

2.1.2. Calorimeter

Calorimeters are placed outside the solenoidal magnet that surrounds the inner detector [1]. They are the combination of dense absorber material to fully absorb incident particle and active material to produce an output signal proportional to input energy. They are sampling calorimeters as while absorbing energy in dense material, periodically sample the shape of particle shower, inferring the emery of the original particle from these measurements. They cover the regions of $\mid \eta \mid = 4.9$. With the higher granularity than the other detectors, and wider range in η of calorimetry, electrons and photons can be measured more precisely. Two kinds of calorimeters are used in ATLAS detector which are electromagnetic calorimeter and hadronic calorimeter.

Electromagnetic (EM) calorimeter can measure the energy of particles that interact electromagnetically i.e. charged particles and photons. It has a high precision on energy and on position measurements (angle precision 0.025 radians). Calorimeter is constructed in shape of accordion, to provide complete ϕ symmetry. As an absorber lead and stainless steel, as a sensing element liquid argon is used. The showers in argon liberate electrons when a particle arrives and by collecting these electrons position and energy is determined. It consists of a barrel ($\mid \eta \mid < 1.5$), which is divided into two identical half barrels separated by a gap at $z = 0$, and two end en caps ($1.4 < \mid \eta \mid < 3.2$) of

two coaxial wheels. It is also segmented in ρ direction, to be optimized as a function of η to increase energy resolution performance.

Hadron calorimeter is used for measuring the energy of particles that pass EM calorimeter, primarily hadrons. It is less precise than EM calorimeter (0.1 radian angle precision). As a sensor tiles of scintillating plastics are used. The shower causes the plastics to emit light when a particle arrives and by detecting light that is produced, energy and position can be determined. It mainly has three parts; tile calorimeter which has a hadronic coverage in central rapidity ($|\ \eta\ | < 1.7$), LAr hadronic end-cap calorimeter, which consists of two independent wheels for each en cap ($1.5 < |\ \eta\ | < 3.2$) and LAr hadronic end-cap calorimeter.

2.1.3. Muon spectrometer

As muons can penetrate the other elements of the detector and leave only minimum ionization energy in the calorimeters, further data from muons are taken by the muon spectrometer [1]. It has a similar working principle with the inner detector i.e. measures the momentum with the curvature of path. It is placed around the calorimeter with a radius from 4.25 m to the full radius of detector 11 m. There is a magnetic field produced by superconducting toroidal magnets and detection elements are muon chambers. These chambers are made of metal tubes, and a central wire is placed inside to tube. They are filled with gas mixture of 93% Ar and 7% CO^2. When muon passes through the tube, it attracts ions and electrons drift to the side and the center. By measuring the time difference, position of the hit can be determined. Technologies used at muon spectrometer are Monitored Drift Tube (MDT), Resistive Plate Chambers (RPC), Cathode Strip Chambers (CPC) and Thin Gap Chambers (TGC). The mosrt of the coverage is MDT and RPC, while CSC and TGC is used at forward regions. The purpose of MDT and CPC is precision measurement and the others are used for triggering. With this construction, Muon spectrometer achieves 10% P_T resolution for muons of $P_T = 1\text{TeV}$.

2.1.4. Magnet System

ATLAS detector system has two large superconducting magnet systems [1]. Due to the Lorentz force, charged particles follow helical trajectories in magnetic fields with high momentum particles curving less than the low momentum ones. This is achieved in the inner detector by a solenoid magnet and in the muon spectrometer by a toroid magnet.

The solenoid magnet is placed around the inner detector and produces an axial 2 T magnetic field in the $+z$ direction. The axis of the solenoid coincides with the beam axis and the axial length is 5.3 m.

The toroidal magnetic field is produced by 8 very large air core superconducting barrel loops and 2 end-caps. It is located outside the calorimeter and within the muon system. The barrel loops are 26 m long and 20 m in diameter and store 1.6 joule energy. The field is not uniform, besides the magnet system is optimized to bend the particles in inner detector and muon spectrometer.

As all the coils are superconducting, they require cooling circuits, a vacuum system and cryostat for optimum thermal insulation. The temperature is kept at 4.8K with liquid helium.

2.2. ATLAS Trigger

The Large Hadron Collider accelerates particles (proton or heavy ion) close to the speed of light and accelerated particles collide at 8TeV energy with 10^{34} $\mathrm{cm}^{-2}\mathrm{s}^{-1}$ luminosity. Particles at this energy level collide as bunches at a rate of 40 MHz [4]. On each bunch crossing an average of 23 collisions, most of which are pile-up events, are generated. However since the ATLAS detector reads data from approximately 10^8 channels, the event sizes are large and the data storage rate is limited to about 200Hz. This means, a very fast trigger system is required to reduce the data rate by a factor of $\frac{40MHz}{200Hz} \simeq 10^5$. The trigger should be highly selective, but at the same time as inclusive

as possible not to miss new physics. It should also be adoptable and easily configurable to handle unexpected rates and changes in luminosity.

To achieve this, ATLAS uses a three level trigger system, which selects a wide variety of events like b-physics events, events with missing energy, high p_T leptons and jets efficiently [5]. These levels are level 1 (LVL1), level 2 (LVL2), and event filter (EF). LVL1 is hardware based, while the other two stages are software based. They all run online and 1 TB/s data is reduced to 300 MB/s, which is written to the mass storage. Trigger flow rate and frequencies are shown in Figure 2.4.

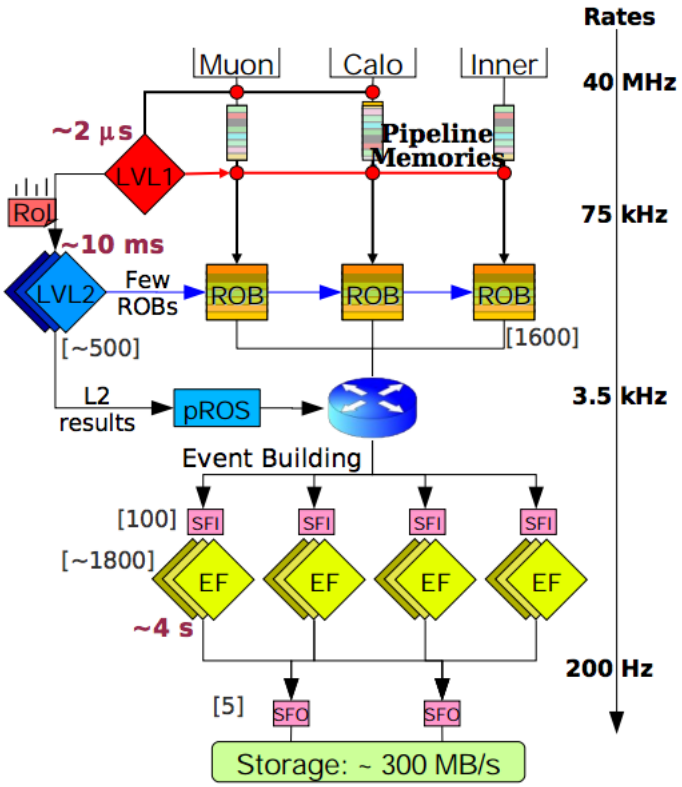

Figure 2.4. ATLAS Trigger architecture and data flow [4].

2.2.1. Level 1 Trigger

Level 1 Trigger is a hardware trigger system, which is placed close to the detector front-ends [4] [5]. It has a fast selection latency, less than 2 μs and high rejection rate, greater than 99.8%. It decreases the rate down to 75 kHz. While the selection algorithm runs full detector data is stored in pipeline memory. After a LVL1 accept; detector data is transferred to buffers and central processors create Regions of interest

(RoI) data to guide the LVL2. LVL1 trigger elements LVL1 looks for jet, muon etc. candidates and specify a region of interest for seed of LVL2, Figure 2.5. To achieve this, data from calo towers, RPC and TGC are used, so it is not running on full detector.

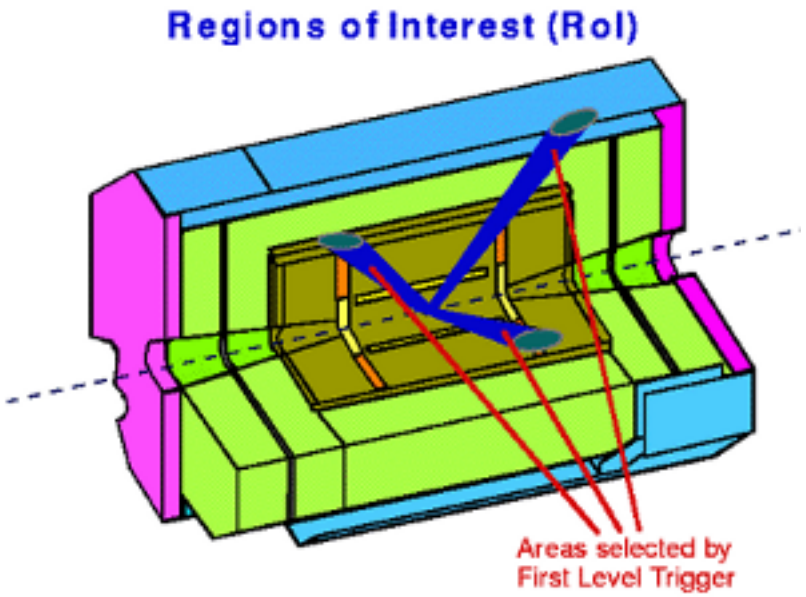

Figure 2.5. Example of region of interest (RoI) [5].

2.2.2. Level 2 Trigger

Level 2 trigger system is a software based online trigger, which runs on a 500 processor farm of about 1800 CPU cores [4, 5]. It accepts LVL1 selected events and reduces the rate down to approximately 3.5kHz, i.e. by a factor of 20-30, and has a latency about 40 ms . It has full granularity different by the LVL1, which means it uses entire detector data including inner detector. LVL2 is the first step where the silicon detector data becomes available. It reduces the data from LVL1 by a factor of 100. For high p_T, LVL2 uses limited detector regions, RoI, identified from LVL1. Trigger elements can be combined to form trigger objects at this level, for example from a thin jet and tracks, taus can be identified. It is also easy to configure. Tresholds can be modified to control rates. Specialized B-physics trigger lines should be achieved at LVL2, so fast tracking algorithms applied to strips (SCT) and pixel are necessary. These tracking algorithms will be discussed in detail in Chapter 3.

2.2.3. Event Filter

Event Filter(EF) is the last stage of the online event selection [4,5]. It select what will be written to mass storage for later offline analysis. It accepts data from LVL2 and reduces the event rate by another factor of 20 with a latency of 4sec. Finally, data is written at a rate of 300 MB/s. EF confirms LVL2 and has the possibility of using LVL2 selections as seeds. It selects the events according to given physics menu list i.e entries for discovery physics, standard physics and calibration events. It also classifies accepted events. It prepares data for monitoring some events for immediate analysis. Unlike the LVL2, which runs specialized algorithms for trigger, most of the EF code is based on the offline software, with minor modifications and some optimizations to allow proper execution in regions of interest.

3. TRACKING

3.1. ATLAS Level 2 Trigger Tracking Algorithms

Charged particles produced after the collision move in helical trajectories in a magnetic field [6]. So by measuring the behaviour, i.e curvature and the direction of the trajectories of the particles, their energy and momentum can be found.

Pattern recognition in the ATLAS detector is not an easy task as can be seen in Figure 3.1. We have Higgs boson decaying into four muons (so called the golden channel for Higgs boson searches), which is recorded by ATLAS at the LHC design luminosity. At LVL2 trigger, we have the problem of having tight time constrain as 10^3 particles emerge from the collision point at a rate of about 75kHz within range $\mid \eta \mid < 2.5$, which results in high track density in the detector.

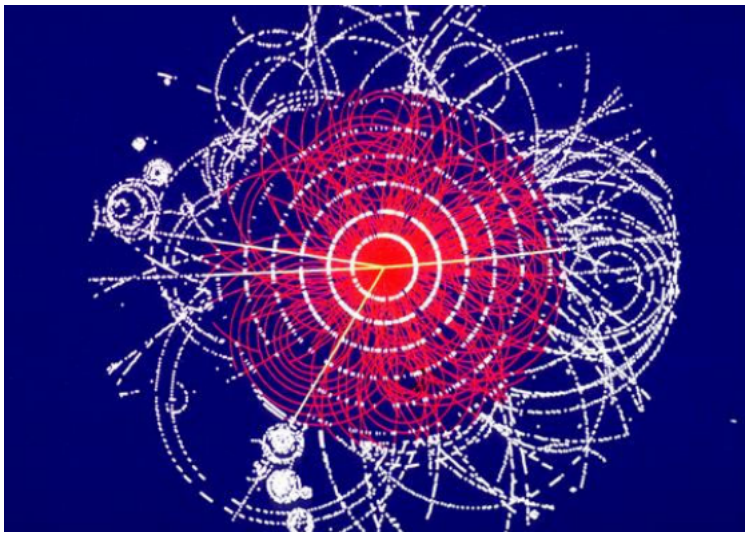

Figure 3.1. A simulated $H \rightarrow ZZ \rightarrow 4\mu$ event with the associated large track density foreseen at high luminosity [1].

Taking the data from LVL1 trigger, if the object of interest is lepton or photon, a refinement of the RoI is done using dedicated algorithms. After that, LVL2 track reconstruction takes place. At this stage, particle identification, momentum and energy measurement can be done from the track identification. Inner detector is used for track reconstruction but calorimeters and the muon spectrometer are also helpful for particle

identification.

At LVL2 trigger of ATLAS, three kinds of tracking algorithms are used; SiTrack, IDScan and TRT Segment Finder [6]. TRT Segment Finder does tracking in the TRT and is not actively used for physics purposes. SiTrack and IDScan take space points from pixel and SCT layers and form cluster of points for each track. They have different pattern recognition approaches but they utilize same tools for track fitting and extrapolation to TRT detector. Their main differences are;

SiTrack is based on a combinatorial approach. Pairs of space points in the inner layers are searched and then combined with the points on outer layers to form triplets, if they satisfy beam line conditions. For SiTrack, the knowledge of the position of the point of interaction is not mandatory but it can be used to reduce combinations in RoIs with high track density or in full scan mode, i.e. when running with data from the entire detector instead of RoIs.

IDScan is based on a projective approach. For IDScan, the position of the point of interaction, i.e. primary vertex along the beam line, is necessary for reconstruction. It is the reference point used for histograming. IDScan then identifies clusters of hits in the $\eta - \phi$ space.

Depending on the selection criteria and physics case the better of these two algorithms can be chosen and used. Also depending on on-line operations in stable beam working conditions and in different luminosity scenarios, one of the two algorithms can be preferred over the other.

3.1.1. SiTrack Algorithm

SiTrack is a tracking algorithm that reconstructs trajectories from the space points of inner detector [6].

The space points are first grouped into logical layers, which are predefined lists of

physical layers and labelled with increasing number moving away from the interaction region. Definition of logical layer varies with respect to different detector conditions or depending on a specific trigger selection. First logical layer, for example, includes the two innermost pixel layers in the barrel and two innermost pixel end-cap disks. At the beginning of the pattern recognition, space points from the two innermost logical layers form a combinatorial pair, called seeds. For each seed, using a straight line approximation, the extrapolation to the beam line is evaluated. Linear extrapolation is used, as charged particles moving in a solenoidal magnetic field form a straight line in the longitudinal plane. For the seeds that point the interaction region, selection is done according to pre-set criteria; minimum momentum and maximum impact parameter. Pairs that satisfy these conditions fill a one dimensional histogram for the z position of the primary vertex and by searching for the maximum, the primary vertex of interaction is found.

As the next step, pairs that do not pass through primary vertex are optionally rejected. Each seed is then extrapolated to the outer logical layers to form triplets. These triplets are then fitted to straight lines in the longitudinal plane and parametrized as a circle in the transverse plane, i.e. they are crudely fitted to helix equations. A single full track is reconstructed by grouping the triplets having similar track parameters. Finally a full proper track fit is performed to obtain the final tracks.

3.1.2. IDScan Algorithms

IDScan also uses silicon space points like SiTrack for a fast pattern recognition [6]. The main difference between them is that, in IDScan first of all the z position of the primary vertex along the beam line is determined. Then the space points are filtered based on their location on detectors and track parameters.

At the design luminosity of LHC, an ATLAS event has tens RoIs which contains total of thousands of SCT and pixel points on average. Combinatorics are needed to be significantly reduced, to shorten execution the time. There are two principles to achive this;

- The z position of the primary interaction vertex has a significant spread in z position ($\sigma_z \approx 6$cm), but if the position can be identified, many combinatorics can be easily eliminated, as pile-up vertices will also tend to be separated from the primary vertex.
- Physics collisions have higher transverse momentum than pile-up collisions on average.

Using the principles IDScan reconstructs tracks in 4 separate steps; z-finder, hit filter, group cleaner and track fitter.

(i) Z-finder

The aim of the z-finder is to determine to determine the z position of the interaction point which will then be used in the hit filter. RoI data from LVL1 is used. The RoI is divided into a large number of ϕ bins. In each bin, the z position is calculated from all pairs of space points by doing a extrapolation to the beam line. To reduce binning effects i.e. not to lose the tracks at the ϕ the boundaries, pair of hits from adjacent ϕ bins is are also allowed. By accepting the pairs which pass through the interaction region, a one dimensional histogram of the z position is built and by searching for the maximum of the z position is found. The width of the bins determines the minimum transverse momentum, so to accept high momentum particles, the width is set to approximately $0.2^\circ - 0.3^\circ$.

(ii) Hit filter

The aim of the hit filter is to select hits from high momentum tracks. The idea is that the high P_T tracks are contained in a small solid angle in (η, ϕ) i.e. do not vary much in η (pseudo rapidity) and ϕ as they do not bend much. So a two dimensional histogram of (η, ϕ) is filled by number of detector layers containing hits for each bin. To reduce the binning effects, neighbouring bins are grouped together, and if there are 5 hits in a group, the hits at that group are accepted. Around 95 percent of space points are rejected at this step.

(iii) Group cleaner

At group cleaner remaining noise is removed and tracks are reconstructed. At this step any triplets of hits in each group are taken and track parameters are calculated. Cuts, like minimum P_T, are then applied to each triplet. A two dimensional histogram of ϕ_0 and $\frac{1}{p_T}$ is then filled from the parameters of triplets that pass the cuts. If the total number of unique hits from all the triplets that contribute to a given (ϕ_0, P_T) bin is at least 4, that bin is considered to contain one good track candidate.

(iv) Track fitter

At track fitter, good track candidates, which are taken from the group cleaner, are passed to a standard Kalman-type fitting package [8].

3.2. Helix Equations

ATLAS inner detector system has a magnetic field of 2T [10]. Charged particles moving in this magnetic field are bent, and leave tracks in form of helices. We use helix equations in cylindrical coordinates to fit the tracks. Beam protons move along the z axis and collide at $z = z_v$. As the collision will be on the beam axis, $x = y = 0$ for the interaction point. We use the usual cylindrical coordinates where;

$$\phi = \arctan(y/x); \quad \rho = \sqrt{x^2 + y^2}; \quad r = \sqrt{x^2 + y^2 + z^2}; \quad \lambda = \arctan(\frac{z - z_v}{\rho}) \quad (3.1)$$

The projection of a helix on the $x - y$ plane is a circle. From the Figure 3.2 and by using Equations 3.1, following equations can easily be written;

$$x_c = -R_0 \sin(\phi_0) \qquad x = x_c + R_0 \sin(\phi_0 + \Delta\alpha) \quad (3.2)$$

$$y_c = R_0 \cos(\phi_0) \qquad y = y_c - R_0 \cos(\phi_0 + \Delta\alpha). \quad (3.3)$$

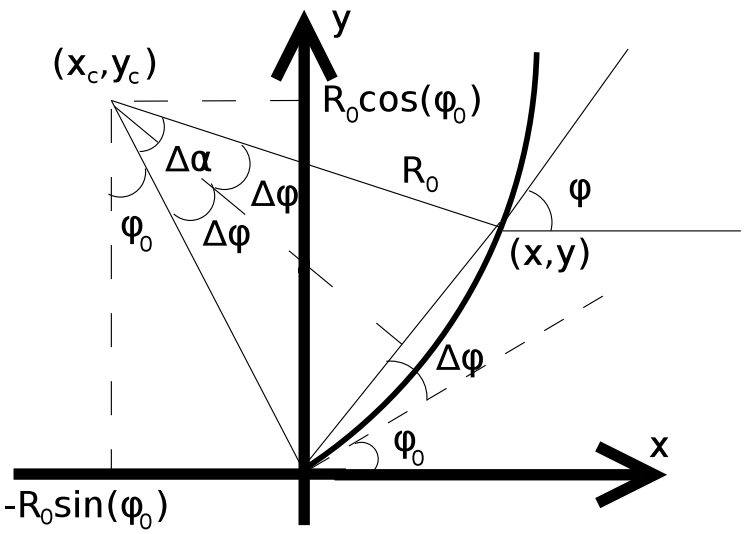

Figure 3.2. $x-y$ projection of helix in cylindrical coordinates.

As there is a magnetic field along the z axis in ATLAS detector, the z coordinate of the helix is proportional to $\tan(\lambda_0)$ and R_0 . So;

$$\Delta z = z - z_v = \tan(\lambda_0) R_0 \Delta\alpha \tag{3.4}$$

As we are interested in high momentum tracks ϕ varies slightly so we can approximate:

$$\Delta\phi = \phi - \phi_0 \simeq \sin(\Delta\phi) = \frac{\rho}{2R_0} \tag{3.5}$$

which can be seen from Figure 3.2.

We can also express $\Delta\phi$ as;

$$\Delta\phi = \frac{z - z_v}{2\tan(\lambda_0) R_0} \tag{3.6}$$

by using Equation 3.4

If we combine Equation 3.5 and Equation 3.6, we can conclude that;

$$\rho = \frac{z - z_v}{\tan(\lambda_0)} \tag{3.7}$$

We can take λ_0 constant along the helix, as we are interested in only high momentum tracks. So from Equation 3.1 z_v can be extracted from any two points on the helix as;

$$z_v = z_1 - \rho_1 \frac{z_2 - z_1}{\rho_2 - \rho_1} \tag{3.8}$$

3.3. Z-Finder Algorithm

In this section, we define an algorithm similar to the z-finder described briefly in Section 3.1.2 [11]. Z-finder is an algorithm which aims to find the z position of the primary vertex by extrapolating helix equations to space points of pixel and SCT detector. The algorithm can be applied to full detector data or data from any given RoI. While it is possible to extend the algorithm to work in the end-caps. Here we describe a version that runs on only the space points from the pixel and SCT barrel section. For the primary vertex, as the interaction is assumed to happen on the beam line, $x_v = 0$ and $y_v = 0$ is taken. Here, the question is; what is z_v? To determine z_v, high momentum tracks are considered. To parametrize tracks, with the notation introduced in Section 3.2.

As we are interested in only high momentum tracks, assumption in Equation 3.5 can be made. Hence we note that; this algorithm would not work properly for low momentum tracks, i.e. there is an inherent minimum momentum requirement, which is acceptable and even desired as this separates the interesting physics objects from noise and pile-up.

The core idea is encapsulated in Equation 3.6, which is reproduced here:

$$\rho = \frac{z - z_v}{\tan(\lambda_0)} \tag{3.9}$$

What this algorithm does is that; it histograms the z positions of vertex from pairs. To reduce the influence of multiple scattering, two hits from pixel detector, i.e. innermost three layers, are selected. These two hits are used to calculate the z position from the equation.

$$z_0 = z_1 - \rho_1 \frac{z_2 - z_1}{\rho_2 - \rho_1} \tag{3.10}$$

where (z_1, ρ_1) and (z_2, ρ_2) are the coordinates of ay two hits. These points are selected from different pixel layers, and first they are checked, if they pass through the interaction region where;

$$-15\text{cm} < z_0 < 15\text{cm} \tag{3.11}$$

From Equation 3.5. This helix has the radius;

$$R_0 = 0.5 \frac{\rho_2 - \rho_1}{\phi_2 - \phi_1} \tag{3.12}$$

Only the high momentum tracks are interesting, so to eliminate pile up events, a momentum cut, P_{cut}, is set. We want to select only the pairs of space points, which could make a helix segment satisfying $P_T \geq P_{cut}$, in other words; $R_0 \geq R_{cut}$ and this leads us to the cut on ϕ as, $\Delta\phi_{cut}$. $\Delta\phi_{cut}$ is a function of layers i and j, from where the points are selected. It can be written as:

$$\Delta\phi_{cut} \leq \Delta\phi_{cut}(i, j) = \frac{\rho_i - \rho_j}{2R_{cut}} \tag{3.13}$$

$P_{cut} = 5\text{GeV}$ is set and using the radii of the pixel detector layers;

- $\Delta\phi_{cut}(0,1) = 0.00419$
- $\Delta\phi_{cut}(0,2) = 0.00611$
- $\Delta\phi_{cut}(1,2) = 0.00192$

in radians for the layers of the pixel detector. One can also find vertices of pile up events by changing P_{cut}, but it should be still large enough to avoid multiple scattering, so that the linear equations still hold for the layers of the pixel detector.

For the region, that the algorithm is defined, i.e. barrel region, we can also define an acceptance cut λ_{cut}, which comes from the definition of barrel;

$$|\lambda| = |\arctan(\frac{z_2 - z_1}{\rho_2 - \rho_1})| \leq 49^\circ.$$

New algorithms for the end-cap and the transition region can be constructed for further studies.

The algorithm, first loops over the first or second innermost layers of the pixel detector, and then by looping over an outer layer of pixel detector, pairs are combined and checked if they satisfy the conditions described above. If they pass P_{cut}, the histogram of z position is filled by extrapolating to the beam line. This is the simplest method that one can follow. As seen in the results at the end of this chapter, basic z-finder works fine if there is no noise, but it does not perform well enough for high luminosity events.

For high luminosity events, containing pile up events and noise, SCT space points should be used. After 2 points are selected from two nested loops over pixel layers, it is checked whether there hits on the SCT layer that can further extrapolate to helix.

At two dimensional histogram of $\phi - \eta$ of SCT space points is constructed first. If a pair can pass the conditions above, it is then extrapolated to SCT, and checked if

there are points on SCT bins (or on the nearmost to eliminate binning errors).

- If there are space points on 2 layers of the SCT, the track is given a weight of 1,
- If there are space points on 3 layers of the SCT, it is given a weight of 2,
- If there are space points on 4 layers of the SCT (on all layers), it is given a weight of 3.

The code of algorithm implemented in C can be found in Appendix A.

To check the algorithm three sets of space points from the barrel region of inner detector, which are taken from the same full detector data of a particular event, are used:

(i) 442 pixel + 416 SCT space points from triggering interaction,
(ii) 4685 pixel + 3891 SCT space points from triggering interaction and pile up events,
(iii) 13285 pixel + 9509 SCT space points from triggering interaction and pile up events including noise.

If 2 nested loops are used by using only the space points of the pixel detector, the resulting histograms of the z position can be seen in Figure 3.3a, 3.3b, 3.3c with $P_{cut} = 5\text{GeV/c}$ and bin size of 0.05cm. As seen in the figure, the most basic algorithm works fine with the data sets (i) and (ii) quite well, but for the data set (iii), which includes pile up and noise, it is difficult to determine the z position of primary vertex. However, by applying 3 nested loops, with taking into account SCT space points also, z position of interaction point can easily be identified for all data sets, as seen in Figures 3.3d, 3.3e, 3.3f.

As mentioned before, the same algorithm can also be used to find the secondary vertices, that come from pile up interactions, by setting $P_{cut} = 0.5\text{GeV/c}$ or lower. The result on the data set (iii) can be seen in Figure 3.4. As seen in Figure 3.4a, even for the high luminosity event, z position of secondary vertices can be found from the peaks of histograms easily. And if we compare the results with actual locations of the

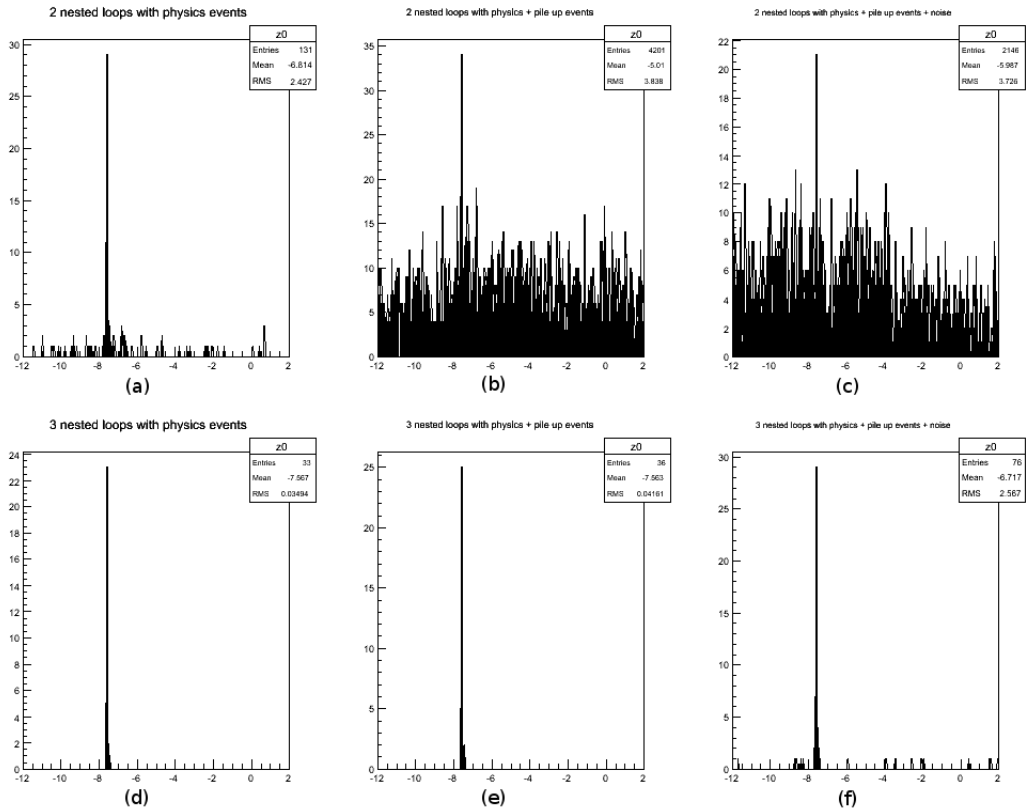

Figure 3.3. Histograms of z position of the primary vertex, after applying 2 or 3 nested loops, on 3 different data sets.

vertices of the simulated tracks and reconstructed tracks, which are in Figures 3.4b, 3.4c, one observe that all of the secondary vertices could be determined successfully.

The z-finder algorithm can successfully determine z position of the vertices. It runs not only on full detector data but also on RoI, as the performances can be seen seen Section 4.1. Reducing the combinatorics makes the algorithm faster.

For example if we consider the full detector data, as seen in Figure 3.5, and assume that there are a total of N points, and each of 3 points out of N points are combined, then the resulting number of combinations "n" is found to be;

$$n = \frac{N(N-1)(N-2)}{3!} \simeq \frac{N^3}{6}.$$

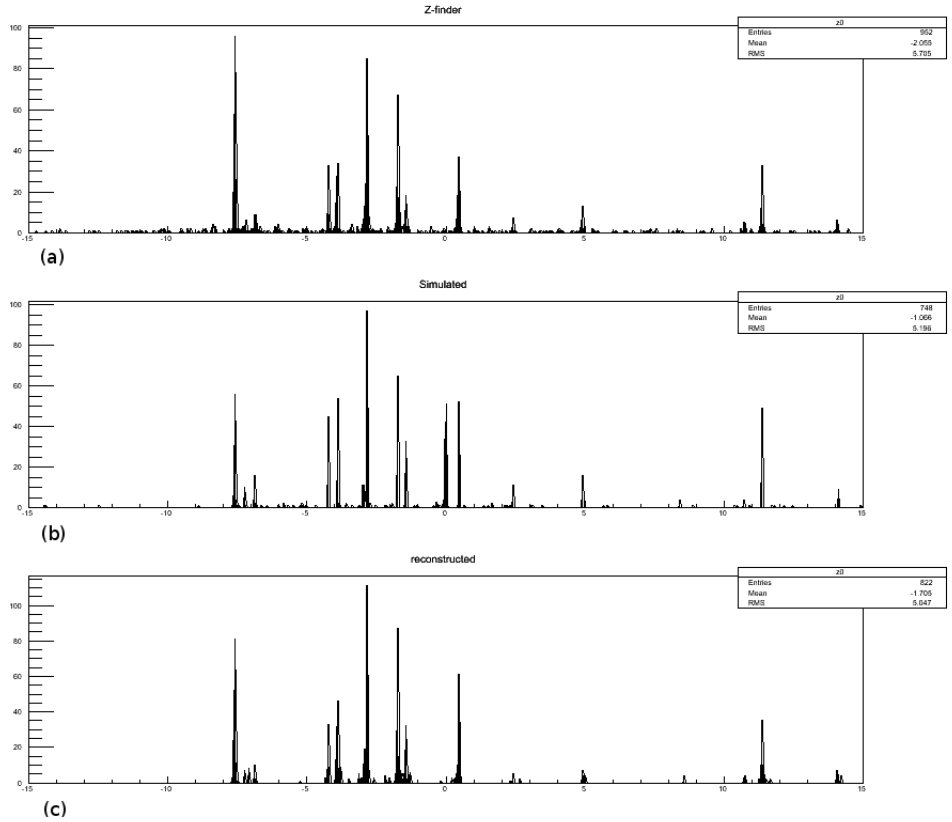

Figure 3.4. Histograms of z position of the pile-up events, after applying 3 nested loops.

But we can divide the data into M exclusive sets, so combinations are reduced to:

$$n \simeq \frac{M(\frac{N}{M})^3}{6} = \frac{N^3}{6M^2}.$$

For instance, when $M = 30$, the number of combinations are reduced by a factor of approximately 10^3.

In brief, the algorithm loops over all the hits in the barrel part of innermost pixel layers and to combine pairs, instead of looping over all higher pixel layer, just the hits on ϕ range, defined by the P_{cut} of higher pixel layer are looped. Furthermore, for the SCT hits, just those layers with at least one point when extrapolated $\phi - \eta$ bin are counted, by first constructing matrix for SCT layers. And lastly, by calculating z positions from pixel pairs and weighting them depending on the extrapolation to SCT, a z histogram is filled and the z position of the primary vertex is determined. Hence,

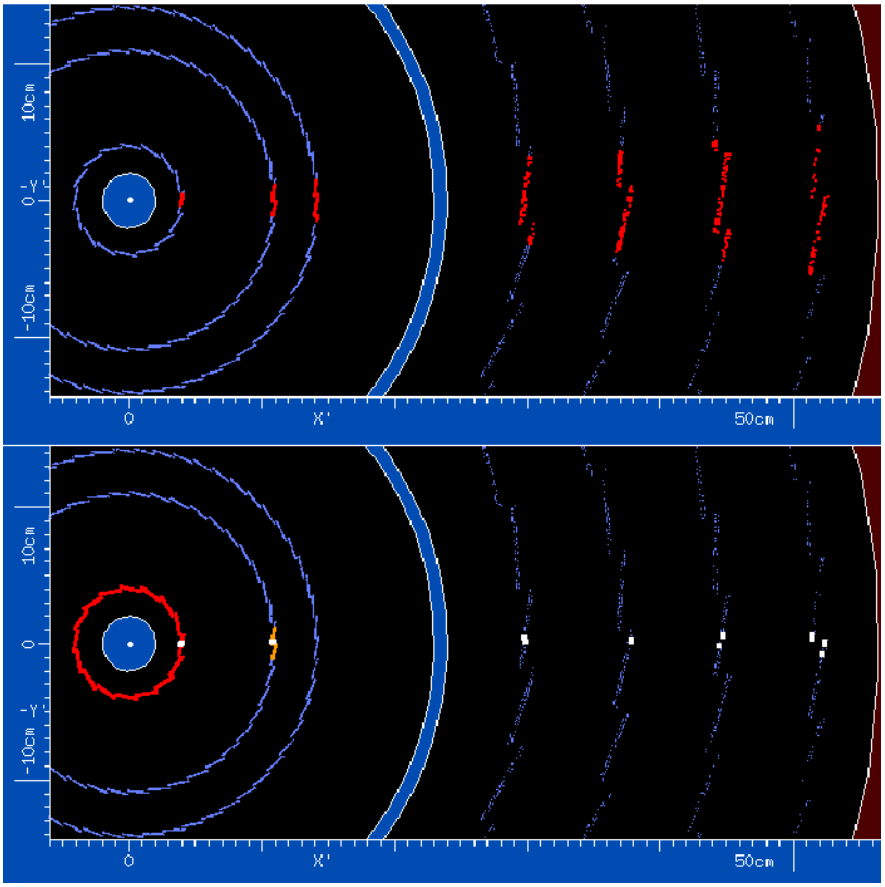

Figure 3.5. Inner detector hits divided into bins [11].

a fast algorithm to find primary vertex of interaction has been realized.

3.4. V-Plot

V-plot is a two dimensional representation of tracking data for HEP experiments [12]. In this thesis, the results of the filtering algorithm will be represented using V-plot.

While, it is easy for computers to connect and fit points to lines or curves, it is sometimes difficult to connect and visualize them by eye. V-plot helps us to visualize tracks, and extract physics from the tracks by just glacing of them. V-plot depends on 2 basic ideas:

(i) If you would like to meaningfully represent a three dimensional point in 2D, a

line, i.e two points, is necessary to extract coordinates. For example, as seen in Figure 3.6, if you just put one point, it is completely meaningless. But instead of a point, if one would draw a line from the point to the plane, all 3 coordinates of this point can easily be read, and furthermore, it can be visualized.

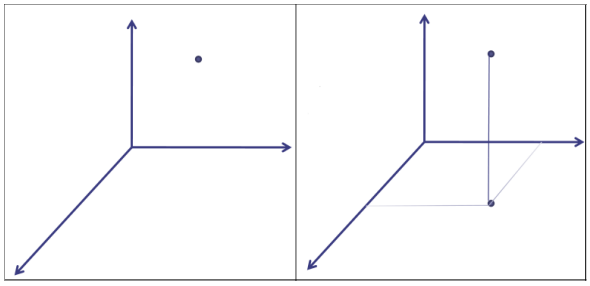

Figure 3.6. A three dimesional point is represented with a line in 2D.

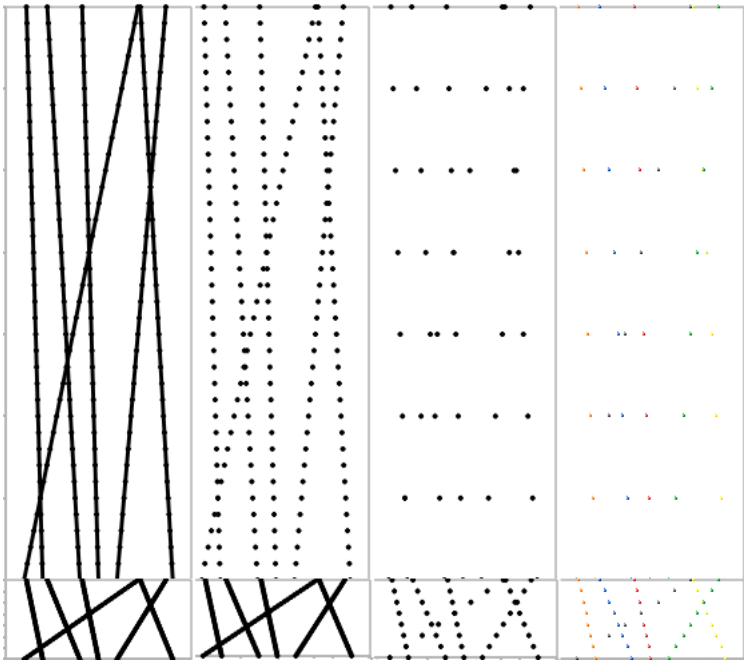

Figure 3.7. Compression of points from diffirent kinds of detectors.

(ii) The second basic idea used by v-plot is compression. If we were visualising data from detectors like calorimeters (Figure 3.7a) or bubble chambers (Figure 3.7b), compression would not be needed to visualize tracks. But tracks taken from ATLAS inner detector is most likely in Figure 3.7c. So even when the hits from each track are coloured as seen in Figure 3.7d, it is still difficult to find some tracks by eye. So what v-plot does is, to compress the hits as shown in the bottom part of Figure 3.7, so that human visual pattern recognition can read it.

Compression also allows us to recognise kinks and to identify tracks in noisy environments, if the compression is perpendicular to track. For example in Figure 3.8a, there are three visualization of a given tracks, a continuous line, a set of distinct points, ans a set of close points. When they are not compressed, even for the continuous line representation, it is difficult for human eye to identify where the kink is. But when they are compressed, even for the distinct points, one can easily identify where the kink is. Also, if we put the tracks in Figure 3.8b in a noisy environment as in Figure 3.8c, without compression none of the two tracks can be identified. However, with the compression shown below, the track that is vertical along the compression can be identified. To identify the other track, before the compression, a rotation s0 that the track becomes vertical, can be made.

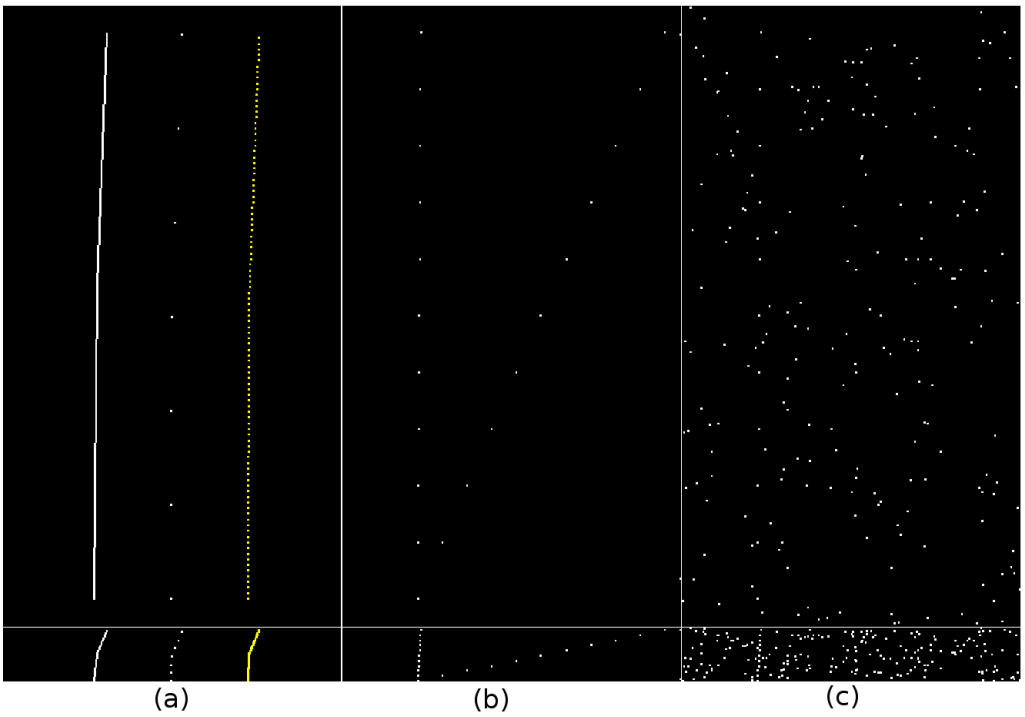

Figure 3.8. (a)Compression to identify kinks (b)Compression for vertical and oblique tracks (c)Compression in a noisy enviroment [12].

ATLAS detector is a noisy environment and we are trying to identify tracks here. As seen in Figure 3.9a, high momentum tracks of triggering interaction (no noise and pile up) coming from the interaction point are like lines on the $x-y$ plane. To compress them, projection to the $\phi - \rho$ plane is taken in Figure 3.9b. Distinct tracks can be identified here but close tracks are difficult to identify. And when the compression is

done along the ρ axis, as in Figure 3.9c, all the tracks can be identified clearly.

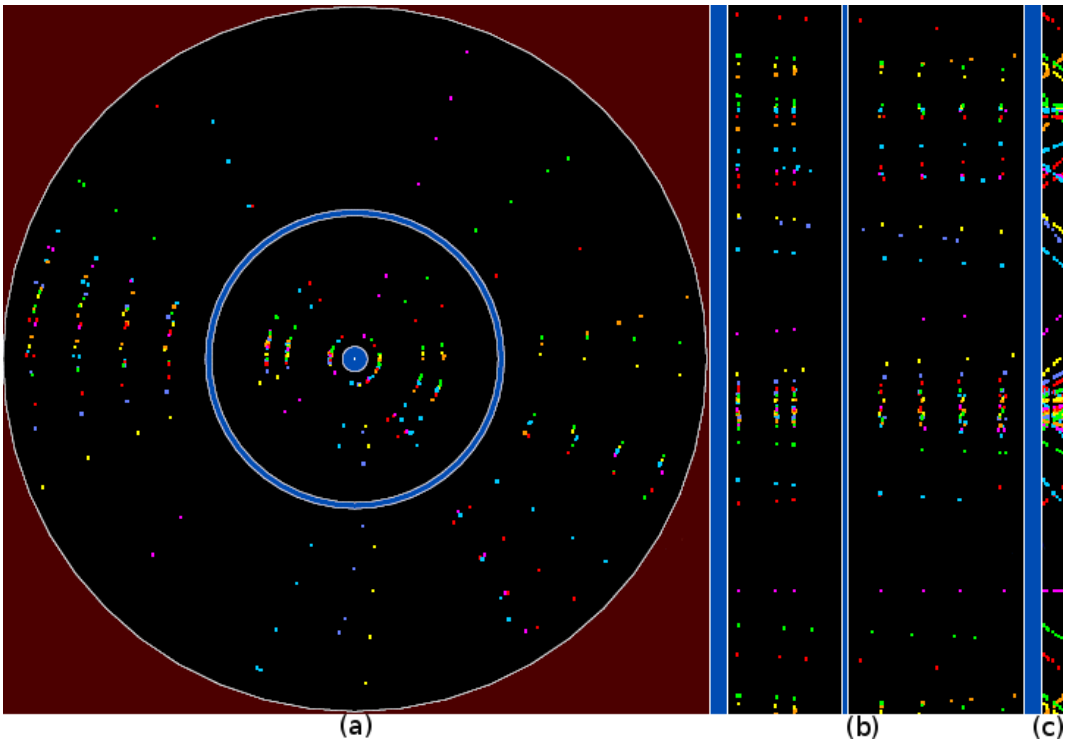

Figure 3.9. ATLAS inner detector points (a) on $x-y$ plane (b) projection on $\phi-\rho$ plane (c)compression of $\phi-\rho$ projection [12].

V-plot can be thought as a special form of $\phi-\eta$ projection. It is used in ATLAS event display program; ATLANTIS [7]. The V in the name comes from the resulting shapes of the particle tracks.

To construct v-plot, let us start with two helices, coming from the primary vertex, which can be seen in Figure 3.10a in $y-x$ coordinates,in Figure 3.10b in $\phi-\rho$ coordinates,in Figure 3.10e in $\rho-z$ coordinates. As the B-field is along the z direction, they are linear in the $\rho-z$ projection and slightly curved in the $y-x$ projection. In $y-x$ coordinates there is a limit for ρ coming from the last layer of SCT detector, which is denoted by ρ'', as in $\rho-z$ and $\phi-\rho$ plane. From these projections, if the projection on $\phi-\eta$ plane is taken as in Figure 3.10f, two tracks are completely seperated. Here η is defined as;

$$\eta = \ln(\tan(\lambda) + \sqrt{1+\tan^2(\lambda)}). \tag{3.14}$$

By using the second idea, which is compression, the tracks can now be distinguished. However, we have still no idea about the charge and the transverse momentum of particles. At this point, first basic idea of representing a point with a line in 2D helps us. The Figure 3.10b shows that for one track, ϕ decreases with increasing ρ and for the other one vice versa. This means that, while one particle bends clockwise the other one bends anticlockwise. And how much they bend can be seen easily from this figure, which gives us an idea about the momentum of the particle. So the aim is now to combine information from 3.10f and 3.10b. At this point, instead of using η, a new variable η' is defined such that;

$$\eta'_{-} = \eta - k(\rho'' - \rho) \tag{3.15}$$

$$\eta'_{+} = \eta + k(\rho'' - \rho) \tag{3.16}$$

where k is just a parameter with dimension 1/length, which can be selected interactively to set the width of V depending on the scale. In this thesis, k is selected such that, a 45° angle is produced between the arms of a V corresponding to a track with momentum 1 GeV/c.

Now by using Equation 3.15 and 3.16, Figure 3.10c and 3.10d are obtained respectively. Combining these two projections, we end up with the Figure 3.10g, which represents the different tracks with different V's. The mathematical proof, (only) high momentum tracks coming from the primary vertex can have "V" shapes under these transformations, can be done by starting from the helix equations, see appendix C

The question at this point is, what the shape of a V tells us. It is obvious from Figure 3.10b,c,d that, the vertex of the "V" comes from the hit in the outermost layer of the SCT detector. And if the track is bending clockwise, i.e a negatively charged particle in the magnetic field, an up-right V is drawn; and if the particle is positively charged, an inverse V shape is reached on the $\phi - \eta'$ plane. One can also see this

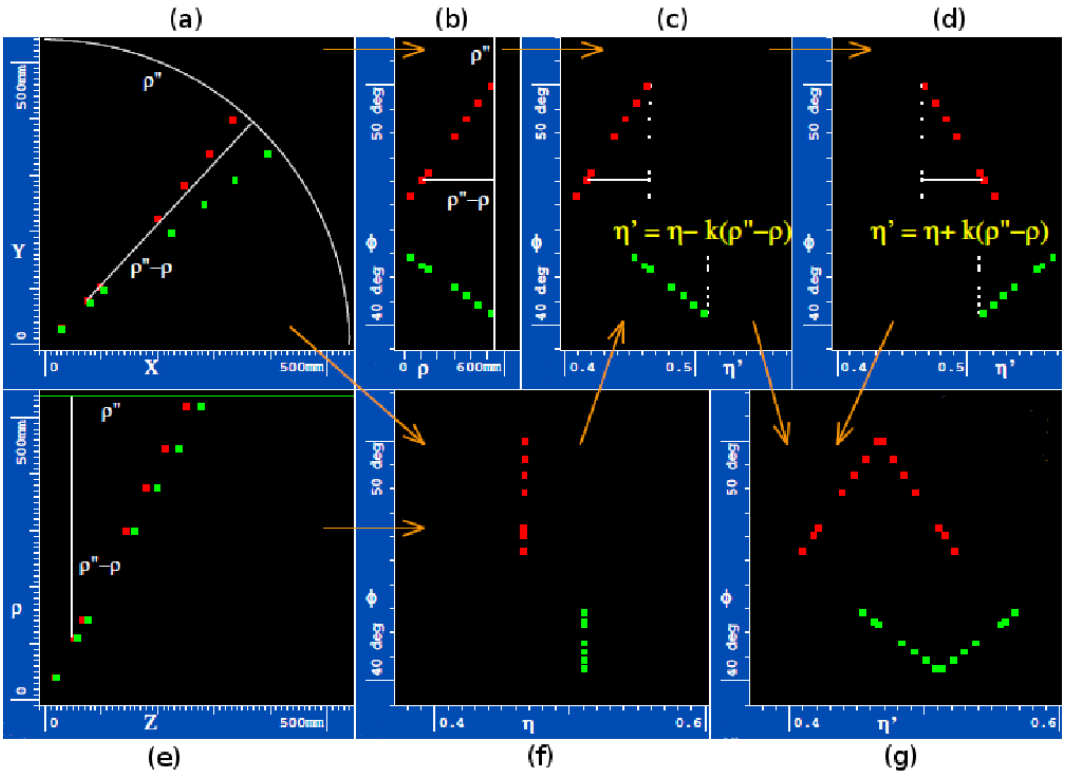

Figure 3.10. Construction of V-plot by using 2 high momentum tracks coming from the primary vertex [12].

from Equations C.2 and C.1 that. As R_0 changes sign with the charge of particle, the direction of "V" changes with the sign of the particle. Furthermore, examining Equations C.1 and C.2 deeply, R_0 is proportional to the transverse momentum of the particle P_T. As a result;

$$\frac{d\phi}{d\eta'_+} \propto \frac{1}{P_T} \tag{3.17}$$

i.e the slope of the arms are inversely proportional to the momentum of the track. If there is a high momentum track, arms will be wide apart, and for low momentum tracks, arms will be closer. For very high momentum tracks, arms will be so wide that, the shape will almost become a horizontal line. Examples can be seen in Figure 3.11.

For low momentum tracks, and for the tracks not coming from the primary vertex, one will not get the shape V under $\phi - \eta'$ projections, as the assumptions in the proof (see Appendix C) not hold. There will be deformations in the arms or the shape will

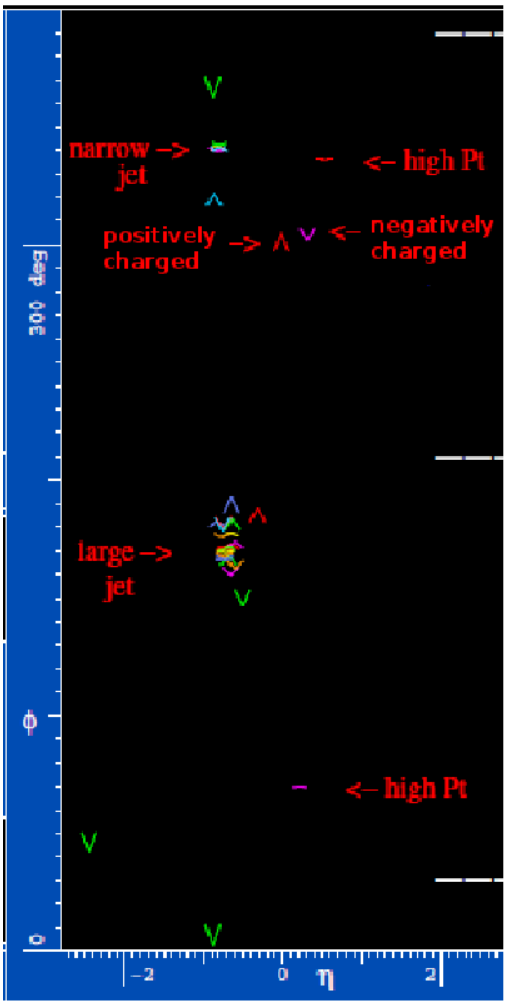

Figure 3.11. Examples of representing some tracks via V-plot [12].

not be symmetric as the slope will not be constant or not equal for the derivative of η'_+ and η'_-.

However, interesting events, i.e. physics events, have high momentum particles and these particles come from the primary vertex. So now in 2D, with the help of V-plot representation, one can not only identify the high momentum tracks, but also can have an idea for the charge and the momentum of each particle. Besides, from this picture, by back projecting, from the 2D image of V-plot, 3D coordinates of all hits can be extracted.

3.5. Hit Filter Algorithm

In this section a filtering algorithm to find tracks using the ATLAS inner detector space points is described [13]. As metioned in Section 2.2, ATLAS inner detector is a noisy, high luminosity environment and the filtering algorithm aims to find interesting space points from a huge set of data consisting of pile up events and noise. Interesting space points are sometimes 1% of the total space points and contains the hits of high momentum tracks ($P_T \geq 1\text{GeV}$), coming from the primary vertex. Hit Filter should be efficient enough not to lose any of the space points, coming from the physics events, and at the same time should clean the noisy and pile up events as much as possible. There are also time constraints that, it should be fast enough to be used at LVL2 trigger.

Hit filter uses the inner detector pixel and SCT space points, and unlike the z-finder code. It can be applied to both the barrel and the end-cap parts of the detector. The position of the primary vertex should be known to apply filtering, and the results will be expressed using V-plot.

With the z-finder algorithm, the position of the primary vertex is assumed to lie on the beam line ($x_v = 0$, $y_v = 0$), and the position z_v is determined. Using this position, the coordinate system is shifted along the z-axis, such that the origin is at the interaction point.

In Section 3.2, it is explained that for a helix, coming from the origin, $\lambda = \lambda_0$ is constant, and ϕ varies depending on the transverse momentum from Equation 3.5, where R_0 depends on transverse momentum, P_T only. But as only high momentum tracks are of interest, variation in ϕ is negligible. So the basic idea is, to group hits on a $\phi - \lambda$ matrix like in V-plot. But this time, not only the cell that contains enough hits but also the neighbours are grouped so as to avoid losses caused by measurement errors and from the change of ϕ as the track bends. Since we are not only interested the cell that contains a hit, but also in the neighbours, assuming that ϕ is in range $[0, 2\pi[$, a rap around is done for minimum and maximum of ϕ bins. But for λ, this is

not necessary as λ range of matrix is selected one bin larger than the detector system.

Three loops are used in the hit filter, and some manipulations are done on these loops to increase performance and to add grouping feature to hit filter. Instead of looping over cells of the $\phi - \lambda$ matrix, hits are looped at each step. This provides the algorithm a faster execution time, as the number of hits is much smaller than the number cells even for high luminosity events. The loops are as follows:

(i) Cell filling loop: In this first loop the matrix of $\phi - \lambda$ is filled.
(ii) Flagging Loop: Cells and hits are flagged as good or bad.
(iii) Resetting Loop: Resetting the matrix by looping over the space points.

At cell filling and flagging loops, to each cell two numbers are assigned, which have the meanings:

I_1 = number of entries.
I_2 = flag $\rightarrow$ $I_2 = -1$ untreated
$I_2 = 0$ bad cell.
$I_2 = 1$ good cell.

At the beginning $I_1 = 0$ and $I_2 = -1$. And at cell filling loop I_1 is filled.

There is also another number N_2 for space points, and if $N_2 = 0$, it is flagged as a bad space point, if $N_2 = 1$, it is flagged as a good space point. Here, "good" means, space points coming from the physics events and they form clusters on $\phi - \lambda$ plane. Let us assume, there is a hit with flag N_2, and it falls on a cell with I_1 and I_2. Flagging loop works as:

- If $I_2 = -1$, i.e. the cell is untreated, and its neighbours sums upto I_{sum};
  - if $I_{sum} \geq N_E$, where N_E is the minimum number of hits that a cell should have to pass filter, than the cell and the hit are flagged as good i.e. $N_2 = 1$

and $I_2 = 1$

- if $I_{sum} \leq N_E$, than the cell and the hit are flagged as bad i.e. $N_2 = 0$ and $I_2 = 0$

- If $I_2 \neq -1$, i.e. the cell is treated, it means that calculations and the decision is done for that cell and there is no reason to redo it. So the hit gets the property of the cell i.e. $N_2 = I_2$ is set.

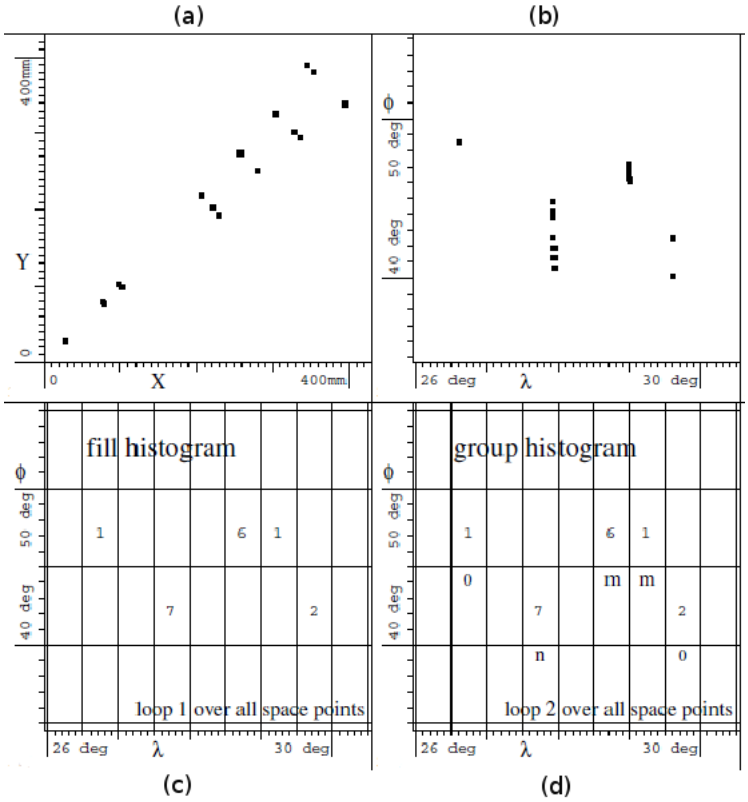

Figure 3.12. Principle of cleaning and grouping [13].

An example, how the algorithm works, can be found in Figure 3.12. Hits on $x-y$ plane can be seen in Figure 3.12a, and at first the projection on $\phi - \lambda$ plane is taken. Then the histogram is filled as in Figure 3.12. Setting $N_E = 5$, the clusters of 6-1 and 7 are selected and labelled as good and others are rejected. Then the matrix is resetted. It can easily be seen that, number of cells are 2 times greater than the number of hits.

To make the algorithm better, grouping feature is added by introducing a new number for cells, N_G, which stores the group number that hits from the same track fall in. It works like the following;

- If the cell is untreated before ($I_2 = -1$), I_{sum} is calculated.
  - If $I_{sum} \geq N_E$, than instead of setting $I_2 = 1$, I_2 is set to a new positive number. Let us call it N_G, group number. And $N_2 = N_G$ is set. Also for neighbours of this cell, if they satisfy $I_{sum} \geq N_E$ and have at least one entry at that cell, they have the same $I_2 = N_G$. This procedure is done recursively for all good neighbouring cells. By running the algorithm like this, calculations are decreased.
  - If $I_{sum} \leq N_E$, $I_2 = 0$ and $N_2 = 0$ are set.
- If it is treated once and it is a good cell ($I_2 > 0$), than $N_2 = I_2$
- If it is treated once and it is a bad cell ($I_2 = 0$), than $N_2 = 0$

As a result, grouping is achieved, as well as filtering. For example, in Figure 3.12d, instead of just setting the cells and hits, which satisfy the minimum requirement, as good; they are labelled as m, n indicating that these two sets belong to different groups.

At ATLAS setup, because of magnetic field, sometimes, very low momentum particles spiral, and this causes a cluster of space points at the same layer. This is something unwanted, because at $\phi - \eta$ plane, they may seem as high momentum tracks. To prevent this, instead of counting the hits, that fall in that cell, the layers that have hits are counted at that cell. To save this record, bitwise operations are used.

empty	empty	empty	empty	empty
empty	2 hits on layer 6 1 hit on layer 3 3 hits on layer 1	1 hits on layer 1 1 hit on layer 2	empty	empty
empty	1 hits on layer 3 2 hit on layer 7	empty	empty	empty
empty	empty	empty	4 hits on layer 2 2 hits on layer3	empty

Figure 3.13. An example to layer histogram.

Let us see an example; assume that there is a histogram of hits from inner detector as in Figure 3.13. While cell fiiling loop, if the yellow cell is in interest, to set I_1, layers that contain hits are counted instead of counting hits. And if there is a hit on a layer

i, ith digit on mod 2 is set to 1:

$$I_{1,yellow} = (0100101)_2 = 37$$

and the for the others:

$$I_{1,blue} = (0000011)_2 = 3$$
$$I_{1,green} = (1000100)_2 = 68$$
$$I_{1,grey} = (0001010)_2 = 10.$$

As a result I_1 is a number from 0 to 127, and bits are meaningful now. To calculate I_{sum} for yellow cell, bitwise or is used;

$$I_{sum} = I_{1,yellow} \mid I_{1,green} \mid I_{1,blue}$$
$$I_{sum} = 0100101 \mid 0000011 \mid 1000100 = 1100111$$

To compare it with N_E, number of 1's of I_{sum} at mod 2 are counted which is 5 for this case and this cell is flagged as good. Also for the grey cell, with the previous method, $I_{sum} = 6$ is counted and it was flagged as good, but with the new method, $I_{sum} = 2$ is set with bitwise operations, and the cell is set bad, while it is obviously clustering of points from very low momentum particles. As a result, clustering of space points caused by very low momentum particles are prevented.

Up to now, setting the width of ϕ bins, $\Delta\phi = 1^\circ$ and $N_E = 5$ of ATLAS setup, tracks with $P_T \geq 5$GeV pass filtering. This is a rather strong filtering and could not filter lower momentum tracks. If lower momentum tracks are in intereset, there is a way to filter them, using skew transformations.

For example, in Figure 3.14a, there are two tracks, and from the histogram in Figure 3.14c, the upper track, which has high momentum, can be found, but because

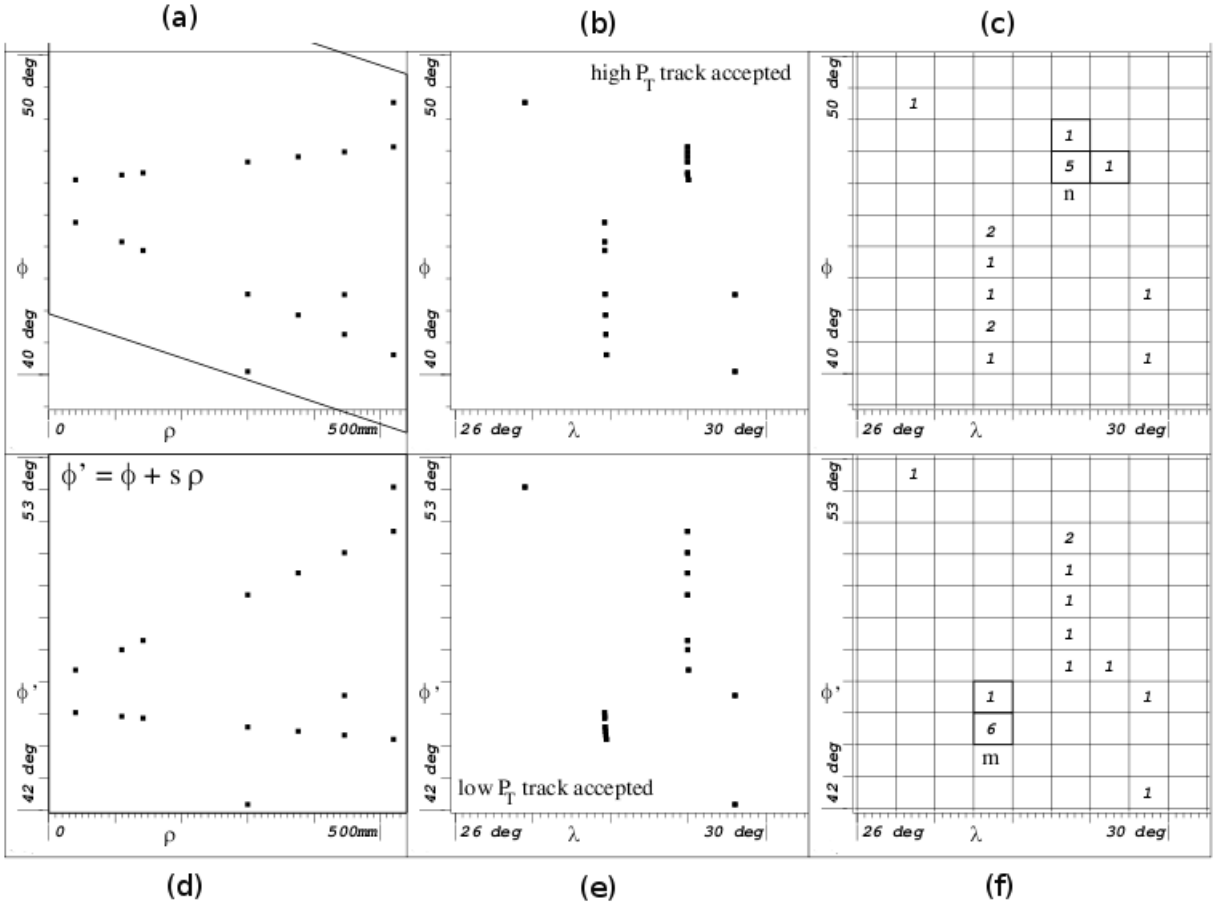

Figure 3.14. Skew transformations applied to two tracks [13].

of the great variation in ϕ, lower track, which has low momentum, can not be found. At this point, shift in $\phi - \rho$ plane is applied such that;

$$\phi' = \phi - s \cdot \rho \tag{3.18}$$

where s is the skew number and has the relation;

$$R_0 \approx \frac{1}{2s} \tag{3.19}$$

Here if $s > 0$, then $R_0 > 0$, which means negatively charged particles, and if $s < 0$, then it is a positively charged particle. Also absolute value of s, $\mid s \mid$, determines the momentum range.

So for the selected range of P_T, one can calculate R_0 and s, and by making $\phi' - \lambda$ histogram filter hits at that range of P_T as in Figure 3.14d,f. To filter highest momentum particles, $s = 0$ is set. By running the algorithm for different s values, one can find tracks with different momentum ranges, and this fastens the algorithm.

Another modification to the algorithm is that; after flagging loop, one more loop over all groups are done, and the groups that contain space points on less than N_E layers are rejected.

Lastly, instead of using $\phi' - \lambda$ histogram, $\phi' - \eta$ histogram is used, because η histogram gives more equal distribution of space points than λ for LHC experiments.

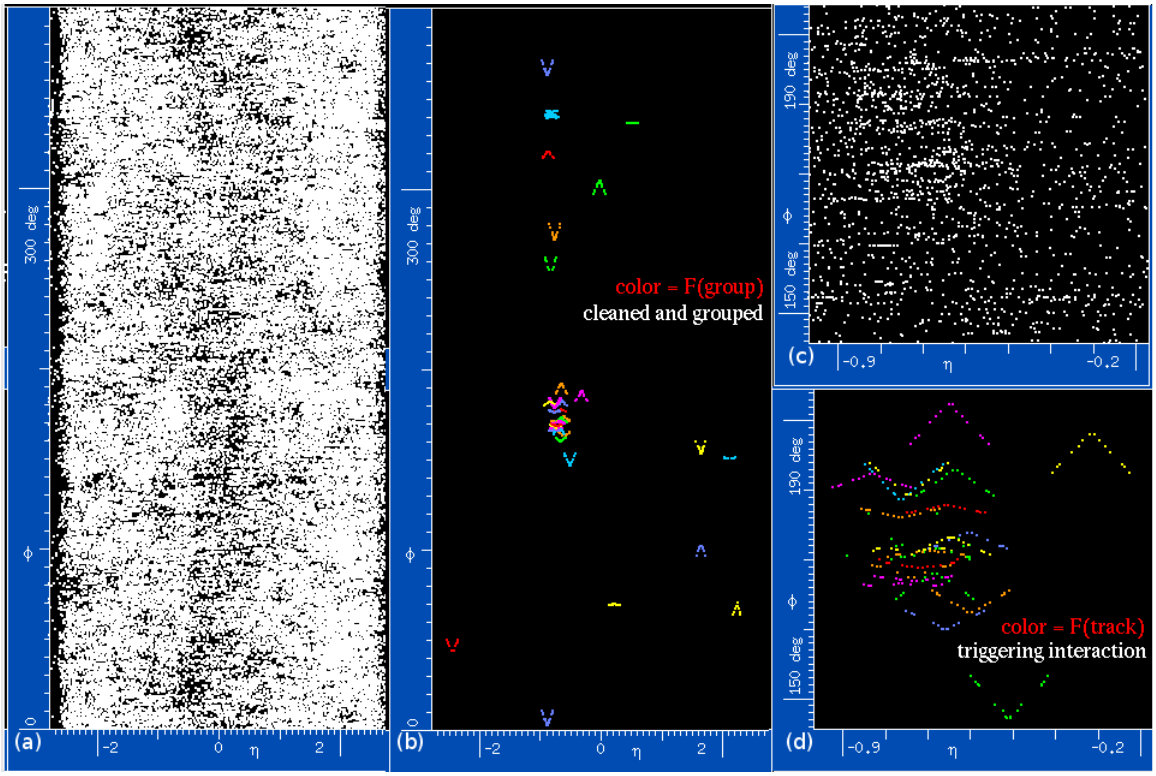

Figure 3.15. Results of hit filter running on full detector data [13].

To measure the performance of hit filter algorithm, the same data set, which is used in z-finder algorithm is filtered. Starting from 39061 space points as seen in Figure 3.15 a, and setting $\delta\phi = 1°$ and $N_E = 4$, 5 weak cleaning, i.e. for 5 different skew values, is performed and the umber of the hits are reduced to 4728. Then, setting ϕ bins smaller ($\delta\phi = 0.2°$), 31 strong cleaning is done, and it leaves us 660 space points.

At the last step, one last weak cleaning is done to remove cells with too few space points and for grouping. As a result 352 hits in 30 groups are left, which can be seen in Figure 3.15b. Figures 3.15c and d just zoomed parts of first two figures, where there is a jet. And groups are drawn in same colour in Figure 3.15b.

The comparison of simulated space points with $P_T > 1$GeV from stand alone program (Figure 3.16a,b) and the filtered space points (Figure 3.16c,d) can be seen in Figure 3.16. There are two tracks that filtering algorithm finds, but the stand alone code could not find. Checking them from the simulation they have the momentum 0.95GeV and 0.96GeV so they are not drawn in Figure 3.16a. And there are some missing space points in tracks which are close to vertex of V, i.e. in SCT detector. This is caused by the geometry of the SCT layers, and will be discussed in results chapter (Chapter 4).

The hit filter code, in C programming language can be found in Appendix B.

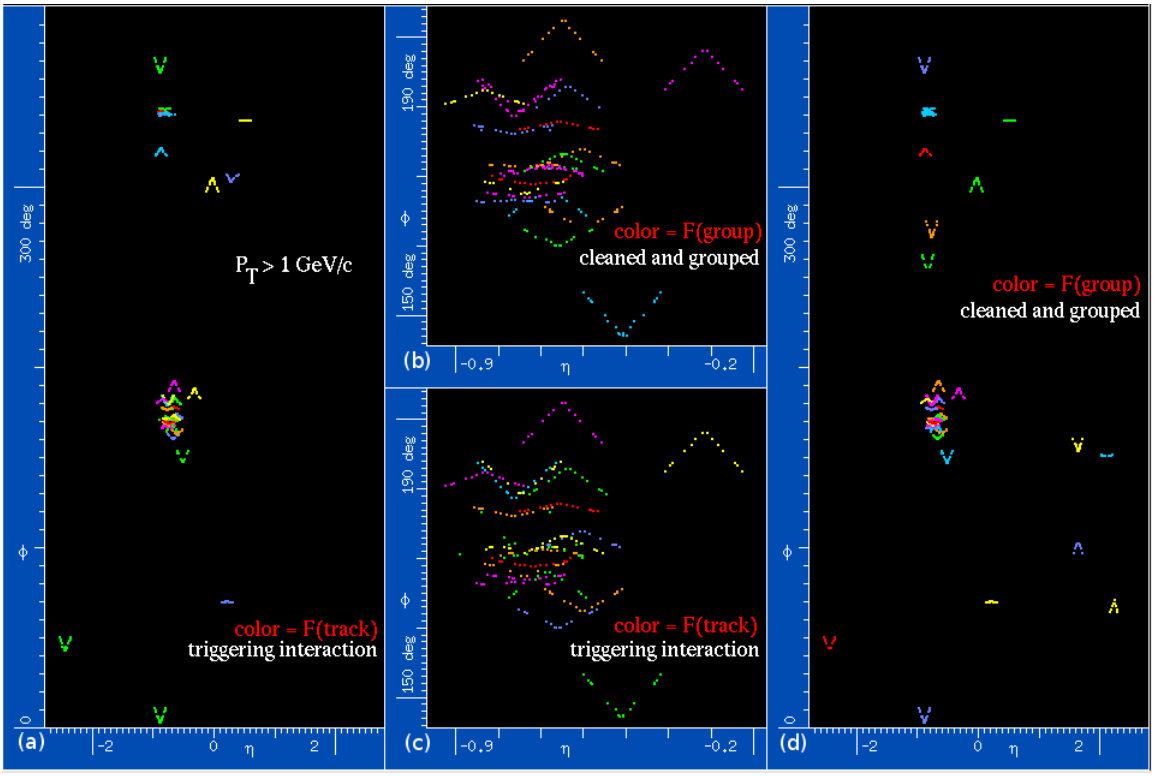

Figure 3.16. Comparision of results of hit filter with the stand alone simulation results [13].

4. ANALYSIS OF PERFORMANCE

How the z-finder (Section 3.3) and the hit filter (Section 3.5) algorithms work have been discussed up to now. Originally developed by Dr. Hans Drevermann, they were implemented in Fortran programming language. However, ATLAS trigger algorithms are written in Python and C++ languages. So they have been converted to C language in collaboration with Dr. Hans Grote. Combining the codes in the Appendices A and B, setting up the necessary parameters for the z-finder and the hit filter, the code has been optimized for high luminosity, full detector data. It has then been tested for RoI data with slightly different detector geometry. All measurements have been performed using Monte Carlo simulated with Geant4 [9] and extra random noise added as necessary. In this section, the efficiency and the results of the speed tests will be given for full detector and RoI data.

4.1. Performance of the Z-Finder

To optimize and test the code, an example high luminosity event with 39061 hits have been used with the distribution shown in the event display in Figure 4.1.

First, the position of the primary vertex is found using the z-finder algorithm. Currently the algorithm uses data only from the barrel section, which is defined by $\lambda \leq 49°$. By using z-finder algorithm, one can also find secondary vertices, by just setting a low threshold of $P_T = 0.5$GeV. The comparison of results with the simulation can be seen in Figure 4.2.

As seen in the figure, the algorithm is able to find both the primary vertex and the secondary vertices quite clearly in this event. Then, to check the efficiency in other events, simulated events with muon RoIs are used. In these events single muons of 10GeV that are fired from random positions along the interaction region, are embedded into high luminosity environment. To properly measure the z-finder performance, only these events, where the muon goes along the barrel region, are selected. The results

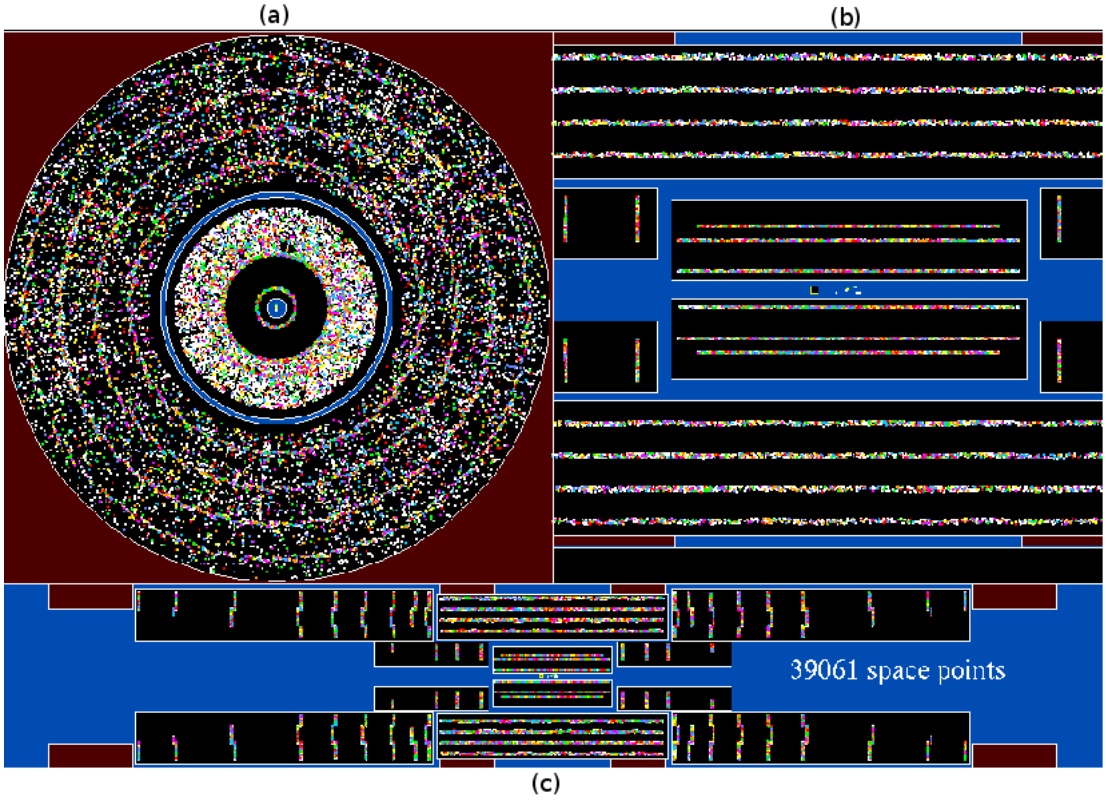

Figure 4.1. Event display for the example event used in the optimization for full detector running. Projections are shown in: a)$x-y$ plane b)$\rho-z$ plane (barrel part) c)$\rho-z$ plane. Due to the large number of hits the detector module are clearly visible [13].

are compared with the generated vertex positions, that event filter (EF) and IdScan can find. The comparison is done with the event filter, because event filter is an offline algorithm for ATLAS detector system, that is why it is best for trigger tracking. And also, if EF could not find a track, it means that even for unlimited time of tracking, the track could not be found, i.e. it is lost. The comparison is also done with the IdScan, because, the algorithm defined here is similar to IdScan. However, comparison with SiTrack gives the same results. A data set of 1030 events, the new algorithm defined in Section 3.3, is able to find 95.4% of primary vertex positions of data sets. It should be noted that there were initially a couple of problematic data sets, where the position was found incorrectly, but analysing the data in detail showed that, two z positions existed in these events, and optimizing the part of the code, which finds z_V from the histogram filled, this problem could be solved for 2.6% of the data sets. If this problem is solved by manipulating on the code or optimizing the cuts, efficiency of 98% can be

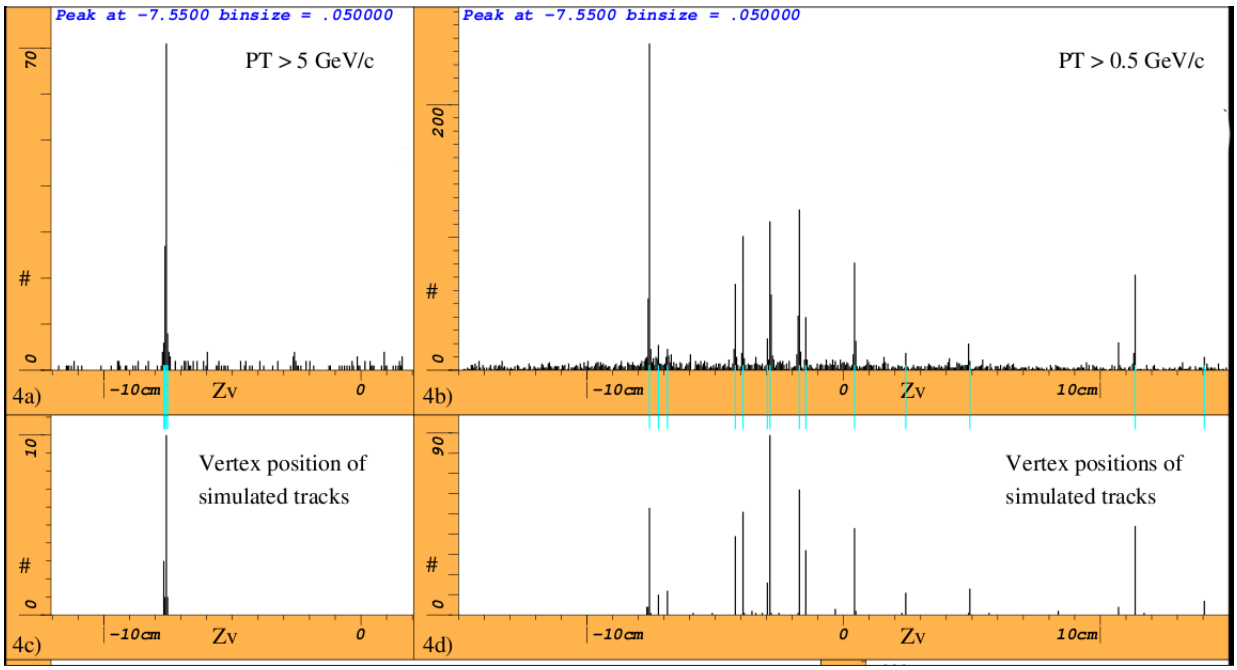

Figure 4.2. Comparison of results of z-finder and the simulation, for primary (left) and secondary (right) vertices [11].

achieved. Fitting the differences between the reconstructed and the true z_V positions with a Gaussian, it is observed that the position of the primary vertex is found with no bias and with a resolution of 0.2 mm, as can be seen in Figure 4.3.

As mentioned before, the algorithm currently functions only in the barrel part of the inner detector. Hence for the RoIs, that fall partially or fully in the end-cap region, z position can not be found. Hence, extending the algorithm to the end-caps is necessary before any further studies are done.

4.2. Performance of the Hit Filter

After the position of the interaction has been found, the hits are sent to the hit filter. Hit filter selects the hits which belong to tracks with $P_T > 1$GeV and groups them. We first use the full-detector example event to study the performance of the hit filter. Amongst the 39061 hits, 334 hits are selected and grouped in to 30 groups. As seen in Figure 3.16, hit filter gives the same results as the simulation. For the RoI data, hit filter gives the hits of the single embedded muon, i.e. the single physics track,

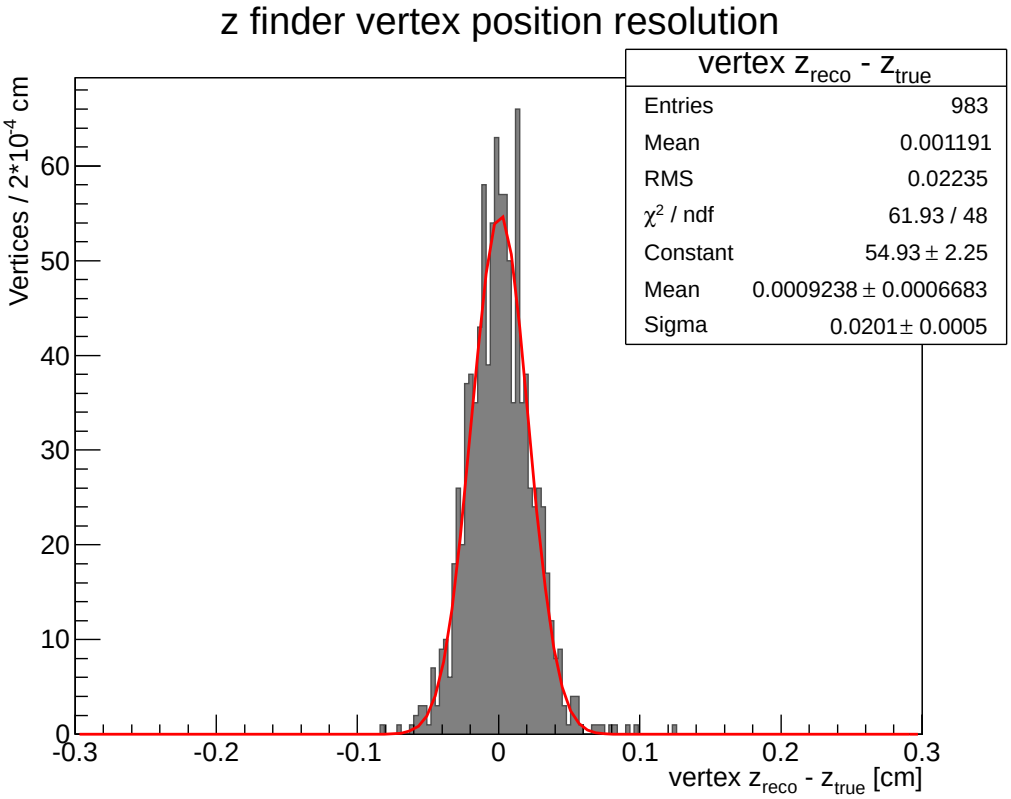

Figure 4.3. Difference between the primary vertex positions from the z-finder and the simulation.

quite well also.

To measure the filtering performance of RoI data, comparison of ϕ_0 and η of the single track, found by the hit filter and reconstruction, is done. As seen in Figures 4.5 and 4.6, both *eta* and ϕ can be found with no bias and with an error of 0.00020 for *eta* and 1.8mRad for ϕ.

The only problem, with the filtering algorithm is some small shift on the SCT layers. In Figure 4.4, this problem is demostrated. In this Figure, a reconstructed track is shown. The track seems straight on $\rho - z$ plane. However applying the skew transformations, to make the track seem vertical, such that;

$$z = z - \rho \cdot \tan(\lambda)$$

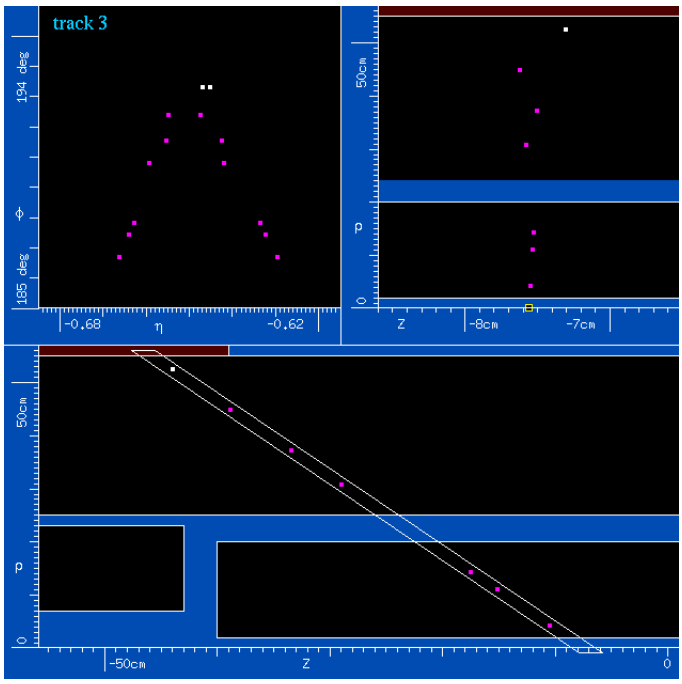

Figure 4.4. Lost hits from the SCT layers, because of shifts on SCT hits on V-plot [12].

in figure on top-right, the hits on the SCT layers are seen to make zigzags. This is because of the geometry of SCT layers, where for each SCT component there are 2 strips, ϕ strip and stereo strip, placed with a pitch of 80μm and stereo angle of 2.3°. When a particle passes through the strips, extrapolation is done with the information coming from these strips. Despite this zigzaging behaviour, the hit filter seems to be successful in putting together the relevant hits from the tracks, thanks to the fact that it is looking at neighbouring bins during reconstruction. However, we also note that, the zigzag behaviour worsens as we move away from the beam line, and hence, some outermost hits could be lost in general. Further studies are needed to address this issue and improve the performance.

4.3. Timing Performance

A trigger tracking algorithm is expected to have 100% efficiency and high background rejection rate. We observe that the filter code is working fine for both full detector and RoI data, and the vertex positions are successfully found by the z-finder. Another essential feature of a trigger tracking algorithm is the speed, because there is limited time between the bunch crossings with strict constraints on latency. Time for

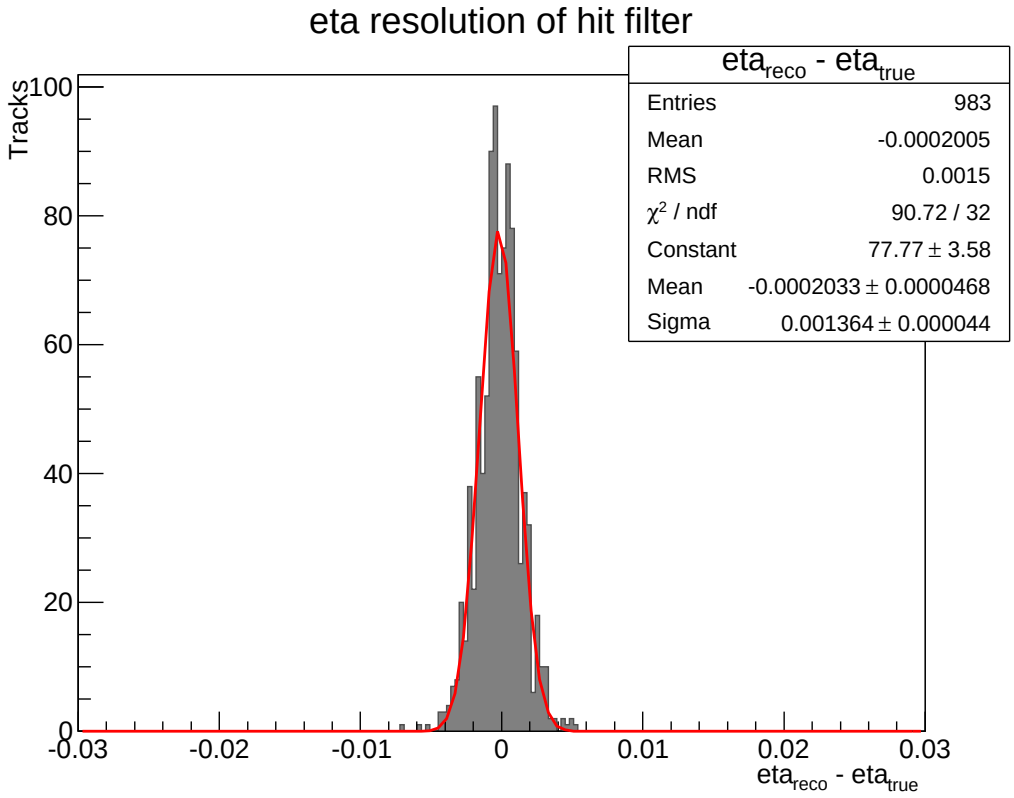

Figure 4.5. η resolution of hit filter.

the total LVL2 decision should be approximately 40ms on average. So, to check the fastness of algorithm, two data sets have been considered, one full detector data with 39061 space points and one muon RoI data 315 space points. The tests are performed on a computer with Intel(R) Core(TM) i7 CPU running at 2.93GHz and with 4GB of memory. The results are summarized in Table 4.1. As seen from the table, timing-wise the original code which is written in the Fortran programming language does not differ from the code written in the C programming language. Taking into consideration that the computational farms used by the ATLAS Trigger system are similar to the computer that is used to check timing of the code, this algorithm seems to be satisfactory for being used by the ATLAS LVL2 trigger both for full detector data and RoI interest data.

As a result, the algorithm is able to find z-position of primary vertex, to filter the hits that belongs to tracks with $P_T > 1$GeV successfully. And also it seems to be fast enough to be used as a part of the ATLAS trigger tracking system.

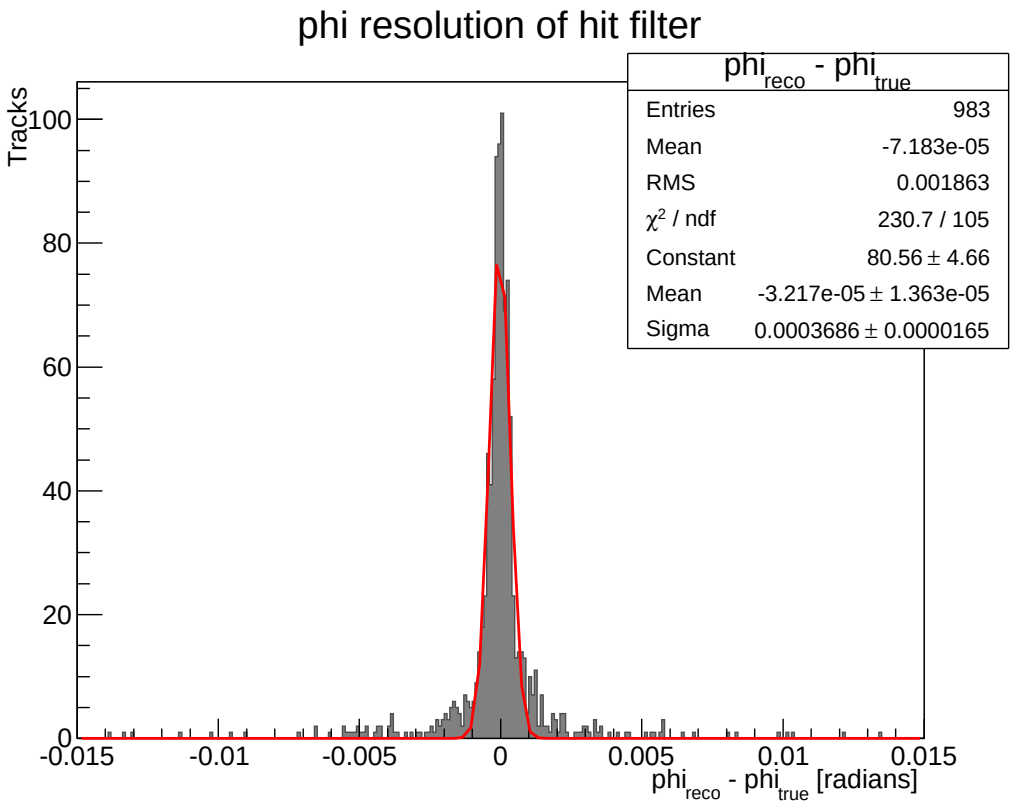

Figure 4.6. ϕ resolution of hit filter.

Table 4.1. z-finder and hit filter timing results for full detector and RoI data.

		Fortran code	C code
Full detector data	(barrel) z-finder	2.74±0.17msec	2.24±0.14msec
	z-finder+hit filter	32.0±1.8msec	29.7±1.7msec
RoI with 315 hits	(barrel) z-finder	82.8±3.70nsec	23.5±1.0nsec
	z-finder+hit filter	570±10nsec	523±21nsec

5. CONCLUSION

A new trigger tracking algorithm, to be used for high luminosity ATLAS data is described. The algorithm consists of mainly two parts; the z-finder and the hit filter, and can be applied to both full detector data and RoI data. It was originally written in Fortran programming language, and it has now been converted to C, for applicability to ATLAS LVL2 trigger system. For RoI data with 10GeV muons, it can provide 98 % efficiency with an average run time of 0.6 msec. Also even for the full detector data the algorithm has a run time of 30 msec, which might be considered acceptable for special-purposes uses.

Z position of the primary vertex is found with no bias and with a resolution of 0.2mm. For full detector data, z-finder is able to find not only the primary vertex, but also the secondary vertices by just setting a low P_T cut-off. Using the identified z vertex positions, the algorithm is able to reconstruct the embedded physics track in muon RoIs, with η and ϕ_0 resolution of 0.0002 and 1.8mRad respectively.

Hence, the z-finder and the hit filter seem to satisfy both the performance and the timing needs of ATLAS LVL2. during the upcoming long shut down of ATLAS, the code can be modified with the new proposed detector geometry, and can be converted to into a format,that is fully compliant with the LVL2 software infrastructure.

As further studies, the working range of the z-finder should be extended beyond the barrel region to cover the entire detector including the end-caps. Some minor details, that has been outlined on the thesis, indicate that further performance improvements are also possible.

And finally, we look forward to optimizing the algorithm for different physics events and real detector data.

APPENDIX A: Z-Finder Algorithm in C programming language

```
1
2  void zv_execute(int *np, int mp_loc, int *nk1, int *nk2, int *nu, float *
       zv)
3  {
4      int ipx01[] = { 0, 0 }, isct = 0;
5      int ipmf[] = { 0, 0, 0, 1, 1, 1, -1, -1, -1 };
6      int ipmz[] = { -1, 0, 1, -1, 0, 1, -1, 0, 1 };
7      int ifi, i, k, k1, k2, l, l1, l2, ls, ly, nf, nf0, nsct3, num, nz, ns
           , nbf1, izz;
8      float dz12, z0, qz, df, qf, zz;
9      *np = 0;
10
11     /* ..................................... link list */
12     nbf1 = c_param.nbf - 1;
13
14     for (k = c_data.nh; k > 0; k--) {
15   ly = c_data.iw[jl][k];
16   zz = c_data.w[jz][k];
17   if (ly <= 6 && zz >= -c_param.zl[ly] && zz <= c_param.zl[ly]) {
18       if (ly >= 3) {
19     ifi = c_data.w[jf][k] * c_param.bf;
20     izz = (int) (zz * c_param.bz);
21     for (i = 0; i < 9; i++) {
22         c_zero.ns[ly - 3][ifi + ipmf[i] + nsoff][izz + ipmz[i] + mlz] =
               1;
23     }
24     c_data.iw[j1][k] = isct;
25     isct = k;
26       } else if (ly >= 1) {
27     ifi = c_data.w[jf][k] * c_param.af;
28     c_data.iw[j2][k] = c_zero.ifi12[ifi][ly];
29     c_zero.ifi12[ifi][ly] = k;
30     if (ly == 1) {
31         c_data.iw[j1][k] = ipx01[ly];
```

```
32          ipx01[ly] = k;
33      }
34        } else {     /* ly=0 */
35
36      c_data.iw[j1][k] = ipx01[ly];
37      ipx01[ly] = k;
38        }
39    }
40      }
41      /* ..............................  sct wrap around */
42      for (l = 0; l <= 3; l++) {
43    for (nz = -mlz; nz <= mlz; nz++) {
44        c_zero.ns[l][0 + nsoff][nz + mlz]
45      = c_zero.ns[l][c_param.nbf + nsoff][nz + mlz]
46      | c_zero.ns[l][0 + nsoff][nz + mlz];
47        c_zero.ns[l][c_param.nbf + nsoff][nz + mlz]
48      = c_zero.ns[l][nsoff - 1][nz + mlz]
49      | c_zero.ns[l][c_param.nbf + nsoff][nz + mlz];
50    }
51      }
52    /*========== main loop ========== */
53      nsct3 = c_param.nsct - 3;
54      for (l1 = 0; l1 <= 1; l1++) {
55    for (l2 = l1 + 1; l2 <= 2; l2++) {
56        k1 = ipx01[l1];
57        while (k1 != 0) {
58      nf0 = c_param.af * c_data.w[jf][k1];
59      for (nf = nf0 - c_param.ndf[l1][l2]; nf <= nf0 + c_param.ndf[l1][l2];
            nf++) {
60          k2 = c_zero.ifi12[c_param.irr[irroff + nf]][l2];
61          while (k2 != 0) {
62        dz12 = c_data.w[jz][k2] - c_data.w[jz][k1];
63        z0 = c_data.w[jz][k1] - c_data.w[jr][k1] * dz12 / (c_data.w[jr][k2]
              - c_data.w[jr][k1]);
64        if (z0 >= -zmax && z0 <= zmax) {
65            qz = dz12 / (c_data.w[jr][k2] - c_data.w[jr][k1]);
66            df = c_data.w[jf][k2] - c_data.w[jf][k1];
67            if (fabs(df) <= c_param.fl[l1][l2]
```

```
68          || fabs(df) >= c_param.fh[l1][l2]) {
69          qf = df / (c_data.w[jr][k2] - c_data.w[jr][k1]);
70          num = 0;
71          for (ls = 0; ls <= 3; ls++) {
72              izz = (int) (c_param.bz * (c_data.w[jz][k1]
73                      + qz * (c_param.rs[ls] - c_data.w[jr][k1])));
74              ifi = c_param.bf * (c_data.w[jf][k1]
75                + qf * (c_param.rs[ls] - c_data.w[jr][k1]));
76              if (c_zero.ns[ls][ifi + nsoff][izz + mlz])
77            num++;
78              if (num < ls + nsct3)
79            goto nine;
80          }
81          if (*np >= mp_loc)
82              goto eleven;
83          *np = *np + 1;
84          nk1[*np] = k1;
85          nk2[*np] = k2;
86          nu[*np] = num;
87          zv[*np] = z0;
88            }
89        }
90              nine:k2 = c_data.iw[j2][k2];
91          }
92      }
93      k1 = c_data.iw[j1][k1];
94        }
95    }
96      }
97      /*.............................. set 0 */
98    ten:
99      k = isct;
100     while (k > 0) {
101   ly = c_data.iw[jl][k];
102   ifi = c_data.w[jf][k] * c_param.bf;
103   izz = (int) (c_data.w[jz][k] * c_param.bz);
104   for (i = 0; i < 9; i++)
105       c_zero.ns[ly - 3][ifi + ipmf[i] + nsoff][izz + ipmz[i] + mlz] = 0;
```

```
106     k = c_data.iw[j1][k];
107       }
108       for (l = 0; l <= 3; l++) {
109     for (nz = -mlz; nz <= mlz; nz++) {
110         c_zero.ns[l][nsoff + 0][nz + mlz] = 0;
111         c_zero.ns[l][c_param.nbf + nsoff][nz + mlz] = 0;
112     }
113       }
114       isct = 0;
115       for (ly = 1; ly <= 2; ly++) {
116     for (ifi = 0; ifi <= mfi12; ifi++)
117         c_zero.ifi12[ifi][ly] = 0;
118       }
119       ipx01[0] = 0;
120       ipx01[1] = 0;
121       return;
122     eleven:
123       printf("np= %d  too bign", *np);
124       goto ten;
125 }
```

APPENDIX B: Hit Filter algorithm in C programming language

```
1
2  void fi_filter(float zvx)
3  {
4      char trun[2][12] = { "preloop", "main loop" };
5      int j, n;
6      float skew;
7
8      fi_store_all(zvx);
9
10     c_nhits.nfrom[0] = c_fi_data.nv;
11     c_nhits.nfrom[1] = c_fi_data.nv;
12     c_nhits.nfrom[2] = c_fi_data.nv;
13
14     for (j = 0; j < 2; j++) {
15   if (c_fi_param.nloop[j] > 0) {
16       n = 0;
17       skew = 0.;
18       fi_clean(c_fi_param.bet[j], c_fi_param.bfi[j], c_fi_param.lcut[j],
             skew, j);
19
20       for (n = 1; n <= c_fi_param.nloop[j]; n++) {
21     skew = skew + c_fi_param.dskew[j];
22     fi_clean(c_fi_param.bet[j], c_fi_param.bfi[j], c_fi_param.lcut[j],
           skew, j);
23     fi_clean(c_fi_param.bet[j], c_fi_param.bfi[j], c_fi_param.lcut[j], -
           skew, j);
24       }
25       if (c_fi_param.nloop[j + 1] >= 0 || c_fi_param.nloop[2] >= 0) {
26     c_nhits.nfrom[j] = c_fi_data.nv;
27     fi_store_group1();
28     c_nhits.nto[j] = c_fi_data.nv;
29       }
30   }
31     }
```

```
32
33      if (c_fi_param.nloop[2] >= 0) {
34    fi_group(c_fi_param.bet[2], c_fi_param.bfi[2], c_fi_param.lcut[2]);
35    c_nhits.nfrom[2] = c_fi_data.nv;
36    if (c_fi_param.nloop[2] >= 1) {
37        fi_store_group1();
38        c_nhits.nto[2] = c_fi_data.nv;
39        fi_group(c_fi_param.bet[2], c_fi_param.bfi[2], c_fi_param.lcut[2]);
40    }
41      }
42 }
43
44 void fi_clean(float be, float bf, int lc, float skew, int j)
45 {
46      int ihf[] = { 0, 0, 1, 1, 1, -1, -1, -1,
47    1, 0, -1, 1, 0, -1, 2, 2, 2, 2, 2, -2, -2, -2, -2, -2
48      };
49      int ihe[] = { -1, 1, -1, 0, 1, -1, 0, 1,
50    -2, -2, -2, 2, 2, 2, -2, -1, 0, 1, 2, -2, -1, 0, 1, 2
51      };
52      int i, k, n, nl, met1 = met - 1, ifi, iet, lay;
53      float qsk = -0.1719;  /* = 57.296/166.666 = r2dgdi/2c */
54      float eta, skw, phi, rho, zro;
55
56
57      for (i = 0; i < 24; i++) {
58    c_neighbour.ihe[i] = ihe[i];
59    c_neighbour.ihf[i] = ihf[i];
60      }
61
62      /* ---------------------------------------- eta channel */
63      if (skew == 0.) {
64    for (n = 1; n <= c_fi_data.nv; n++) {
65        eta = c_fi_data.rv[me][n];
66        c_fi_data.iv[mt][n] = nint(be * eta);
67        c_fi_data.iv[mg][n] = 0;
68    }
69      }
```

```
70      /* ---------------------------------------- fill histogram */
71      skw = qsk * skew;
72      for (n = 1; n <= c_fi_data.nv; n++) {
73    phi = c_fi_data.rv[mf][n];
74    rho = c_fi_data.rv[mr][n];
75    ifi = bf * fmod(phi - skw * rho + 360., 360.);
76    c_fi_data.iv[ma][n] = ifi;
77    iet = c_fi_data.iv[mt][n];
78    lay = c_fi_data.iv[ml][n];
79    c_filter.nef[met + iet][ifi] |= 1 << lay;
80      }
81
82      /*c------------------------------------------------- decide */
83      for (n = 1; n <= c_fi_data.nv; n++) {
84    iet = c_fi_data.iv[mt][n];
85    ifi = c_fi_data.iv[ma][n];
86    lay = c_fi_data.iv[ml][n];
87    i = c_filter.nef[met + iet][ifi];
88
89    for (k = 0; k < c_neighbour.ito[lay][j]; k++) {
90        i |= c_filter.nef[met + iet + ihe[k]][c_wrap_around_f.ifr[2 + ifi +
              ihf[k]][j]];
91    }
92    nl = c_fi_bits.nbits[i];
93    if (nl >= lc) {
94        c_fi_data.iv[mg][n] = 1;
95    }
96      }
97
98      /*c ---------------------------------------- set zero */
99      for (n = 1; n <= c_fi_data.nv; n++)
100   c_filter.nef[met + c_fi_data.iv[mt][n]][c_fi_data.iv[ma][n]] = 0;
101 }
```

APPENDIX C: Mathematical Proof of the shape V in V-plot

How to construct V shape for in $\phi - \eta'$ is shown in Figure 3.10. Here the mathematical explanation is shown starting from the helix equations. It is obvious from the Equations 3.16 and 3.15, the definitions of eta'_+ and η'_-, that if $\rho'' = \rho$, i.e at the outermost layer of SCT, $\eta'_+ = \eta'_-$. This shows us that, the arms join at sa point. Le us examine the shape of arms. From the Figure 3.10, it is expected that the arms are linear and have the same slope with an inverse sign. To prove this $\frac{d\phi}{d\eta'_+}$ and $\frac{d\phi}{d\eta'_-}$ are evaluated:

$$\frac{d\phi}{d\eta'_+} = \frac{1}{\frac{d\eta'_+}{d\phi}}$$

$$\frac{d\eta'_+}{d\phi} = \frac{d\eta}{d\phi} - k\frac{d\rho}{d\phi}$$

from Equation 3.16.

$$\frac{d\eta}{d\phi} = \frac{d\eta}{d\lambda}\frac{d\lambda}{d\phi}$$

$$\frac{d\eta}{d\lambda} = \frac{\sec^2(\lambda)(1 - \frac{2\tan(\lambda)}{\sqrt{1+\tan^2(\lambda)}})}{\tan(\lambda) + \sqrt{1+\tan^2(\lambda)}}$$

from Equation 3.14. This equation tells us that, even for $\lambda = 0$ $\frac{d\eta}{d\lambda}$ is non zero.

Also from Equation 3.5

$$\frac{d\rho}{d\phi} = 2R_0$$

this equation holds only for high momentum tracks and where $\delta\phi$ changes slightly, so the results are valid only for high momentum tracks.

Combining the results to find $\frac{d\phi}{d\eta'_+}$;

$$\frac{d\phi}{d\eta'_+} = 1/(\frac{d\eta}{d\lambda}\frac{d\lambda}{d\phi} - 2kR_0)$$

We know that $\frac{d\eta}{d\lambda}$ is non zero and $\frac{d\lambda}{d\phi} = 0$ for particles coming from the primary vertex. So the first term of denominator vanishes;

$$\frac{d\phi}{d\eta'_+} = \frac{-1}{2kR_0} \tag{C.1}$$

$$\frac{d\phi}{d\eta'_-} = \frac{1}{2kR_0} \tag{C.2}$$

k is a predefined constant and R_0 is constant for helices, as a result $\frac{d\phi}{d\eta'_+}$ and $\frac{d\phi}{d\eta'_-}$ are constant and equal with different signs, i.e. the arms form a shape of V for high momentum particles coming from the primary vertex.

REFERENCES

1. ATLAS Collaboration, *The ATLAS Experiment at the CERN Large HadronCollider*, Institute of Physics Publishing and SISSA, 2008.

2. ATLAS Colloboration, *SM Higgs Production Cross Sections at* $\sqrt{s} = 14\ TeV$, CERN Report 2011-002 Numbers, 2012.

3. ATLAS Colloboration, *Observation of a New Particle in the Search for the Standard Model Higgs Boson with the ATLAS Detector at the LHC*, ATLAS-HIGG-2012-27-002, CERN-PH-EP-2012-218, 2012.

4. Negri, A., *ATLAS TDAQ System: Integration and Commissioning*, talk on behalf of the ATLAS TDAQ collaboration, University and INFN of Pavia, 2009.

5. Saul Gonzalez, *The ATLAS Trigger System*, talk on behalf of the ATLAS (Trigger) collaboration, University of Wisconsin, Madison, 1999.

6. Coccaro, A., *Tracking and B-tagging for the ATLAS Trigger System*, Ph.D. Thesis, Universita Degli Studi di Genova, 2010.

7. ATLAS Colloboration, *Visualizing the ATLAS Inner Detector with Atlantis*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Vol. 549, No. 1-3, 2005.

8. Ristic B., Arulampalam S., Gordon N., *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Artech House Radar Library, Boston, London, 2004,

9. S. Agostinelli, et al., *Geant4-a Simulation Toolkit*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Vol. 506, No. 3, 2003.

10. Drevermann, H., *Helix Equations*, 2008,
http://drevermann.web.cern.ch/drevermann/helix/, accessed at September 2012.

11. Drevermann, H., *A Method to Find the z-position of the Primary Vertex from Space Points*, 2010,
http://drevermann.web.cern.ch/drevermann/z_ finder/, accessed at September 2012.

12. Drevermann, H., *The V-plot, a Three Dimensional Representation of Tracking Data from HEP Experiments*, 2010,
http://drevermann.web.cern.ch/drevermann/v_ plot/, accessed at September 2012.

13. Drevermann, H., *The Filter*, 2008,
http://drevermann.web.cern.ch/drevermann/hit_ filter/, accessed at September 2012.