laSalle ENG
Ramon Llull University

**Escola Universitària d'Enginyeria Tècnica de Telecomunicació La Salle**

Treball Final de Grau

Grau en Enginyeria Informàtica

# An approach to track reconstruction for the SciFi tracker at LHCb using ANNs and spatial indexing

Alumne
Gabriel Cammany Ruiz

Professor Ponent
Alessandro Camboni

# ACTA DE L'EXAMEN
# DEL TREBALL FI DE CARRERA

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. Gabriel Cammany Ruiz

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

An approach to track reconstruction for the SciFi tracker at LHCb using ANNs and spatial indexing

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

# An approach to track reconstruction for the SciFi tracker at LHCb using ANNs and spatial indexing

by

Gabriel Cammany Ruiz

Computer Engineering

at

Ramon Llull University, La Salle BCN

May 2019

**Abstract**

An approach to track reconstruction for the SciFi tracker at LHCb using ANNs and spatial indexing

by

Gabriel Cammany Ruiz

Computer Engineering

Ramon Llull University, La Salle BCN

PhD. Alessandro Camboni, Tutor

PhD. Xavier Vilasís Cardona, Co-tutor

The LHCb experiment at CERN will undergo an internal transformation over the coming two years, during a maintenance and upgrade period known as Long Shutdown 2 (LS2). This improvement aims at extending the physics reach of the experiment allowing it to run with a proton–proton collision rate 5 times higher than before at which it will be working once the Large Hadron Collider (LHC) restarts in 2021.

The upgrade not only affects the sub-detectors but also the pattern recognition algorithms whose purpose is to reconstruct particle trajectories, since there is a need to cope with the increased complexity of the physical environment. This project aims at using Artificial Neural Networks and spatial indexing data structures, such as R-trees, to provide a different approach to track reconstruction with respect to the algorithms which are currently used in LHCb.

**Key words**

Track reconstruction; SciFi; LHCb; LHCb upgrade, Artificial Neural Networks, R-trees, Pattern Recognition;

# Contents

**Contents** **i**

**List of Figures** **iii**

**List of Tables** **vi**

**1 Introduction** **1**
1.1 Standard model . . . . . 1
1.2 Particle Accelerators . . . . . 2

**2 The LHCb Experiment** **5**
2.1 LHC at CERN . . . . . 5
2.2 The LHCb detector . . . . . 6
2.3 LHCb systems . . . . . 8
2.4 LHCb Upgrade . . . . . 13

**3 The Scintillating Fibre Tracker** **15**
3.1 General description . . . . . 15
3.2 Scintillating Fibres . . . . . 16
3.3 Silicon Photomultipliers (SiPM) . . . . . 17
3.4 Read-out electronics . . . . . 17
3.5 Patter recognition Algorithms . . . . . 18
3.6 Performance indicators . . . . . 19

**4 Tracking data** **21**
4.1 Monte Carlo Simulation . . . . . 21
4.2 Dataset description . . . . . 22

**5 The Hybrid Seeding algorithm** **29**
5.1 Overview . . . . . 29
5.2 Find x-z projections . . . . . 29
5.3 x-z clone killing . . . . . 30
5.4 Stereo Hits . . . . . 31

5.5 Global clone removal . . . . . 32
5.6 Track recovery . . . . . 33
5.7 Performance . . . . . 33

**6 R-trees** **35**
6.1 Overview . . . . . 35
6.2 Operations . . . . . 37

**7 Artificial Neural Networks** **39**
7.1 Overview . . . . . 39
7.2 Architecture of ANNs . . . . . 40
7.3 Learning . . . . . 41

**8 The algorithm** **45**
8.1 Overview . . . . . 45
8.2 R-trees . . . . . 46
8.3 Artificial Neural Networks Models . . . . . 48
8.4 Station seed reconstruction . . . . . 53
8.5 Unification of stations seeds . . . . . 56
8.6 Performance . . . . . 57

**9 Conclusions and time distribution** **63**
9.1 Conclusions . . . . . 63
9.2 Time management . . . . . 65

**Bibliography** **67**

# List of Figures

1.1 Classification of particles based on their spin. . . . . 2

2.1 Schema of the CERN complex showing the different particle accelerators. In order to achieve 7 TeV per beam the process takes 5 steps. LINAC 2 starts accelerating particles up to 50 MeV, the BOOSTER then delivers particles at 1.4 GeV and injects them into the PS, which reaches the energy of 26 GeV. The following step is represented by the SPS which accelerates particles up to 450 GeV before finally injecting them into the LHC. Figure taken from [2]. . . . . 6
2.2 LHCb detector side view. Figure taken from [2]. . . . . 7
2.3 Track type definition in the LHCb detector. . . . . 10
2.4 Trigger System diagram from the 2012 data taking period (Run 1) . . . . . 12
2.5 Trigger strategies adopted from the Run 1 to the upcoming Run 3. Taken from [13]. . . . . 14

3.1 Schematic 3D view of the *Scintillating Fibre Tracker* located between the dipole magnet and the RICH2 system. Figure taken from [2]. . . . . 15
3.2 Schematic front and side view of the *Scintillating Fibre Tracker* located between the dipole magnet and the RICH2 system. . . . . 16
3.3 Schematic of the plastic scintillating fibre. Particles traversing the fibre produce light in the core, only the light produced inside the trapping angle is propagated within the fibre through total internal reflection. The claddings are arranged by decreasing indices of refraction to increase the amount of trapped light. Figure taken from [2]. . . . . 17
3.4 SiPMs have a total of 128 read-out channels, coupled to a total of six layers of stacked fibres. Each of these read-out channels are composed of a rectangular matrix of pixels, responsible for performing the photon counting. The illustration on the right shows how a position for a given particle is computed, since the position is obtained by the average charge collected from each pixel within the read-out channel. Figure taken from [14]. . . . . 18

4.1 Schematic of the LHCb execution flow from both the real detector and the MC simulation. . . . . 22

4.2 2D view of tracks using *MCHits* together with a visual representation of *X-U-V-X* layers. . . . . 23
4.3 Three dimensional view of 100 particles through the SciFi sub-detector by using the *monte carlo truth* together with a visual representation of the *X-U-V-X* layers. 24
4.4 View of particle trajectories from the first layer looking downstream the experiment can be found in figure (*a*). In fig. (*b*) it is possible to see the small impact the dipole magnet has over the SciFi stations. . . . . 25
4.5 Visual representation of *PrHits* associated to 100 particles. As can be seen, when reconstructing tracks using *PrHits*, those in $U$ and $V$ layers seem deviated from the true particle trajectory, because of the $\pm 5$ degree difference between $X$ and stereo layers. . . . . 27

5.1 Overview of the Hybrid Seeding algorithm structure. Figure taken from [14]. . . 30
5.2 Visual representation of the first three steps the Hybrid Seeding Algorithms performs. Figure taken from [14]. . . . . 31
5.3 Graphic representation of how the $y$ position is computed. Fig. taken from [14]. 32

6.1 The leaf nodes of the R-tree store the exact *MBRs* or bounding boxes of the individual geometric objects, along with a pointer to the storage location of the contained geometry. All non-leaf nodes store references to several bounding boxes for each of which is a pointer to a lower level node [4]. Figure taken from [9] . . 36
6.2 Comparison between efficient and inefficient split approaches when performing a node split. Figures taken from [6]. . . . . 38

7.1 Sketch of a *perceptron* structure. Figure taken from [12] . . . . . 40
7.2 Illustration of a artificial network with two hidden layers. Figure taken from [12] 41
7.3 Example of a cost function dependent of two variables. The gradient descendent algorithm would compute the direction that minimises the cost in a given position, the green ball and the arrow. Iteratively, the variables would be updated to make the ball move towards the bottom, thus reducing the function. Figure taken from [12] . . . . . 42

8.1 The first figure show a histogram representation of the the ability to separate the valid and invalid hit combinations. A *ROC* curve is illustrated in the second plot, with an *Area under the Curve* (AUC) of 0.915. . . . . 50
8.2 The first figure show a histogram representation of the the ability to separate the valid and invalid seed combinations. A *ROC* curve is illustrated in the second plot, with a $AUC$ of 0.928. . . . . 51
8.3 The first figure show a histogram representation of the the ability to separate the valid and invalid track combinations. A *ROC* curve is illustrated in the second, with a $AUC$ of 0.998. . . . . 52
8.4 Difference between the predicted positions of *X0-U* and *X1-V*. . . . . 54

8.5 Difference histograms of the real position versus the predicted ones using line projections for *U-V-X1* layers respectively. *n* represents the number of particles used . . . . . 55
8.6 Scatter plots containing the *Track reconstruction efficiency*, *Ghost rate* and *Seed reconstruction efficiency* versus the number of hits per event. . . . . 59

9.1 Representation with the form of a circular diagram the time distribution of the project . . . . . 65

# List of Tables

5.1 Selection of the *X-layer* in T1 and T3 station based on the case. . . . . 30
5.2 Description for each of the sub-sections for the different types of tracks. . . . . 33
5.3 Tracking performances for different track types of the Hybrid Seeding algorithm, along with the ghost rate, hit purity and hit efficiency. . . . . 34

8.1 Proportion of particles with missing hits in the SciFi simulated data. . . . . 58
8.2 Efficiency, *ghost rate* and seed reconstruction efficiency for all particles. . . . . 60
8.3 Efficiency, *ghost rate* and *seed reconstruction* efficiency with particles having hits in all layers. . . . . 61
8.4 Efficiency, *ghost rate* and seed reconstruction efficiency with particles having hits in all layers or missing at most one hit. . . . . 62

# Acknowledges

The realisation of this project wouldn't have been possible without the support and help from the Data Science for the Digital Society (DS4DS) research group.

I would specially thank Doctor Xavier Vilasis Cardona for his dedicated time and my tutor Doctor Alessandro Camboni who has been extremely helpful during the entire realisation of the project.

My family, friends and distinctively to Xavi for their company, patience and advises during the moments of frustration and happiness after long working hours.

# Chapter 1

# Introduction

## 1.1 Standard model

The Standard Model of particle physics (SM) is the theory that describes all known elementary particles and the way they interact through three fundamental forces: the electromagnetic, weak and strong interactions.

The current formulation of the SM was finalised in the 70's and its validity keeps being supported by experiments decades later. It is also known that the Standard Model is not the ultimate theory of Nature, since it is not able to explain some phenomena related to the fundamental structure of matter and various cosmological observations. Nevertheless it is considered the most popular and supported theory describing the known universe.

In the following sections, a brief introduction to the main subatomic particle classification is given to provide the reader with a simplified insight to the main building blocks of the theory.

### 1.1.1 Fundamental particles

As mentioned above, the SM describes all known fundamental particles defined as physical objects for which no evidence has yet been observed of having an internal structure. They are classified according to a quantum number called *spin*, an intrinsic form of angular momentum, resulting in the following two fundamental classes of particles, as shown in Figure 1.1.

#### 1.1.1.1 Fermions

These fundamental class includes *quarks* and *leptons*. Fermions are considered the basic building blocks of matter and are classified according to the way they interact with the strong interaction, resulting in 12 types of elementary fermions: six quarks and six leptons.

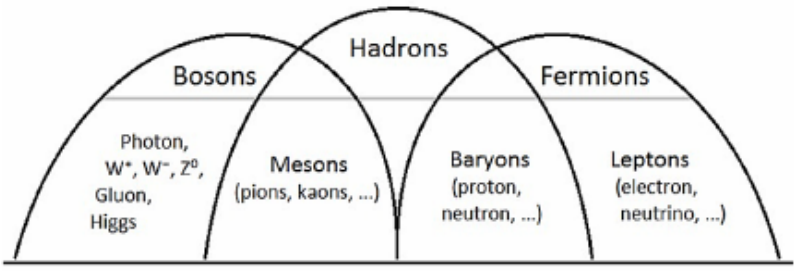



Figure 1.1: Classification of particles based on their spin.

Quarks cannot exist as isolated particles and combine to form composite particles known as *hadrons*. A combination of an odd number of quarks is a *baryon*, which is still a fermion. All of them have half-integer spin and have its own antiparticle.

#### 1.1.1.2 Bosons

The other fundamental class of particles are *bosons* with integer spin. Fundamental force carriers (photon, gluons, W and Z) belong to this class as well as the Higgs Boson, whose existence was predicted by the SM and confirmed by CERN in 2012. A combination of a quark and an antiquark gives an integer spin particle called *meson*.

## 1.2 Particle Accelerators

Particle accelerators are mainly used for basic research in particle physics. Charged particles are accelerated to very high speeds and contained in well-defined beams. In a circular collider two particle beams circulate in opposite directions and cross each other in one ore more interaction points where particle collisions occur.

Particles otherwise not observable in a normal environment are produced in a high energy particle collision where energy is converted into matter according to Einstein's famous equation $E = mc^2$. This provides physicists with a suitable environment to investigate and validate the theory and even discover new phenomenons in high energy particle physics.

### 1.2.1 Characteristics of the Accelerators

About 30.000 accelerators are operating world-wide but only 1% of them actually work with energies greater than 1 GeV. The need of highly advanced superconducting magnets which steer and accelerate particles is the main limitation to work with energies above this threshold, since this technology is extremely expensive and power consuming. As a comparison, the largest accelerator in the world, the *Large Hadron Collider* at CERN, consumes around

200 MW while working at 13 TeV collision energy, equivalent to one third of the power consumption of the entire city of Geneva where it is located.

The geometry of an accelerator is a crucial aspect: circular colliders are capable to deliver the highest collision energies compared to other geometries (*e.g.* linear colliders).

In a circular accelerator kinetic energy of particles increases at each turn until a desired energy threshold is reached, provided that the magnetic field of the superconducting magnets is enough to keep particle beams in a stable circular orbit.

It is also important to define the most significant characteristics of particle accelerators:

1. Type of accelerated particles: a variety of particles can be be accelerated as long as they have an electric charge capable to interact with the electromagnetic field.

2. Energy of collisions, measured in electronvolts.

3. Luminosity, which is the key indicator of the accelerator performance, defined as the number of collisions per unit area over time.

# Chapter 2

# The LHCb Experiment

## 2.1 LHC at CERN

As previously stated before, the LHC (Large Hadron Collider) is the largest and most powerful particle accelerator ever built. With a circumference of 26.7 kilometers at a depth ranging from 50 to 175 meters underground, it is located in a Swiss-France area near Geneva (Switzerland) and it is designed to accelerate counter-propagating proton beams colliding at energies up to 14 TeV. Although the designed energy has never been achieved, a collision energy of 13 TeV have been reached surpassing by far the second most powerful accelerator in the world [1].

This project is run by the European Organisation of Nuclear Research known as CERN (in French *Conseil Européen pour la Recherche Nucléaire*) with more than 10.000 collaborators around the world and hundreds of Universities and Laboratories involved. The achievements of this organisation have impact with significant breakthroughs both in particle physics and technology which have significantly affected our day-today lives, such as the World Wide Web.

The CERN complex is made up of six accelerators, the largest of which is the LHC. In order to achieve the incredible energy of 7 TeV per beam, they are pre-accelerated in several steps through the usage of the smaller ones, as explained in Figure 2.1.

Once the particles are injected into the LHC, a total of 16 radio-frequency cavities are placed along the ring in order to finally accelerate the proton beams to the nominal collision energy. In order to bend the beams and let them circulate through the ring, about 12.300 superconducting Niobium-Titanium dipole magnets are used with an additional 392 quadruples magnets to keep beams stable and focused. Moreover, in order to maintain the magnets at operational temperatures of 1,9K, approximately 96 tonnes of super-fluid helium-4 is needed making the LHC the largest cryogenic facility in the world at liquid helium temperature.

The internal ring structure contains two adjacent parallel beam-lines, each one containing a beam, usually protons. Beams travels in opposite directions around the ring at a speed close to the speed of light. Beam-lines are intersected in four Interaction Points (IPs) where

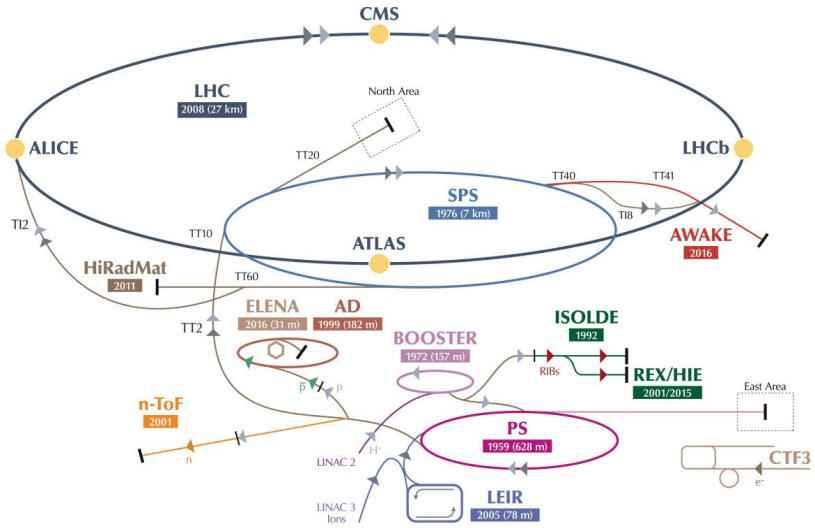



Figure 2.1: Schema of the CERN complex showing the different particle accelerators. In order to achieve 7 TeV per beam the process takes 5 steps. LINAC 2 starts accelerating particles up to 50 MeV, the BOOSTER then delivers particles at 1.4 GeV and injects them into the PS, which reaches the energy of 26 GeV. The following step is represented by the SPS which accelerates particles up to 450 GeV before finally injecting them into the LHC. Figure taken from [2].

beams are allowed to collide.

A particle detector is placed at each IP. They can be divided into two main categories: General-Purpose Detectors (GPDs) and dedicated physics experiments. GPDs at LHC are ATLAS (*A Toroidal LHC ApparatuS*) and CMS (*Compact Muon Solenoid*), which have been designed to study a wide variety of physical processes, with a main focus on the search and study of the Higgs Boson properties.

There are 2 main dedicated physics experiments operating at LHC: LHCb (*Large Hadron Collider beauty*) and ALICE (*A Large Ion Collider Experiment*). They have been designed to provide insights in particular scopes of physics such as the study of $c$ and $b$ hadron decay (LHCb), and the study of quark-gluon plasma (QGP) in heavy ion collisions (ALICE).

## 2.2 The LHCb detector

In order to help explain the different content of matter and antimatter in the Universe, physicists have designed and built the LHCb 5600-tonne detector. Its main purpose is to study the parameters of the CP violation[1] in the *beauty* quark sector.

[1]In quantum mechanics, a parity transformation (P) is the flip in the sign of one spatial coordinate. It can also be thought of as a test for chirality of a physical phenomenon. Charge conjugation (C) is a

With this in mind, the detector has been designed with a particular shape based on the characteristics of the $b\bar{b}$ production mechanisms found in $pp$ collisions. Since they are predominantly produced in the forward and backward directions with respect to the proton beam direction, it has been designed as a single-arm forward spectrometer with a polar angular coverage from 10 to 300 mrad (*milliradians*) in the horizontal and 250 mrad in the vertical plane, as shown in Figure 2.2.

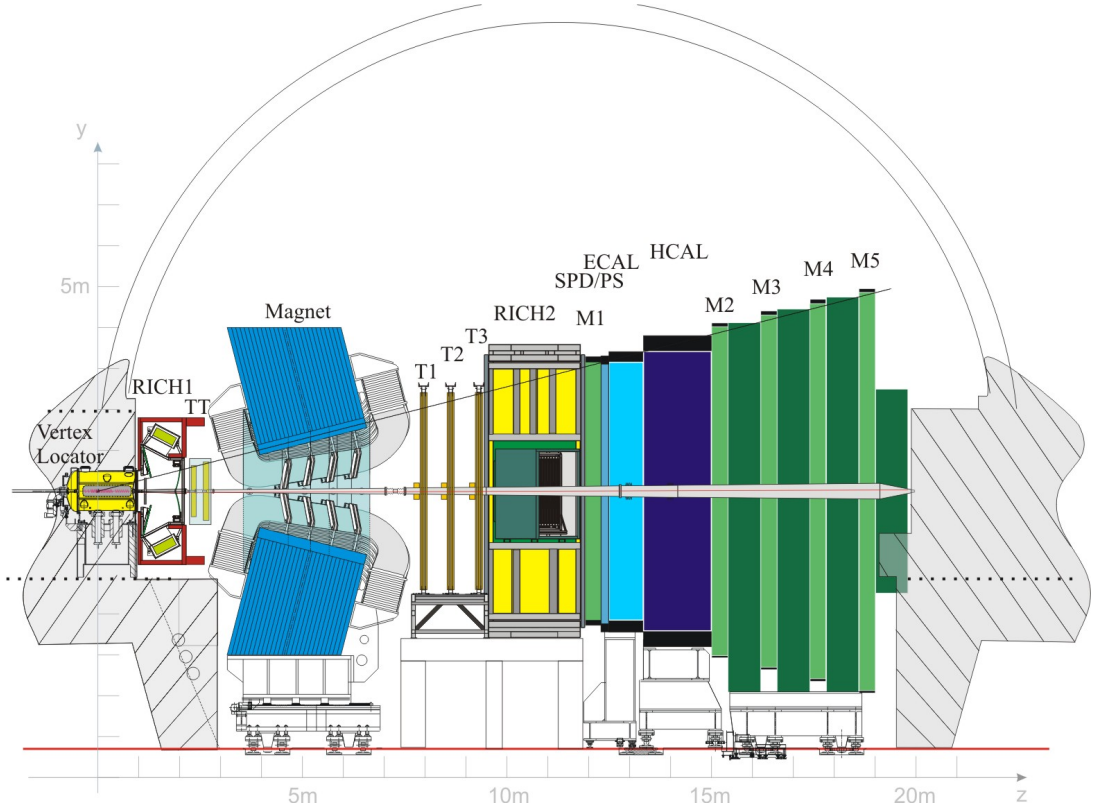



Figure 2.2: LHCb detector side view. Figure taken from [2].

The coordinate system in the LHCb is defined as follows:

- The origin of the system is the interaction point. The collision of two $p$ is defined as Primary Vertex (PV) and the decay point of a particle into new ones ia a Secondary Vertex (SV).
- X axis is horizontal, Y axis is perpendicular to the X axis and the beam line, Z axis points forward from the interaction point in the direction of the beam line.

transformation that switches all particles with their corresponding antiparticles, and thus changes the sign of all charges. The violation of the combination of C and P symmetries plays an important role in the attempts of cosmology to explain the dominance of matter over antimatter in the present Universe.

Charged particles generated in collisions are bent in the Y direction by the dipole magnet located after the interaction point, as can be seen in Figure 2.2.

In particle physics, tracking is the process of reconstructing the trajectory (or *track*) of electrically charged particles in a particle detector known as a *tracker*. The particles entering such a tracker leave a precise record of their passage through the device by interacting with suitably constructed components and materials. In the following an overview of LHCb sub-detectors is given, with a particular focus on the tracking system.

## 2.3 LHCb systems

The detector is made of several sub-detectors aimed at performing different tasks, mainly *tracking* and particle identification.

### 2.3.1 The tracking system

The tracking system consists of the VErtex LOcator (VELO), situated around the interaction region, and four planar tracking stations: the Tracker Turicensis (TT) upstream of the dipole magnet, and the tracking stations T1-T3 downstream of the magnet.

#### 2.3.1.1 VELO

It is the closest sub-detector to the beam interaction point. Its main goal is to locate PVs, assign tracks to the correct PV and evaluate the impact parameter of each track (defined as the distance of closest approach of a track with respect to a PV). This task is crucial since the PV resolution is fundamental to precisely measure CP parameters.

#### 2.3.1.2 Tracker Turicensis

The next Tracking system is the Tracker Turicensis, located upstream of the dipole magnet where fringe field is present. It consists of four different layers oriented in a stereo configuration, defined as $x$-$u$-$v$-$x$, providing 3D information for the track reconstruction thanks to the $u/v$ layer rotation of -/+ 5 degrees around the $z$ axis.

#### 2.3.1.3 Dipole magnet

The following tracking element is the 1.600 tons warm dipole magnet. It allows to measure the charge and the momentum of particles by bending their trajectory in the Y-Z plane. This information is later used in the track fit and local-parametrisations in the pattern recognition algorithms.

#### 2.3.1.4 Inner and Outer Tracker

The IT and the OT are located downstream the dipole magnet covering the inner and outer region of the tracking stations respectively. They consist of three stations (T1, T2 and T3) where each layer structure is similar to the TT, with an *x-u-v-x* configuration. They provide 3D information of tracks passing the magnetic field.

#### 2.3.1.5 Track types

Several types of tracks are defined in the LHCb tracking system, as can be seen in Figure 2.3. They are classified according to the hit content in the three different tracking sub-detectors.

- *Velo tracks*. As the name states, they are solely composed by hits in the VELO sub-detector and are used for vertexing and as starting seeds for tracks found in the rest of the detector.

- *Upstream tracks*. These are composed by hits in the VELO and the TT. The name is due to the fact that these tracks have hits only in the sub-detectors upstream of the magnet. The patter recognition algorithm used to reconstruct these tracks is called `VeloUT` algorithm.

- *T-tracks*. These tracks are reconstructed only using hits from the T stations downstream of the magnet. In this case, the algorithm used in the reconstruction is called `PadSeeding`.

- *Downstream tracks* are mostly associated to long-lived particles which fly a significant distance in the VELO before finally decaying into charged particles. As a result, these are composed of hits in the TT and the T-stations only. In this case, the algorithm used to reconstruct these tracks is called `PrLongLivedTracking`.

- *Long tracks* are the most important track type used in the LHCb, leaving hits in the VELO, UT and T-stations. The algorithms aimed at reconstructing these type of tracks are `PrForwardTracking` and `PrMatchNN`.

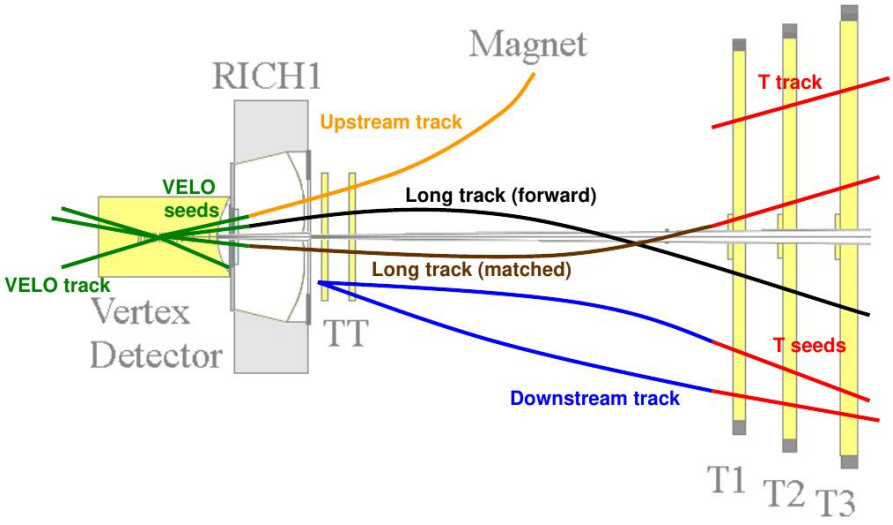



Figure 2.3: Track type definition in the LHCb detector.

#### 2.3.1.6 Track reconstruction

Track reconstruction at LHCb follows a two-fold sequence. First, pattern recognition algorithms are executed combining individual measurements in various tracking systems in order to form track candidates, then a Kalman filter fitting technique is used to determine track parameters by estimating a joint probability distribution over these parameters for each time-frame.

In order to validate these algorithms before being finally deployed, the reconstruction efficiency is measured for each reconstruction algorithm. By using the truth information available in simulated samples produced by Monte Carlo simulations, these are tested against almost real world data in order to determine the efficiency of the different approaches.

Another important measure regarding the efficiency of tracking algorithms is the amount of *ghost tracks*, *i.e.* wrongly constructed tracks generated by these algorithms.

### 2.3.2 LHCb Particle Identification system

The tracking of particles in the LHCb is accompanied with an extensive and complex identification system composed of three detectors, RICH (1 and 2) the calorimeter system and finally the muon system.

#### 2.3.2.1 RICH detectors

The Ring Imaging Cherenkov detectors are positioned on both sides of the dipole magnet, in order to intercept particles flying at different speeds and angles.

They use Cherenkov light produced by some traversing particles to identify different types of hadrons in different locations. The Cherenkov radiation phenomenon occurs when a charged particles passes through a certain medium faster than light does, this situation is similar to the one happening with aircrafts breaking the sound barrier, emitting a cone of light which is then reflected by spherical and flat mirrors outside the LHCb acceptance to a matrix of Hybrid Photon Detectors, which are responsible of determining the shape of the cone of light, as it directly depends on the particle velocity. The combination of data extracted from the HPDs and the track momentum makes it is possible to assign mass to particles, thus identifying them.

#### 2.3.2.2 Calorimeter system

Following the RICh detectors is the Calorimeter system. The main purpose of this system is to trigger on electrons, photons and hadrons. It is composed of multiples sub-detectors whose main objective is to stop particles as they pass through the detector, measuring the amount of energy they lose. The sub-detectors composing this system are: Scintillating Pad Detector (SPD), Preshower (PS), Electromagnetic Calorimeter (ECAL) and the Hadronic Calorimeter (HCAL).

The SPD purpose is to determine whether particles hitting are charged or neutral, while the PS indicates the electromagnetic character of them. The ECAL is then responsible for measuring the energy of lighter particles, such as protons and electrons, while the HCAL is responsible for sampling the energy of protons, neutrons and other particles containing quarks.

These two calorimeters provide the main way of identifying particles possessing no electrical charge, such as photons and neutrons, while the SPD and PS provide insight on charged particles.

#### 2.3.2.3 Muon Stations

Finally the Muon identification plays an important role in the detector, since muons are present as final states in several CP-violating B meson decays.

This system is composed of five different rectangular stations, divided internally into four different regions, which are used to provide space point measurements of the tracks, producing binary identification with yes/no results.

### 2.3.3 Trigger System

When the LHCb is at full operation, the frequency of bunch crossings, the collision of two proton bunches containing thousands of protons each, is 40MHz, but only about 10MHz of events will actually have some particles inside the area of acceptance of the detector. Moreover, as the LHCb is only interested in specific B meson decays, the amount of events worth to store and analyse is reduced to few Hz.

Another key factor to take into account when operating at nominal conditions is the event rate at which the computing capacity is able to record data, which is about 5kHz.

With this considerations in mind, the trigger strategy used during the Run 1 data taking period is summarised in the Figure 2.4.

The earliest stage is the Hardware Trigger or L0 Trigger. Implemented in custom electronics, makes use of the fact that particles from a B decay have higher transverse momentum than particles coming from the primary $pp$ interaction. It is able to reduce the rate from 40MHz to about 1 Mhz.

The second and third trigger stages known as High Level Triggers (HLT1 and HLT2) use a computing farm with thousands of cores with the full detector information. The first one reduces from the 1 Ghz input rate to 80 kHz, being the rate in which the full event reconstruction is able to be performed. After this real-time reconstruction, the trigger rate is later reduced to 5kHz in order to store data to disk for the offline data analysis.

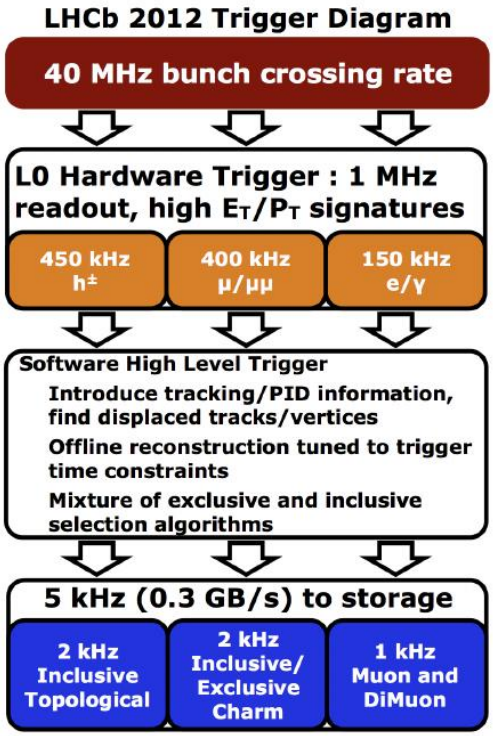



Figure 2.4: Trigger System diagram from the 2012 data taking period (Run 1)

## 2.4 LHCb Upgrade

LHC running behaviour does not follow a continuous approach, but rather separated periods known as *Runs* lasting approximately 3 years in which the collider is functioning at full capacity. Between these, the periods known as *Long Shutdowns* separate the functional cycles, which are aimed at performing upgrades and improvements in each detector to extend the physical reach of each experiment.

There has been two operational runs up to now, starting at 2009 to 2013 where the accelerator was able to run with energies of 4 TeV per beam allowing the discovery of the Higgs boson at the mid of 2012.

The upgrade which took place between 2013 and 2015 was aimed at enabling collisions at 14 TeV by enhancing detectors and pre-accelerators together with other major improvements. The collision energy was finally reduced at 6.5 TeV per beam and the Run II started in March of 2015 until 2018. The number of collected $pp$ collisions exceeded by far the ones recorded during the first run and with higher energy per collision.

At the moment of writing this document LHC is in the *Long Shutdown 2*, a period of two years in which improvements will be implemented on all the detectors, specially at the LHCb in which two key upgrades are planned:

- The current full read-out of the front-end electronics (currently limited by the L0 Trigger at 1.1 Mhz) will be replaced by a new 40 MHz trigger system, meaning that the LHCb will be able to feed events every 25 ns into the data acquisition farm, applying a full software trigger to every single bunch crossing.

- Due to the increase in luminosity, the average number of visible interactions per bunch crossing is going to be around 7.6. This will force a replacement in the front-end electronics and the sensitive elements of the detectors, the most noticeable being the silicon tracking stations T1,2,3 being replaced by a single homogeneous detector based on scintillating fibres called *SciFi*.

### 2.4.1 The tracking system

The tracking system will receive significant changes. The VELO will be replaced with a new hybrid pixel sensor detector located at 5.1 mm from the beam-pipe. The tracker stations upstream and downstream of the dipole magnet will also be completely substituted: The TT will be replaced by four layers of silicon detectors increasing the granularity and acceptance in the central region. The IT and UT will be replace by the new scintillating fibre tracker.

### 2.4.2 Particles Identification

Particle identification sub-system will also be upgraded with a series of partial upgrades to enable it to work in the new conditions.

The optical system of the RICH will be upgraded to cope with the higher occupancy of the spherical mirrors and the hybrid photon detectors replaced with Multi Anode PMTs, enabling the 40 MHz external read-out.

The calorimeter system doe not require a complete rebuilding: SPD and PS will be removed and the read-out electronics will be fully replaced. Finally the Muon system already exceeds the performance of the required specifications for this new environment, so it will receive minor upgrades.

### 2.4.3 Trigger

One of the main aspects in this LHCb upgrade will be the new trigger system. The main key features of this upgrade are: the read-out at 40MHz of the whole detector, full software trigger (reconstruction algorithms will run at collision rate) and the real-time alignment and calibration (which will allow to perform physics analysis directly from the trigger output and up to 90% in storage savings thanks to the efficient removal of useless information to store).

A comparison schematic between the multiple trigger strategies adopted during the pasts runs and the actual can be found in the Figure 2.5.

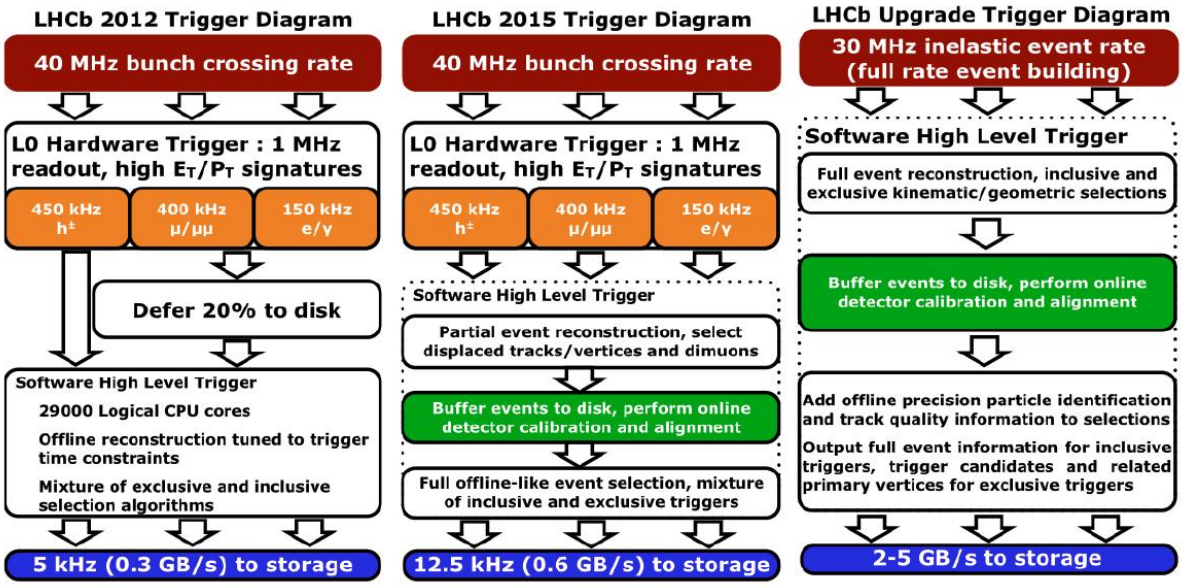



Figure 2.5: Trigger strategies adopted from the Run 1 to the upcoming Run 3. Taken from [13].

# Chapter 3

# The Scintillating Fibre Tracker

## 3.1 General description

The *SciFi* tracker is the successor of the current Outer and Inner Tracker. It has been optimised to work with the increased luminosity expected in the Run 3. It is composed by three different tracking stations, T1, T2 and T3 as shown in Figure 3.1.

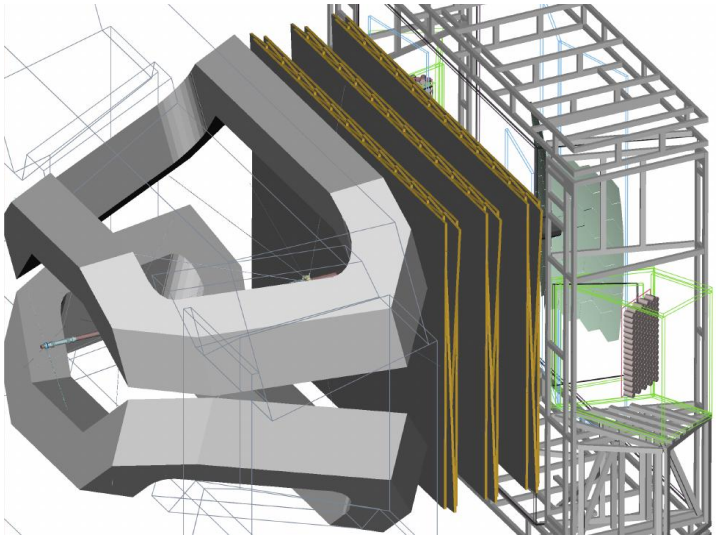


Figure 3.1: Schematic 3D view of the *Scintillating Fibre Tracker* located between the dipole magnet and the RICH2 system. Figure taken from [2].

Each station is composed by 4 different layers (X-U-V-X) separated by a 50 mm air gap. The first and last layer of each station are vertically oriented giving the $x$ position of the tracks for a given $z$ *layer*. The inner layers form a +/- 5 degree angle with the $y$ axis and provide the $u$ and $v$ stereo coordinates, used to extract the information on the $y$-$z$ plane motion of particles [17].

Each detection layer (shown in Fig. 3.2) contain a total of 12 modules (except for the T1, which has 10 modules). Each of these modules contains 8 fibre mats with six stacked layers of 2.5 m long scintillating fibres. Modules are separated by a mirror in order to increase the light yield of the fibres.

Silicon PhotoMultipliers (*SiPM*) are located at the edges of these modules in order to collect the light produced and transported in the fibres. They transfer data to the Front-End electronics responsible for processing the *SiPM* output.

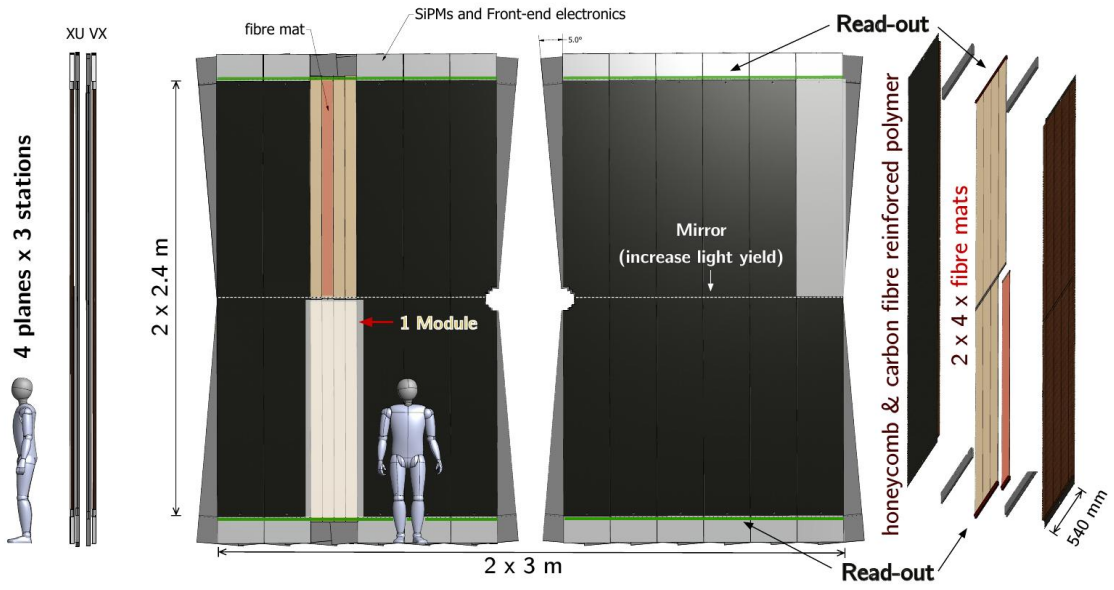



Figure 3.2: Schematic front and side view of the *Scintillating Fibre Tracker* located between the dipole magnet and the RICH2 system.

## 3.2 Scintillating Fibres

The active material used in the SciFi detector is plastic scintillating fibres with a cylindrical shape made of a core surrounded by two claddings as shown in 3.3. This fibres have a total trapping efficiency of 10.7%, as only 30 photons out of 300 are finally fully captured and propagated through, which are later detected by the *SiPM*.

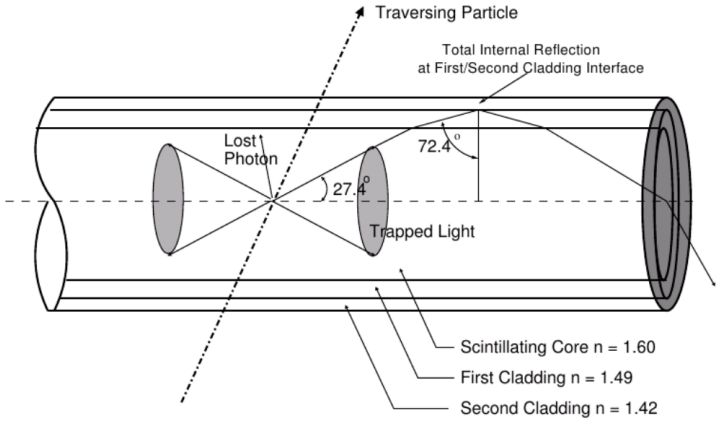



Figure 3.3: Schematic of the plastic scintillating fibre. Particles traversing the fibre produce light in the core, only the light produced inside the trapping angle is propagated within the fibre through total internal reflection. The claddings are arranged by decreasing indices of refraction to increase the amount of trapped light. Figure taken from [2].

## 3.3 Silicon Photomultipliers (SiPM)

Light traversing the fibres ends up being detected by the Silicon Photomultipliers, which are made of custom 128-channel arrays composed by a rectangular matrix of pixels responsible for the photon counting, as shown in figure 3.4.

SiPMs provide high gain, low operational voltage, small granularity, fast response, insensitivity to magnetic fields and the single photon counting properties These characteristics are the key to opt for the adoption of this technology in order to detect traversing particles through the SciFi sub-detector.

## 3.4 Read-out electronics

The SciFi read-out electronics is constituted by front-end and back-end electronics. The FE is responsible for the digitisation and clustering of the SiPM signal output, the BE performs the data processing.

The FE electronics is connected directly to the SiPMs arrays with a custom designed ASIC (*Application-specific integrated circuit*) called PACIFIC. This mixed-signal ASIC is responsible for the amplification, shaping and charge interaction within the limited time window of 25 `ns` together with the conversion of the analogue to digital value.

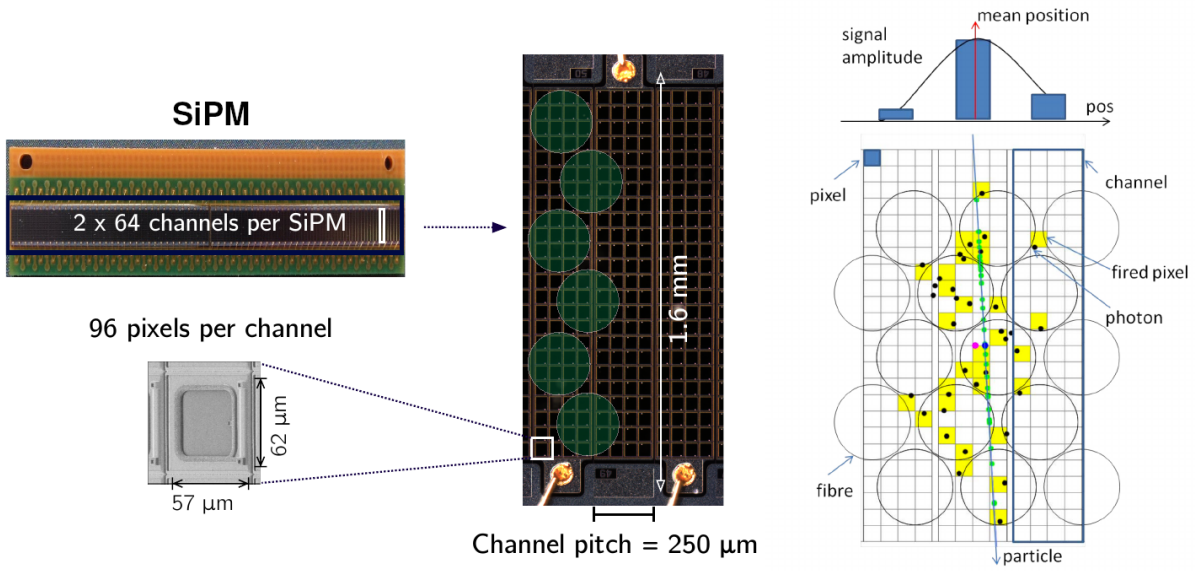



Figure 3.4: SiPMs have a total of 128 read-out channels, coupled to a total of six layers of stacked fibres. Each of these read-out channels are composed of a rectangular matrix of pixels, responsible for performing the photon counting. The illustration on the right shows how a position for a given particle is computed, since the position is obtained by the average charge collected from each pixel within the read-out channel. Figure taken from [14].

Due to the huge amount of data, digital values are then sent to FPGAs which perform clustering of the data, several channels are merged together into a single measurement thus reducing the output bandwidth. GigaBit trans-receivers are used to transfer the remaining high bandwidth clustering data into the BE, responsible to execute the pattern recognition algorithms used for track reconstruction.

## 3.5 Patter recognition Algorithms

Once a *pp* collision occurs, particles travel through the tracking detectors releasing small amounts of energy in the active parts of each sub-detector. This energy is converted into electronic signals producing hits, later used by the Pattern recognition algorithms, who are responsible for connecting these in order to recognise tracks produced by the particles trajectories.

Multiple pattern recognition algorithms are implemented in the LHCb experiment, each one with a focus on a particular type of track. This is due to the fact that classification of tracks is based on the absence or presence of hits in each tracking sub-detector. A simple explanation for each pattern recognition algorithm is given below.

- **PrPixelTracking**
  Given the fact the magnetic field in the VELO region is almost negligible, the `PrPixelTracking` algorithm produces only straight tracks in both $x$-$z$ and $y$-$z$ projections. The hits used to create this tracks are basically created from the read-out of the sub-detector pixels, and at least three of them must be used to create a track candidate. This algorithm creates `VELO` seeds, who are later used in Upstream tracks and Long tracks, as shown in Figure 2.3.

- **PrVeloUT**
  `PrVeloUT` patter recognition algorithm main purpose is to reconstruct `Upstream` tracks, both the ones with low momentum which are bent by the dipole magnet outside the LHCb acceptance, and the high $p_T$ ones. In order to perform this, tracks found by the VELO `PrPixelTracking` are matched with hits in the UT, creating the union of UT hits with VELO tracks.

- **PrForwardTracking** Long tracks are mainly reconstructed by two algorithms, being `PrForwardTracking` one of them. This algorithm is based on a Hough transformation approach and uses VELO or upstream tracks as input searching for compatible hits in the T-stations, or in the SciFi in the upgraded configuration. By projecting each seed track trajectory into the stations, a track candidate is created for each Hough cluster and only the fittest is selected.

- **PrHybridSeeding** This standalone track reconstruction algorithm uses hits solely from the T-stations. This algorithm is described in detail in chapter 5.

- **PrLongLivedTracking** The main purpose of this downstream tracking algorithm is to reconstruct tracks from long-living particles using seeds from the `PrHybridSeeding`, and backward propagating these to the UT central position, selecting track candidates based on the union of UT hits and T-tracks.

  The main reason is due to the fact that long-lived particles decay outside the VELO and their products leave hits only in the UT and SciFi.

- **PrMatchNN** The `PrMatchNN` acts as a complementary to the forward tracking algorithm. It uses VELO and T-tracks as input and tries to match them in order to create *long* tracks [14].

## 3.6 Performance indicators

In order to compare and compute the tracking efficiency of patter recognition algorithm, the performance indicators must be the same for each of these algorithms. These indicators can be determined using simulation studies by comparing the number of tracks the algorithm is able to find (*reconstructed* tracks) with the maximum number of tracks that it is possible to

find (*reconstructive* tracks). Moreover, tracking performances are also determined depending on the track type, such as `Long`, `Downstream`, `Upstream`, `Velo` or `T-track` [14].

A specific criteria for each of the sub-detectors is established to determine the amount of hits a particle is said to be *reconstructible*. In the SciFi, a particle is *reconstructible* if there are at least one *x-layer* and *stereo* hit in each of the three tracking stations.

*Reconstructed* tracks are associated to *reconstructible* particles by checking the amount of hits they share, which must be equal or more than 70%. The following performance indicators are used in order to compare and evaluate algorithms.

- **Tracking efficiency**. One of the most important indicators, it is defined as the ration between the amount of *reconstructed* and *matched* tracks with respect to the total amount of *reconstructible* tracks.

$$Tracking\ efficiency = \frac{reconstructed\ \&\ matched}{reconstructible}$$

- **Ghost rate**. It is the amount of all *reconstructed* tracks having less than a 70% of hits matching with a *Monte Carlo* particle with respect to the total amount of tracks found by the pattern recognition algorithm.

$$ghost\ rate = \frac{reconstructed \text{ not } matched}{reconstructed}$$

- **Hit purity**. For the cases where a *reconstructed* track is *matched* to a simulated particle, it is the fraction of hits the *reconstructed* track has in common with the *matched* track.

$$hit\ purity = \frac{\text{hits shared between the } matched \text{ and the } reconstructed \text{ track}}{\text{amount of hits of the } reconstructed \text{ track}}$$

- **Hit efficiency**. It expresses the efficiency of the pattern recognition algorithm to pick up hits on *reconstructed* tracks which are expected to belong to the associated particle.

$$hit\ purity = \frac{\text{hits shared between the } matched \text{ and the } reconstructed \text{ track}}{\text{amount of hits of the } matched \text{ particle}}$$

# Chapter 4

# Tracking data

Before entering into the tracking algorithms, in this chapter details will be given on the structure and meaning of the detector simulated data used in this work, in particular of the SciFi subdetector.

The Monte Carlo simulation is described in section 4.1. Sections 4.2.1 and 4.2.2 describe the available types of hits used in the reconstruction of tracks by the pattern recognition algorithms.

## 4.1 Monte Carlo Simulation

The present project works on a future upgrade of the detector, thus real data are not available at the time of writing this document. Preliminary studies are possible making use of Monte Carlo simulation techniques.

Monte Carlo method allows to simulate the real world detector and its response to a given $pp$ collision. It is then important to have a highly reliable generator of the physical event and an accurate description of the whole detector in order to test the algorithms. As a result, the MC *truth* is used to create and test the performance of the algorithms for tracking and particle identification [15].

In order to simulate a particular event in the LHCb, the simulation flow is mainly separated in two different steps, the first being the *Gauss Simulation* and the subsequent *Boole Digitisation* [3]. A schematic view of the whole process can be found in Figure 4.1.

*Gauss Simulation* main purpose is to decide which particles are going to be created for the particular event based on probabilities given by the Standard Model. This process is separated in three sub-steps:

1. Generate the primary collision using a particular framework depending on the needs for the simulation. For a general-purpose collision *Pythia* framework is used.

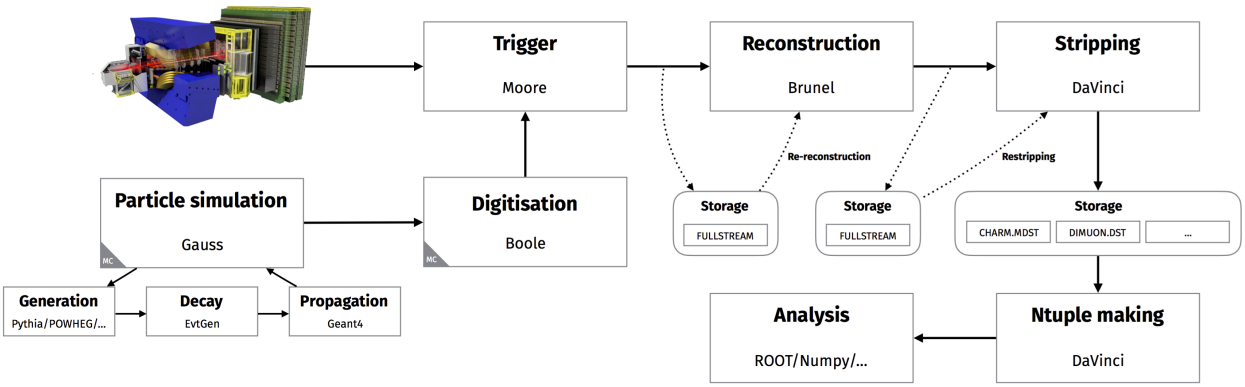



Figure 4.1: Schematic of the LHCb execution flow from both the real detector and the MC simulation.

2. Once particles are generated their decay is simulated by the `EvtGen` generator specialised in b-hadron decays.

3. Finally, the *GEANT4* C++ toolkit is used to propagate particles through the simulated detector. Each particle is moved in steps from few microns to cm computing the interaction with the material it is traversing, as well as the electric and magnetic fields affecting its motion. This leads to the need for an accurate modelling of each part of the detector, from its sensitive material to the glue sticking together each module for a given T-station.

The *Boole Digitalisation* is executed sequentially after the *Gauss Simulation*. Its main purpose is to transform the simulated physical data into the real detector output data format, creating an instance of the `Event model` object. This data structure is later fed into the trigger as it would occur with the real detector following the normal execution flow, as shown in Figure 3.4.

## 4.2 Dataset description

When developing track reconstruction algorithms, there are two different hit types used to construct and test a given algorithm: *MCHits* and *PrHits*, the former being the true hit point of a particle on a given sub-detector layer and the latter a reconstructed hit.

### 4.2.1 MCHits

These represent the precise particle hit in each of the sub-detectors without taking into account the limitations each of them have to resolve positions, e.g. the exact $y$ position

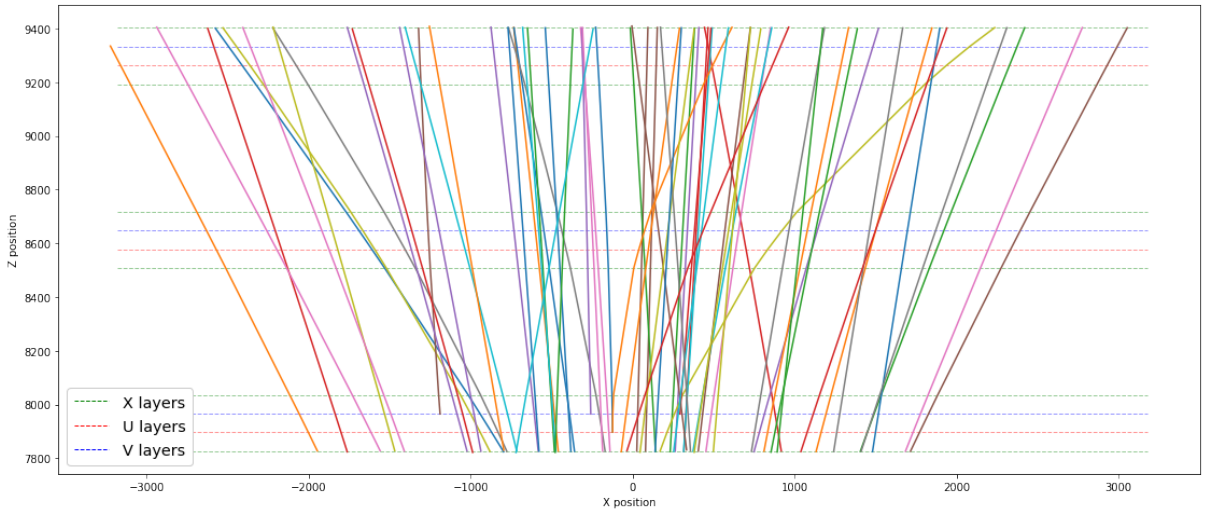
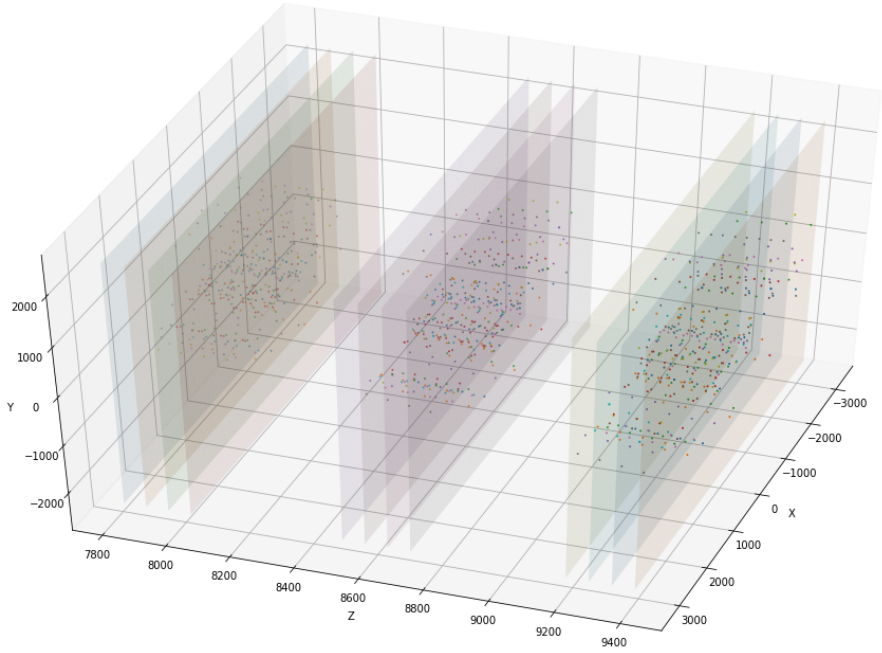


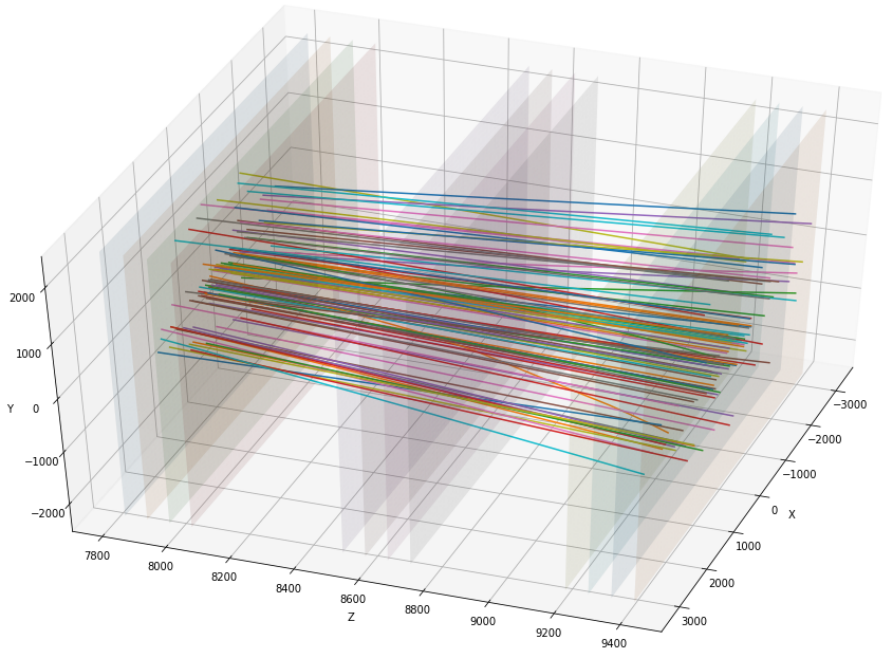

Figure 4.2: 2D view of tracks using *MCHits* together with a visual representation of *X*-*U*-*V*-*X* layers.

for a given hit in a T-station. As a result, they are mainly used for representation and performance indicators rather than being directly used in developing track reconstruction algorithms.

As shown in figures 4.2, 4.3 and 4.4, a perfect representation for the track of a particle can be recreated using the Monte Carlo *truth*, as the exact $x$, $y$ and $z$ positions can be retrieved for each hit of a particle going through the SciFi.

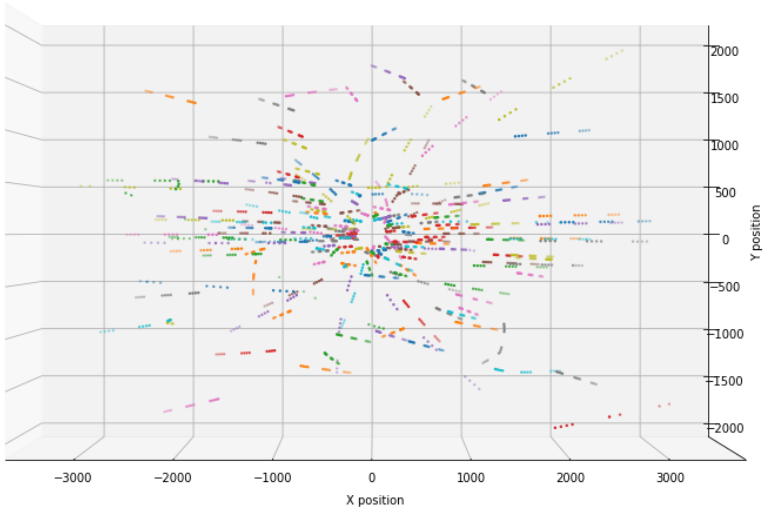



(a) Particle *MChits* from each layer.

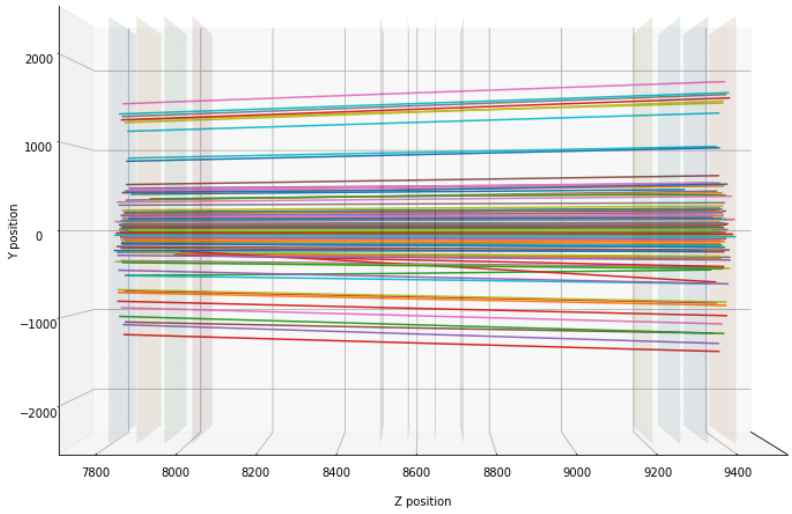



(b) Particle tracks constructed using the *MCHits* of each particle

Figure 4.3: Three dimensional view of 100 particles through the SciFi sub-detector by using the *monte carlo truth* together with a visual representation of the *X-U-V-X* layers.




(a) Front view of *MCHits* from the first layer




(b) Side view of reconstructed tracks sing *MCHits* from each particle.

Figure 4.4: View of particle trajectories from the first layer looking downstream the experiment can be found in figure ($a$). In fig. ($b$) it is possible to see the small impact the dipole magnet has over the SciFi stations.

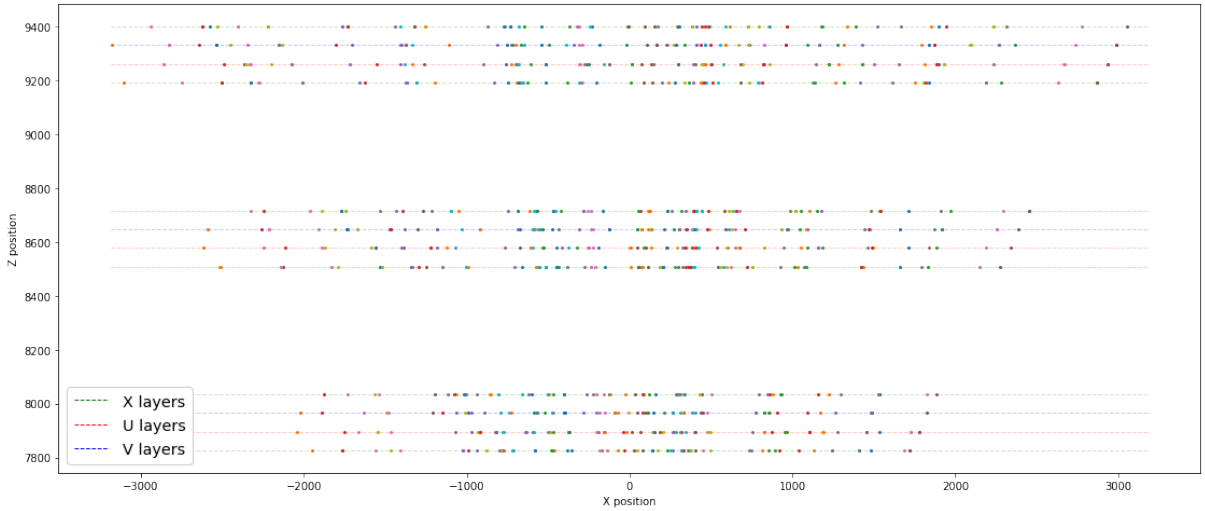
### 4.2.2 PrHits

Unlike *MCHits* which are the exact hits of a given particle, *PrHits* represent a real approximation of the SciFi output in a real event. They are the processed positions obtained from the read-out electronics, explained in detail in sec. 3.4.

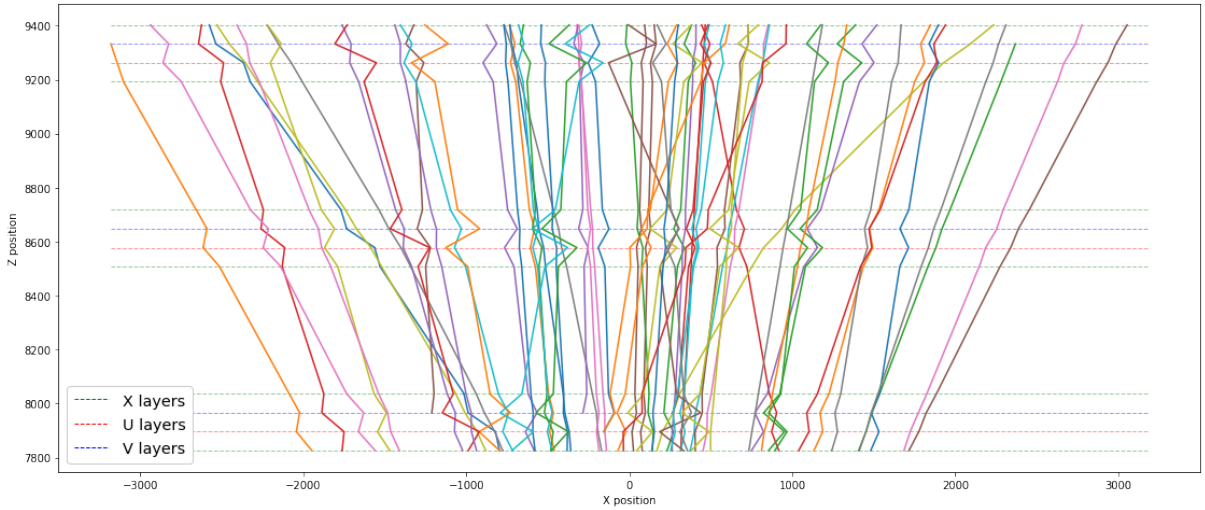
The following attributes are available for each *PrHit*:

- **Event**: unique identifier of an event (bunch crossing).
- **Hit_LHCbID**: identifier of each hit in the same event. It is assigned according to the $x$ and $z$ position of the hit.
- **Hit_Xat0**: The $x$ position in $mm$.
- **Hit_YMax**: Upper $y$ range in $mm$.
- **Hit_YMin**: Lower $y$ range in $mm$.
- **Hit_Zat0**: Exact $z$ position, which takes 12 different values (one $z$ position per station layer), in $mm$.
- **Hit_isX**: Boolean, 0 or 1 defining if the current hit is at a $X$ station or not.
- **Hit_planeCode**: Layers are enumerated from 0 - 11, this attribute states in which layer the hit is.
- **Hit_w**: Width obtained from the cluster size.
- **Hit_zone**: Boolean, 0 in case the hit is in the lower part of the layer or 1 in case it isn't.

As it can be seen from the above attributes, only a 2D view of each hit can be obtained as we are unable to determine the $y$ position directly from the raw data of one hit. The *Hit_YMax* and *Hit_YMin* span an entire zone, leaving to the following visual representation of *PrHits* in fig. 4.2.2.




(a) Two dimensional view of *PrHits*.




(b) Track reconstruction using *PrHits* matched to a given particle.

Figure 4.5: Visual representation of *PrHits* associated to 100 particles. As can be seen, when reconstructing tracks using *PrHits*, those in $U$ and $V$ layers seem deviated from the true particle trajectory, because of the $\pm 5$ degree difference between $X$ and stereo layers.

# Chapter 5

# The Hybrid Seeding algorithm

## 5.1 Overview

The Hybrid Seeding is a stand-alone algorithm created to reconstruct tracks using data exclusively from the SciFi detector. It is the successor of the algorithm documented in the Upgrade Technical Design Report, with significant improvements in all the important performance indicators found in pattern recognition algorithms for track reconstruction. It is described in detail in [14] and resumed in this chapter.

The algorithm follows multiples sequences of 8 separated steps after finally reconstructing all hits into LHCb track objects, based on what are defined as *cases*. This *cases* are used as a way to separate particles with different momentum, due to their difference in track structure. Moreover, as there are less than 0.01% of tracks trespassing the middle section, hit search is bounded in the upper/lower part.

The eight steps are described in detail in the following sections according to their order in the algorithm flow, as can be seen in fig. 5.1.

## 5.2 Find x-z projections

The first part of this algorithm is devoted to create $x$-$z$ tracks using hits from the *X-layers* from all the different stations. This produces three different steps visually represented in Fig. 5.2 and described below.

1. **Two-hit combination**: A hit is selected from one *X-layer* in T1 and another in T3, depending on the case shown in table 5.1, the first or last *X-layer* per station is selected.

2. **Three-hit combination**: After selecting a possible two-hit combination a third hit is searched, defining a track structure of a parabola.

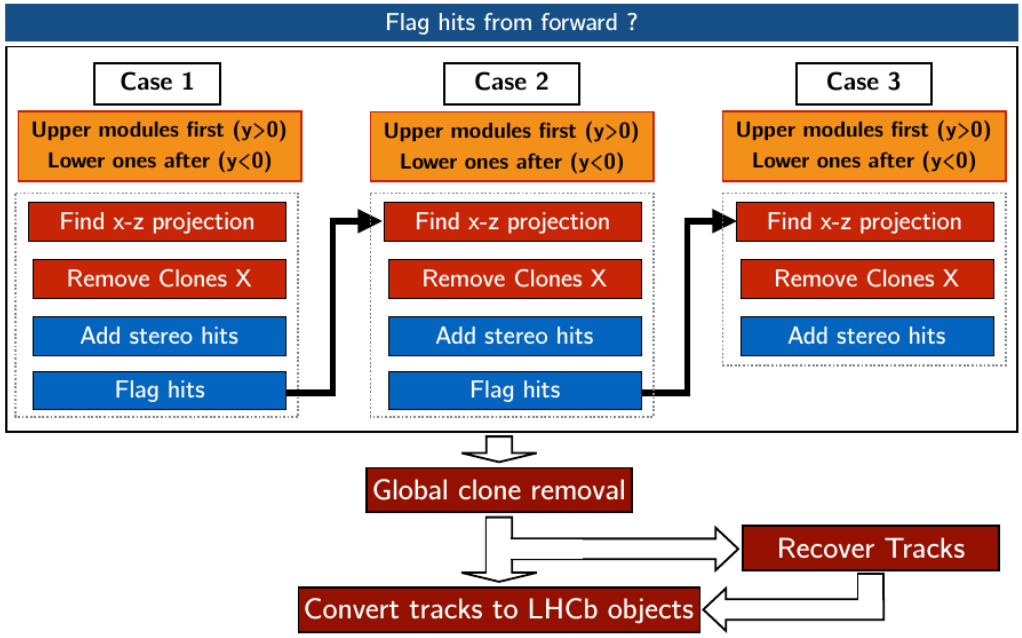



Figure 5.1: Overview of the Hybrid Seeding algorithm structure. Figure taken from [14].

| Cases | T1 X-layer | T3 X-layer |
|---|---|---|
| 0 | *T1 - 1* | *T3 - 2* |
| 1 | *T1 - 2* | *T3 - 1* |
| 2 | *T1 - 1* | *T3 - 1* |

Table 5.1: Selection of the *X-layer* in T1 and T3 station based on the case.

3. **Complete and fit**: The selected three-hit combinations are then completed adding hits in the remaining *X-layers* and filtered according to the number of hits combined together with a track-fit procedure.

## 5.3 x-z clone killing

After selecting all tracks from the $x$-$z$ projection, a clone killing process is performed, which impacts positively in reducing the final ghost rate. It is performed by comparing tracks which are close to one another, and selecting only one of them.

The clone killing procedure is based on the assumption that a track containing six hits is a well constrained track and more likely to be a good one, while tracks with a lower number of hits are more likely to be a ghost.

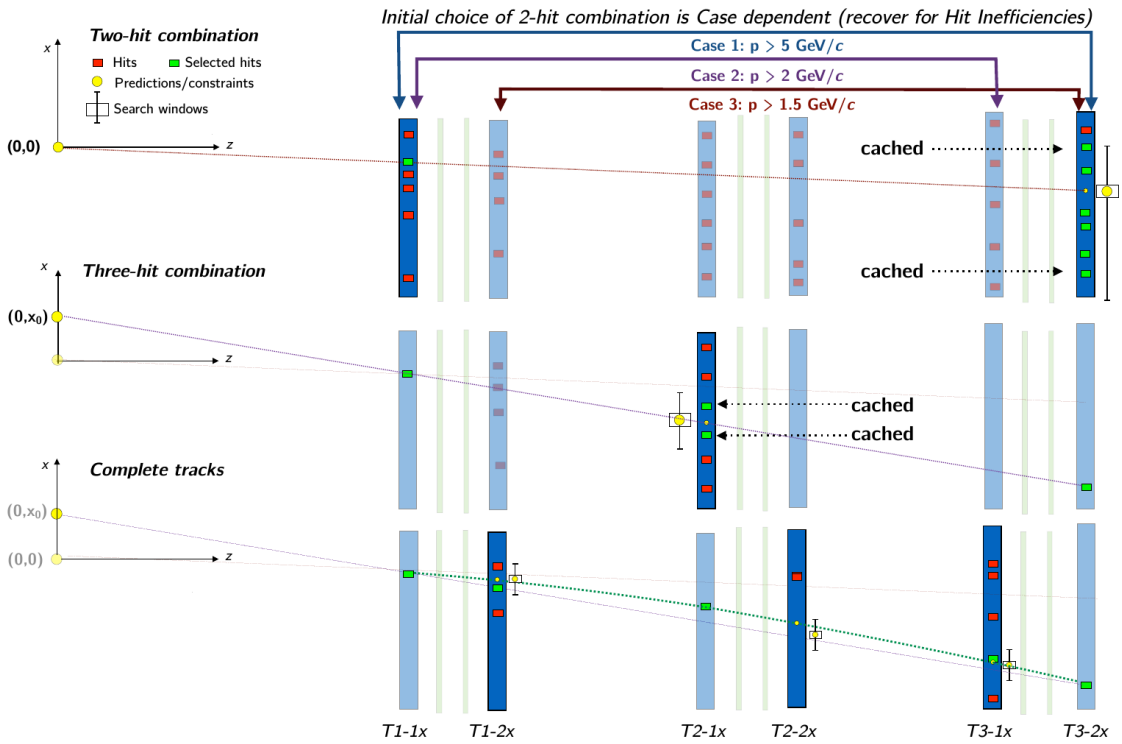



Figure 5.2: Visual representation of the first three steps the Hybrid Seeding Algorithms performs. Figure taken from [14].

## 5.4 Stereo Hits

Vertical coordinate $y$ is added to track candidates in the $x$-$z$ projection using $u/v$ layers. The addition of stereo hits follows these steps:

- **Collect compatible $u/v$ hits for each $x$-$z$ projection**. This is performed by predicting the $x$ position at the $u/v$ layers and computing the difference between the prediction and the measurement. This difference is used to obtain the angle $\beta$, as shown in the following formula. It is later used to know the $y$ measurement for a given hit, since it represents the angle formed in the $y$-$z$ plane by the projection from the (0,0) position to the $U/V$ layer, as shown in fig. 5.3.

$$\beta = \frac{x^{hit} - x^{predicted}}{\alpha_{stereo} \cdot z^{hit}}$$

  All the hits compatible in $y$ with respect to the seed $x$-$z$ projection are collected and stored in a container.

- **Hough-like Cluster search**. Based on the values computed in the previous step, all groups of hits sharing a similar value of $\beta$ define a potential line candidate to

be attached to the $x$-$z$ projection. The selection of these groups is done by using a `Hough-like Cluster`.

- **Hough-like Cluster to line candidate conversion**. These clusters undergo a selection procedure aiming at generating a list of straight line candidates made of single $U/V$ hits per layer. For each of these line candidates a fast $y$-$z$ projection fit is performed.

- **Full track fit**. After the stereo hit line candidates are selected, they are merged with the $x$-$z$ projections to create the final track candidates. These full tracks are then fitted and selected depending on their $y$ position.

- **Selection of tracks from same seed $x$-$z$ projection**. Only the best track candidate according to the number of fired layers is promoted as a final track candidate among all the candidates produced from the same seed $x$-$z$ projection.

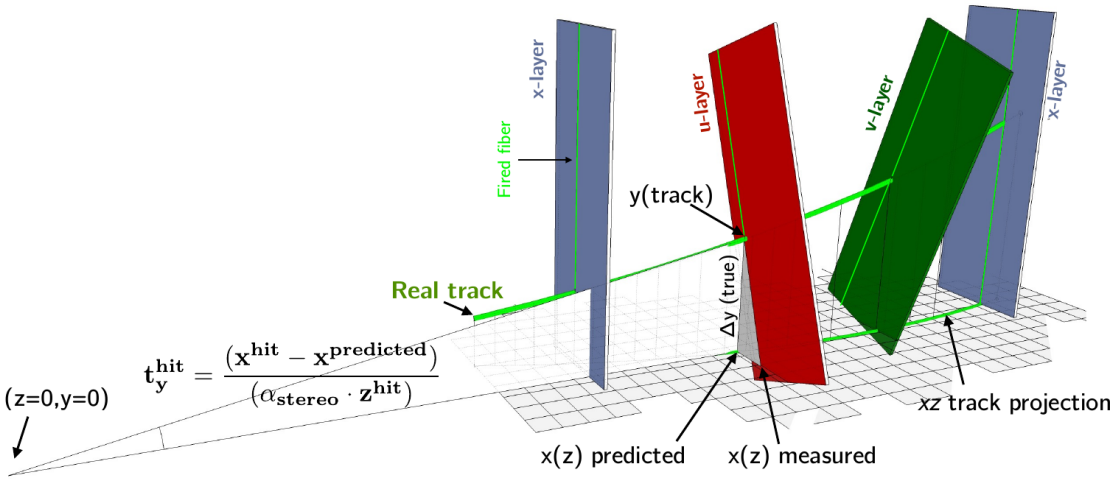



Figure 5.3: Graphic representation of how the $y$ position is computed. Fig. taken from [14].

## 5.5 Global clone removal

After all the *cases* have been executed, a clone removal step is performed based on the number of hits and fit parameters of each track. Tracks sharing more than the permitted number of hits are compared and the one with the lowest number of hits is killed. For the cases where they share the same number of hits, the one giving the best track fit is the one selected. This step directly produces standard LHCb objects ready to be used by other algorithms.

| Name | Property |
|---|---|
| *hasT* | reconstructible in SciFi |
| UT + SciFi | reconstructible in SciFi and UT |
| *noVelo* | not reconstructible in the VELO |
| *long* | reconstructible in VELO and SciFi |
| *strange* | daughter of a strange particle |
| *from B* | belongs to the decay chain of a b hadron |

Table 5.2: Description for each of the sub-sections for the different types of tracks.

## 5.6 Track recovery

Not all $x$-$z$ projections from the *cases* execution are promoted to full tracks when looking for matching $u/v$ hits. In this step, they are recovered in order to use hits which are not used by any of the full track candidates.

Although the execution of this step may increase the overall efficiency by about 3-4%, the *ghost rate* increases dramatically to 30-40%, a factor of 3-4 with respect to the baseline performance (when no track recovery is performed). This is due to the fact that the most important part of the building sequence is to not find matching hits and by being more tolerance to hit recovery and lowering the hit requirements it produces a high amount of fake tracks.

This is defeated by requiring a very limited amount of hits to be shared between tracks, which is done in two steps:

1. Selection of the $x$-$z$ projections to recover. All $x$-$z$ projections found by each *case* that haven't been selected are stored in a particular container. These recollected tracks are filtered and suppressed based on how unique they are, i.e. how many hits have which are not being used by already found tracks.

2. After this cleaning up, all these tracks are processed again through the stereo hit step and finally converted to LHCb objects as valid tracks.

## 5.7 Performance

This final section of the *Hybrid Seeding algorithm* is devoted to illustrate the performance of the algorithm, which can be found in [14]. A description of each of the track types can be found in 5.2 and the overall tracking performances can be seen in table 5.3.

| Track Type | Efficiency |
|---|---|
| *hasT* | (67.2 ± 0.1) % |
| long | (90.6 ± 0.1) % |
| *long P >5 GeV/c* | (94.8 ± 0.1) % |
| *long from B* | (93.4 ± 0.1) % |
| *long from B P >5 GeV/c* | (95.4 ± 0.1) % |
| UT + SciFi *strange* | (89.7 ± 0.1) % |
| UT + SciFi *strange* P >5 GeV/c | (95.2 ± 0.1) % |
| *noVELO* + UT + SciFi *strange* | (89.4 ± 0.1) % |
| *noVELO* + UT + SciFi *strange* P >5 GeV/c | (95.0 ± 0.1)% |
| *ghost rate* | (7.9 ± 0.1) % |
| *ghost rate (evt.avg)* | 4.5 % |
| *hit purity* | 99.7 % |
| *hit efficiency* | 96.8 % |

Table 5.3: Tracking performances for different track types of the Hybrid Seeding algorithm, along with the ghost rate, hit purity and hit efficiency.

# Chapter 6

# R-trees

One of the key differences in the implementation of the seeding reconstruction algorithm is the tree-like data structure called *R-tree* used in the present work. This chapter focuses on describing in detail such structure: an overview of the method is given in section 6.1, search, insertion and delete operations are explained in sec. 6.2.1, 6.2.2 and 6.2.3 respectively.

## 6.1 Overview

Proposed by *Antonin Guttman* in 1984 [6], *R-trees* are mainly focused on indexing multi-dimensional information such as geographical coordinates. This property makes them ideal for applications aimed at locating or searching for nearest neighbours objects by location.

They are considered similar to *B-trees*, both being considered balanced search trees, meaning that all leaf nodes are at the same height, data is organised in pages and are mainly designed to be stored in disk.

*R-trees* main idea is to group nearby objects and represent them with a *minimum bounding box*. Minimum bounding boxes are rectilinear shapes that completely contains the bounded objects, enclosing data objects or other bounding boxes [9]. It is constructed hierarchically by grouping the leaf boxes into larger boxes, higher level boxes which may themselves be grouped into even larger boxes at the next level. This logic produces overlapping of non-leaf node boxes, as shown in fig. 6.1.

For a given *R-tree* with order $(m, M)$, the following characteristics have to be satisfied:

- The root node of the tree contains at least two entries, unless it is a leaf, in which case it can contain zero or one entry.
- All internal nodes can store between $m \leq M/2$ and $M$ child entries. Each of these entries have the form of $(p, mbr)$, being $p$ a pointer to a child node and $mbr$ is the

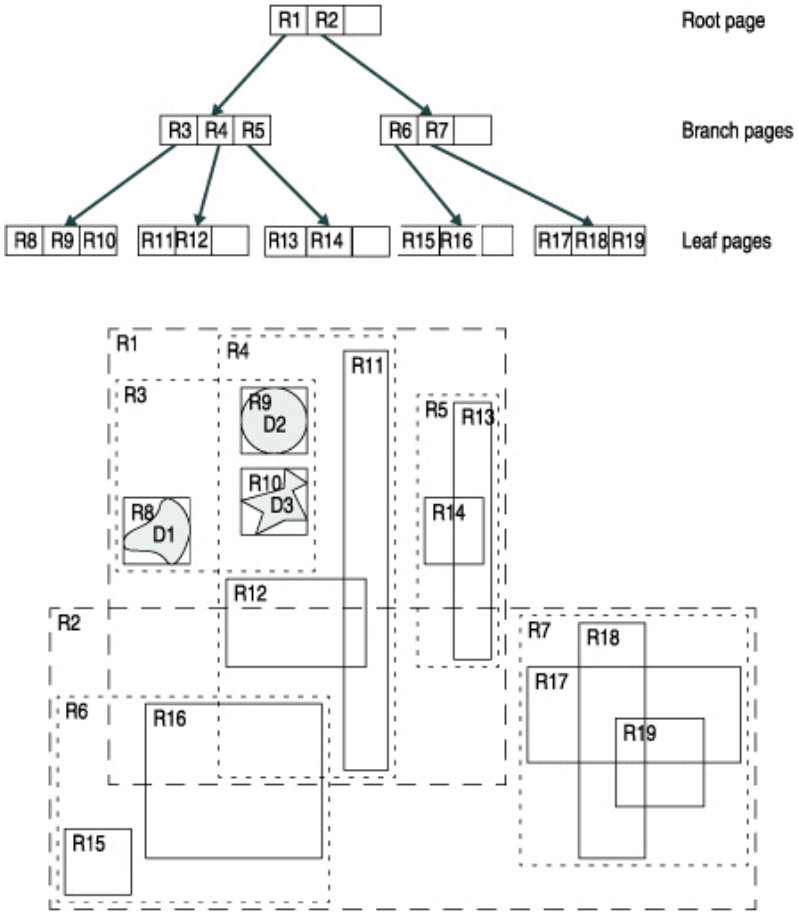



Figure 6.1: The leaf nodes of the R-tree store the exact *MBRs* or bounding boxes of the individual geometric objects, along with a pointer to the storage location of the contained geometry. All non-leaf nodes store references to several bounding boxes for each of which is a pointer to a lower level node [4]. Figure taken from [9]

MBR that spatially encloses all entries contained in the sub-tree rooted to this child.

- All leaf nodes can store between $m \leq M/2$ and $M$ entries. Each of these entries have the form of ($oid$, $mbr$), being $oid$ the object identifier and $mbr$ the MBR that spatially encloses this object.

- The tree structure is height-balanced, i.e. all leaves have the same level.

## 6.2 Operations

### 6.2.1 Search

Searching in *R-trees* is commonly done by having a search rectangle or window as an input, and thanks to the properties above mentioned, it has a rather simple logic.

Starting from the root node and continuing to the child nodes, every rectangle in a node is checked. If the rectangle overlaps the search rectangle, all the corresponding children have to be searched too. Rectangles that do not overlap are skipped saving search time and work. This procedure is continued recursively until reaching the leaf nodes of all overlapping nodes, the contents of which are then tested against the extent of the search window.

### 6.2.2 Insertion

*R-tree's* insertion is similar to *B-trees*. The new index records are added to the leaves, nodes that overflow are split, and splits propagate up the tree.

More in particular, *R-trees* insertions produce a traverse search in order to locate the new entry. The entry is then inserted in the given leaf updating all the nodes from this position upwards until reaching the root node. For the cases the entry cannot be accommodated because the node already has $M$ entries, a split into two nodes is performed. Given the premise that nodes are allowed to overlap, the objective of the split algorithm is to minimise the probability of retrieving the two new nodes with the same query, which translates into minimising the minimum area, as shown in fig. 6.2.2. In order to accomplish this, there are three different criteria:

- **Exponential Split**. Probably the most straightforward way to find the minimum area node split is to generate all possible groupings and choose the best. The number of possibilities is $2^{M-1}$, which becomes too large for large values of $M$.

- **Quadratic Split**. The objective of this algorithm is to find a small-area split, but does not guarantees that this will be the smallest of all. As the name states, this algorithm has a quadratic cost in $M$ and linear in the number of dimensions.

- **Linear Split**. Similar to the quadratic split but using a different version of the selection algorithm, the cost ends up being linear in $M$ for both splits and dimension.

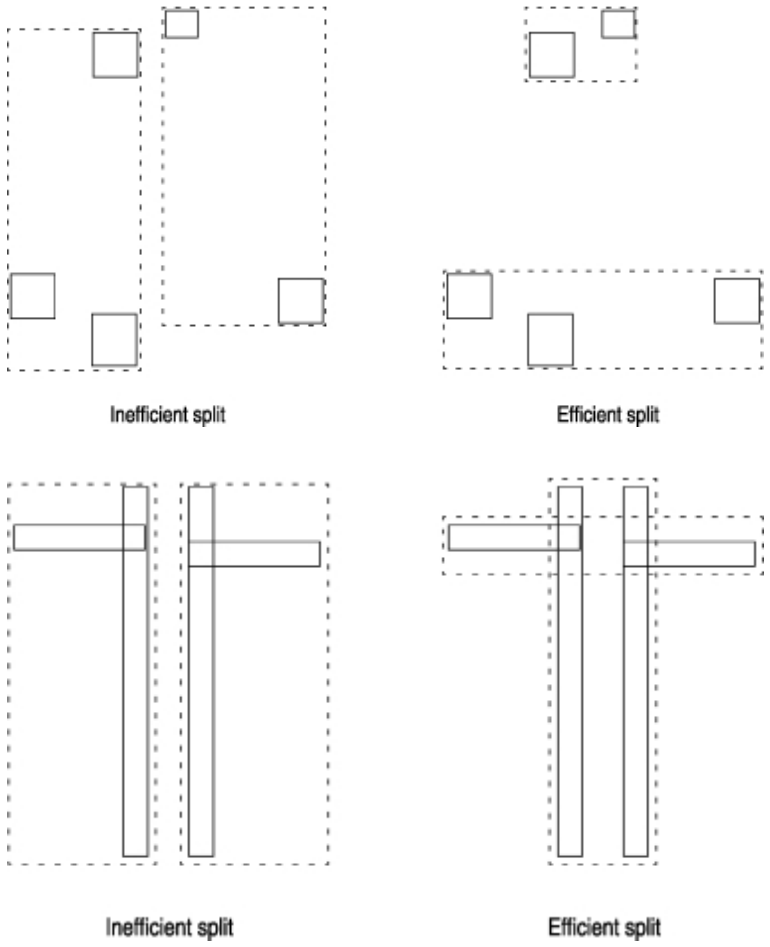



Figure 6.2: Comparison between efficient and inefficient split approaches when performing a node split. Figures taken from [6].

### 6.2.3 Deletion

When deleting entries in the tree, the insertion algorithm is applied again when a underflow occurs, *i.e.* when some node ends up with less than $m$ children. The reason why re-insertion is used instead of merging nodes, as explained in [6], is due to two main reasons:

- In order to reduce complexity. The *insertion* routine can be reused and the end result is similar.
- To prevent gradual deterioration. Re-insertion increments the spatial structure of the tree and prevents a gradual deterioration that might occur if each entry were located permanently under the same parent node.

# Chapter 7

# Artificial Neural Networks

The objective of this chapter is to give an overview of what Artificial Neural Networks (ANN) are in order to provide the reader with a background to understand the basics, together with the motivation to use them.

An overview of what ANNs are can be found in sec. 7.1 where the basic features are described. Their architecture is illustrated in sec. 7.2 and finally the learning process of ANNs in section 7.3.

## 7.1 Overview

Artificial Neural Networks, are computing systems which have been inspired by biological neural networks from animal brains. These systems are used as a way to create programs that learn from data in a supervised way to perform specific tasks, such as classification, prediction, etc.

### 7.1.1 Perceptrons

*Perceptrons* are artificial neurons first developed in the 1950s. These elements are simple systems that takes several binary inputs and produces a binary output and can be seen as a simple logic gate (see fig. 7.1.1). They are the basic building blocks inside the ANNs.

These inputs induce a given output base called *weights*, each of the input has one attached. These *weights* express the importance of the input to the overall output of the *perceptron*. So in order to produce a 1 or 0 as an output, the weighted sum of the inputs is performed and it will only output a 1 if it surpasses a given *threshold*. One *perceptron* cannot do much by itself, but when multiple of them are interconnected they are able to resolve complex problems, which is the reason why ANNs are interesting.

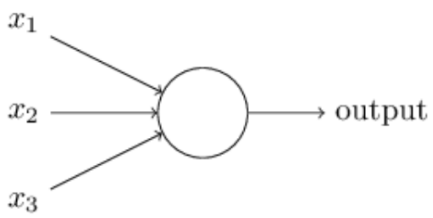



Figure 7.1: Sketch of a *perceptron* structure. Figure taken from [12]

### 7.1.2 Sigmoid neurons

Once we have a network of neurons, in this case *perceptrons*, we are able to adapt the network by changing the weights in order to solve a given problem. But as the number of *perceptrons* increases, it is difficult to predict how a small change in the weight of an input for a specific *perceptron* will impact the overall network.

This is solved by one of the most popular types of artificial neurons called *sigmoid neuron*. The particularity of this one is that a small change in the weight and bias can only cause a small change in the output. As the output of *perceptron* follows a step function, meaning it can only output 1 or 0, the *sigmoid neurons* produce a much more smooth version thus producing any real value ranging from 0 to 1.

## 7.2 Architecture of ANNs

Artificial neural networks structure can be separated in three sections. The leftmost layer, called the input layer, contains the *input* neurons. The middle section, contains the middle layers or also called *hidden layers*. Finally, at the rightmost part is the output layer, where the *output neurons* are.

This structure can be clearly seen in the figure 7.2, where there are 6 *input neurons*, 2 hidden layers with 4 and 3 *neurons* respectively, and finally a single *output neuron*. Networks having *hidden layers* are also called *multilayer perceptrons* (*MLPs*), despite being made up of *sigmoid neurons*, not *perceptrons*, or *feedforward neural networks*. The latter name is due the way information flows: it enters through the input and outputs at the last layer, there is no feedback [5].

ANNs are capable of doing incredible amount of things, from image classification to stock predictions. In any of these situations, both the input and output layer structure can be easily decided according to the objective, e.g. for an image classification system the input layer is going to have as many neurons as pixels in the images to classify and the output layer as many classes as the data contain.

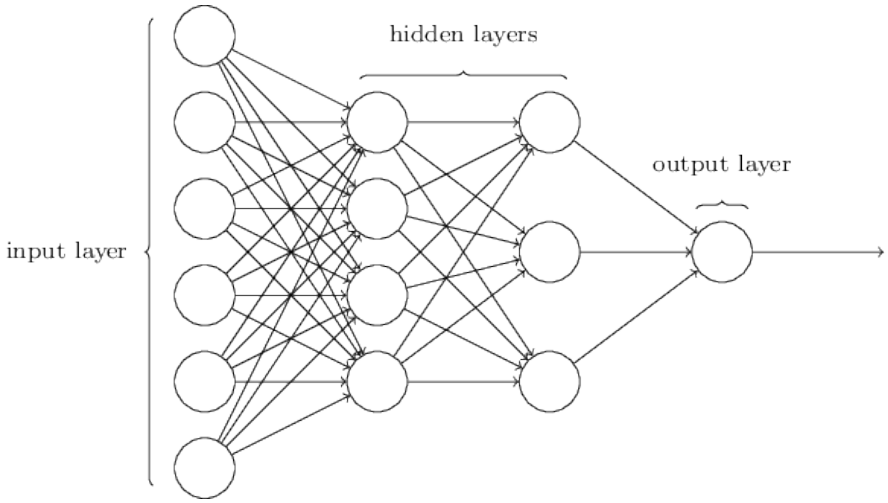



Figure 7.2: Illustration of a artificial network with two hidden layers. Figure taken from [12]

But when designing the *hidden layers* there is no such luck. There are many heuristics that can be used to help determine how to trade off in the number of hidden layers versus the complexity to train the network. Nevertheless for most of the problems, following the rule to have only a single hidden layer and have the number of neurons to be the mean between the input and output gives a decent performance. Besides it, an optimisation must be performed for each case as this is a generic solution.

## 7.3 Learning

One of the most interesting parts of ANNs is their ability to *learn*. By learning, it is referred as the fact they are able to find for a given task a specific pattern which enables them to solve it optimally.

In order to drive and measure the learning process, a concept called *cost function* represents the most important measure. It gives how far away a particular solution is from an optimal solution to the problem to be solved. The only way we are able to reduce it is by finding a set of weights and biases which will make the cost as small as possible. For this task the most common algorithm is the one known as *gradient descent*.

### 7.3.1 Gradient Descent

The gradient descent algorithm is used in order to minimise any function. It starts by a given function with a set of parameters and iteratively moves toward a set of values for these parameters that minimise the given function. This iteration is performed using differential

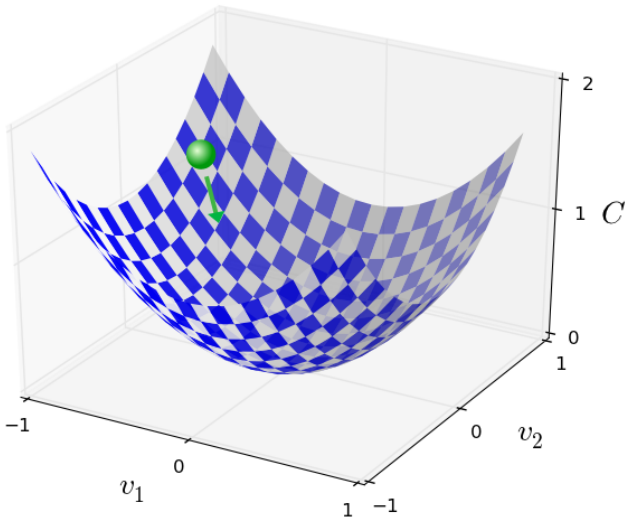



Figure 7.3: Example of a cost function dependent of two variables. The gradient descendent algorithm would compute the direction that minimises the cost in a given position, the green ball and the arrow. Iteratively, the variables would be updated to make the ball move towards the bottom, thus reducing the function. Figure taken from [12]

calculus by taking steps in the negative direction of the gradient function. A simple 3D function minimisation plot can be seen in figure 7.3.1.

Although the objective of this algorithm is to find the global minimum of the function, there are cases in which it may fall into a local one. This is the reason, for the ANNs, weights are selected randomly and multiple training procedures are recommended.

### 7.3.2 Stochastic Gradient Descent

A variation of the *gradient descent* algorithm, named *Stochastic Gradient Descent* (SGD), solves the complexity found when working with huge data-sets. In the traditional GD, when using larges amounts of data, each step is quite expensive as all the training samples have to be used. Instead, SGD reduces the amount of samples to compute each step by selecting a random group out of the training samples and using them to compute the error. This technique not only reduces the computing time but it is also less prone to select a local minimum.

### 7.3.3 Backpropagation

The method responsible to calculate the gradient of the loss function with respect to the weights of a ANN is the *backpropagation algorithm*. The execution of this method is what provides the learning for a ANN. This is done by three main steps, simplified below.

1. A training sample is fed into the network (this process is commonly called *forward propagation*) and the result is gathered.

2. The error of the output is computed, also called cost, and the *learning algorithm* such as SGD uses this difference to know the new values of the weights in order to minimise it

3. Once the new weights are known, they are *backward propagated* into to the network, updating the actual weights.

# Chapter 8

# The algorithm

The hybrid seeding algorithm represents the latest and best performing track reconstruction algorithm using solely the information provided by the Scintillating Fibre Tracker, described in sec. 5. The performance of this algorithm represents the baseline which any newer approach to track reconstruction in the SciFi sub-detector must compare to. This chapter focuses on describing an approach to a new algorithm which makes use of Artificial Neural Networks and spatial indexing data structures.

A simple overview of the algorithm is performed in sec. 8.1. Following a description of the r-trees usages and structures in sec. 8.2. Section 8.3 will describe the Artificial Neural Networks models and architectures alongside the performance indicators. The two main steps of the algorithm is described in sec. 8.4 and 8.5. Finally, the performance of the algorithm is in sec. 8.6.

## 8.1 Overview

The main idea at the basis of this algorithm is to use Artificial Neural networks for seed validation and spatial indexing to perform hit and seed selection, with the purpose of improving the tracking efficiency and reducing the *ghost rate*. Considering it is a first approach, it has been entire developed in `Python`, because of its dynamism and flexibility. Moreover, thanks to the number of libraries and frameworks available, the development time of an ANN model or a spatial indexing data structure is reduced drastically.

An overview of the algorithm steps is illustrated below.

1. All hits are added to the hit index except the ones extremely close to each other ($\pm 1$ *mm*). These are not introduced into the data structure in order to reduce clone tracks.

2. A parallel seed reconstruction at each station is performed. Resulting seeds are added to a new index storing seeds from each station.

3. Pairs of seeds from the first and second station are selected with the use of an ANN described later.

4. From each selected pair, a third seed is selected in the last station to complete a full track.

## 8.2 R-trees

The usages of R-trees for hit selection provides easy and fast access to location based search, which is of high importance in track reconstruction. This section will first describe the library used and then the two different indexes that have been used.

### 8.2.1 Library

The usage of *R-trees* in `python` has been possible thanks to the *Rtree spatial indexing* library. *Rtree* is a *ctypes*[1] `python` wrapper of `libspatialindex` [7] that provides a number of advanced spatial indexing features for `python` users [8].

Creating and using the index is quite simple. Moreover, for each one only the `insert` and `intersection` methods have been used, as they already provide all the necessary features.

- When creating an index, it is possible to determine the number of dimensions data is going to be indexed by. Furthermore, it is possible to define the order of coordinates, either (x0,y0,x1,y1) or (x0,x1,y0,y1) by the usage of the `interleaved` parameter. All the indexes created in the algorithm have this value set to `False`, meaning the last coordinate order will be the one used when referring to coordinates for all the chapter.

```
p = index.Property()
p.dimension = 3
index_tree = index.Index(properties=p, interleaved=False)
```

- `insert`. Inserting an element in the index is rather simple: the coordinates and the data of the object are introduced. For single point objects the minimum and maximum values must be the same. E.g. coordinates (0,0,1,1).

```
insert(id, coordinates, obj=None)
```

- `intersection`. When performing spatial search, this function will return all elements contained in the region introduced as coordinates. E.g. in a 2D environment, coordinates (0,1,0,1), will gather all objects inside the square area of 1x1.

```
intersection(coordinates, objects=False)
```

[1] *ctypes* is a foreign function library for Python. It provides `C` compatible data types, and allows calling functions in *DLLs* or shared libraries. It can be used to wrap these libraries in pure Python.

### 8.2.2 Indexes

This section will describe in detail the two r-tree indexes used: `hits_index` and `seeds_index`.

- `Hits_index`. Stores hits with a 3 dimensional coordinate system (see detailed description in 4.2.2):

  1. **Hit_X**. X position, ranging from -3182 to 3182 *mm*.
  2. **Hit_planeCode**. Layer code a hit belongs to, from 0 to 11 (e.g. 6 for the *U-layer* in the second station).
  3. **Hit_Zone**. Boolean, 0 or 1 depending if the hit is in the upper or lower part of the layer respectively.

  Hits are stored as single points in the index, i.e. they are not stored as geometric entities with ranges for each of their dimensions (e.g. *min_x*, *max_x*). The following data per hit are stored as an object:

  - **Hit_LHCbID**. Hit ID, unique only event-wise.
  - **Hit_X**. X position, ranging from -3182 to 3182 *mm*.
  - **Hit_zone**. Hit zone, describing if it is located in the upper or lower part of the layer.
  - **Hit_particle_id**. Identification of the particle type the hit belongs to. Only used in performance indicators.

- `Seeds_index`. It stores station seeds with a 3 dimensional coordinate system:

  1. **X position**. First seed Hit_X position, ranging from -3182 to 3182 *mm*.
  2. **Y position**. Height of the seed. Two values are used, the height extracted from the first *X-layer* with *U-layer* and the height from the second *X-layer* with *V-layer*.
  3. **Station**. Number of the station, from 0 to 2.

  In this case, objects stored can be seen as lines when represented in a 3D environment. This is due to the fact that height is defined with a min and max values, but all the other dimensions only have one. Regarding the objects, they contain the following data (all positions are expressed in *mm*):

  - **Seed**. Each station seed has the following attributes:
    1. First *X-layer* $x$ position.
    2. *U-layer* $x$ position.
    3. Computed height from the first *X-layer* with *U-layer*.

4. *V-layer* $x$ position.
5. Computed height from the second *X-layer* with *V-layer*.
6. Last *X-layer* $x$ position.
7. Station number
8. Tuple containing all the IDs of each hit in the given seed.
9. Tuple containing all the particle IDs of each hit in the given seed. The created ID for each particle is unique for a single event.

– **Hash**. Hash value from the string representation of the seed.

## 8.3 Artificial Neural Networks Models

Two of the main applications of Artificial Neural Networks is to predict or classify data. This algorithm uses the classification power of ANNs to validate the hit structure of single seeds and the unification of seeds from different stations. Three validation models have been developed for the following situations:

- **Combination of hits for station seed construction**. All possible combinations from a the same starting *x-layer* hit are passed through the `seed_station_validation` ANN model to get the one with maximum accuracy.

- **Pair of station seeds**. Given a first station seed, the list of possible selected second station pairs is filtered by the `two_seed_validation` model, to select one final combination with the best possible accuracy.

- **Complete track combinations**. After gathering the possible first and second station pairs, a selection of the third is done by using the `track_validation` model, which is responsible of filtering the final combination.

For the development of this models, the framework `Keras` [11], using the `TensorFlow` [16] back end, has been used together with `sklearn` [10] for utilities such splitting of train and test samples. In the following sections, a more detailed description of the ANN models is done together with their respective performances.

### 8.3.1 Architecture and Training of ANN models

This section will describe the architecture and training data for each model. Regarding the general configuration, all of them share the same training optimiser, *sgd*. The number of internal layers and neurons together with the activation function have also been shared between them. Being 12 and 8 the number of neurons for each internal layer respectively with the *Rectified Linear Unit* activation function.

#### 8.3.1.1 `seed_station_validation`

It has a 7-12-8-1 neuron structure, from the input layer to the output.

- **Input Layer**. The data fed into this ANN has the same content as a stored *seed* into the `seeds_index` (seen in sec. 8.2.2), except the tuples containing the Hit and particle IDs. This means the network has 7 inputs.
- **Output Layer**. Single neuron defining the probability the combination has a valid seed structure. The activation function of neurons in this layer is the *sigmoid* function.

When performing the learning process, seeds are considered valid if all four hits come from the same particle and invalid if none of them share the same particle ID.

#### 8.3.1.2 `two_seeds_validation`

Following the algorithm execution, the next prediction model is the two_seeds_validation. Its structure in terms of number of neurons per layer is 14-12-8-1.

- **Input Layer**. The input layer receives two concatenated seeds. The first one always being the one closest to the interaction point followed by a seed in the next closest station.
- **Output Layer**. Single neuron whose output is the probability the combination has a valid seed pair structure.The activation function of neurons in this layer is the *sigmoid* function.

Regarding the training, the combination of the input data is valid if they share at least 7 out of 8 hits from the same particle and invalid if they share less than 7.

#### 8.3.1.3 `complete_seeds_validation`

The last prediction model is the complete_seeds_validation. Its structure in terms of neurons per layer is 18-12-8-1.

- **Input Layer**. The input layer receives three concatenated seeds. The first one is a station 1 seed and the last one is from station 3. Instead of feeding the same *seed* structure as the other two models, the *station* attribute is no longer used, since it does not provide any useful information in this situation.
- **Output Layer**. Single neuron giving the probability that a combination is a valid SciFi track. The activation function of neurons in this layer is the *sigmoid* function.

During the training process, a combination of seeds is considered valid if the combination shares more than 10 out of 12 hits from the same particle. Regarding the invalid combinations, only those which share less than 8 are considered invalid. Remaining combinations with hits ranging from 8 to 10 are discarded.

### 8.3.2 Performance indicators

For each of the ANN models, the *ROC* curve and the classes separation is shown.

#### 8.3.2.1 `seed_station_validation`

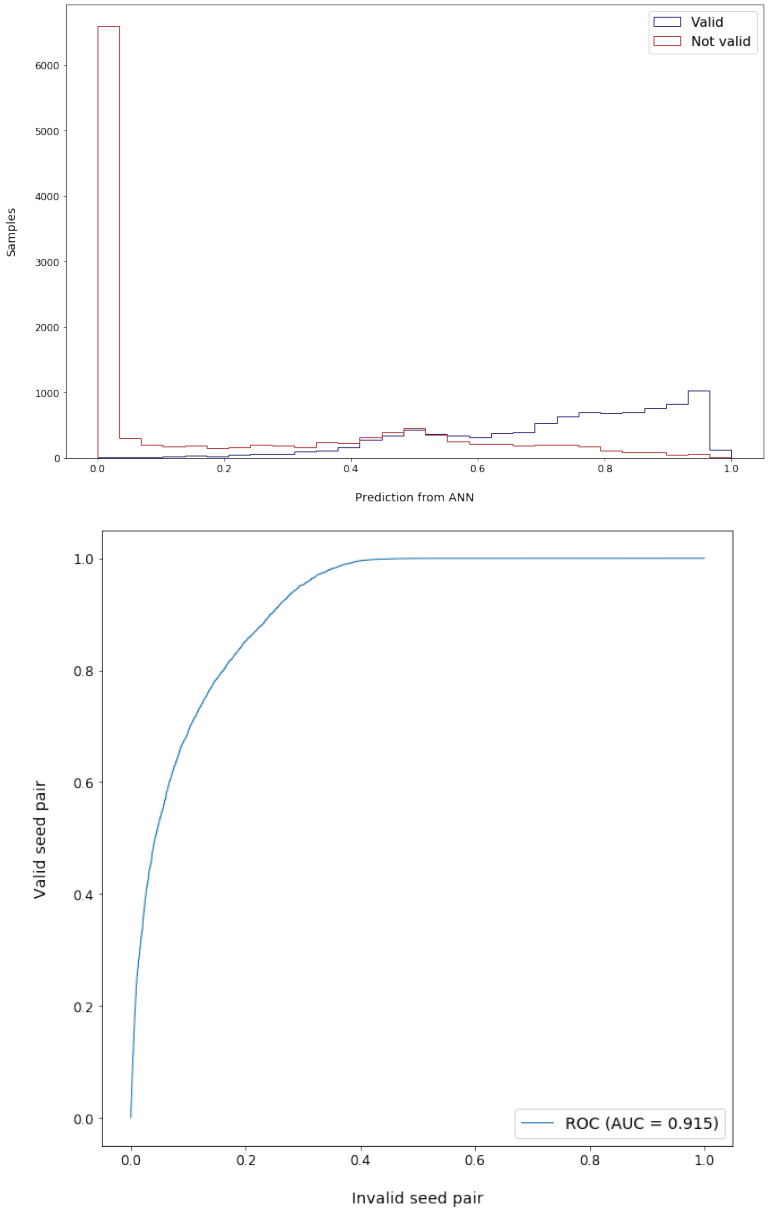



Figure 8.1: The first figure show a histogram representation of the the ability to separate the valid and invalid hit combinations. A *ROC* curve is illustrated in the second plot, with an *Area under the Curve* (AUC) of 0.915.

#### 8.3.2.2 `two_seeds_validation`

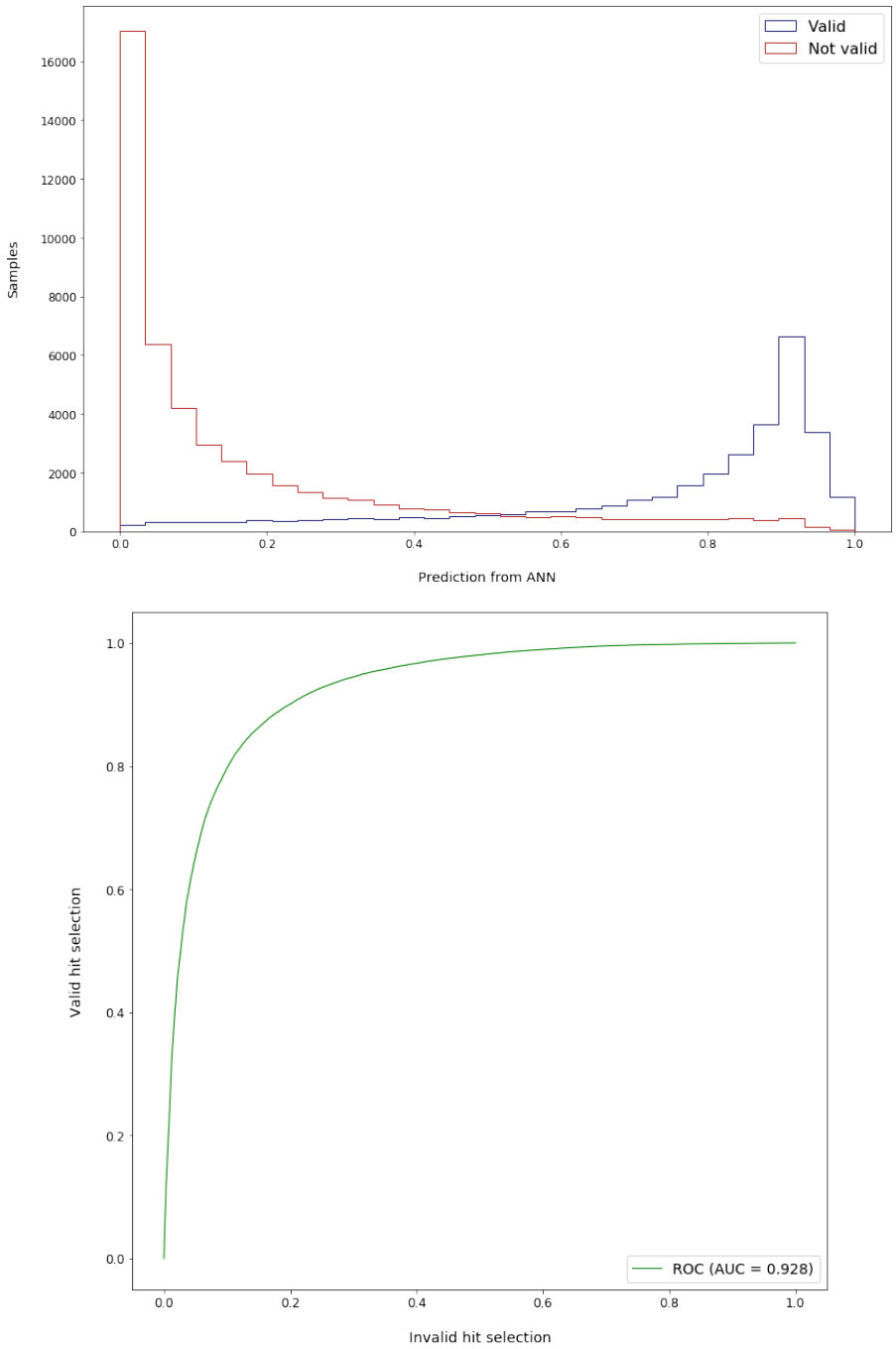



Figure 8.2: The first figure show a histogram representation of the the ability to separate the valid and invalid seed combinations. A *ROC* curve is illustrated in the second plot, with a *AUC* of 0.928.

#### 8.3.2.3 `complete_seeds_validation`

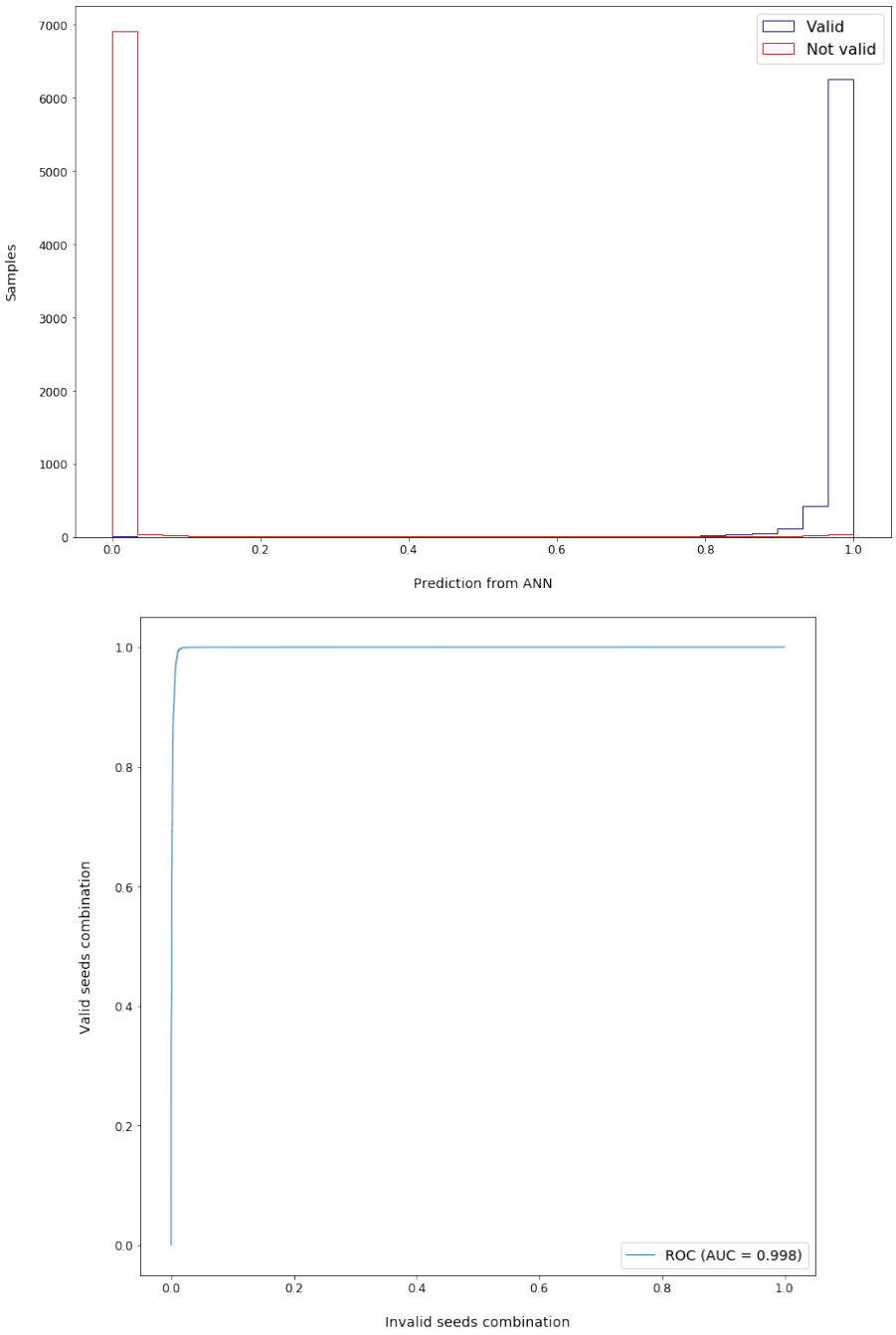



Figure 8.3: The first figure show a histogram representation of the the ability to separate the valid and invalid track combinations. A *ROC* curve is illustrated in the second, with a *AUC* of 0.998.

## 8.4 Station seed reconstruction

The first of the two main steps in this algorithm (bullet 2 in 8.1) is the `seedStationReconstruction` routine. It selects for a given station all possible hit combinations that may produce a valid seed for each of the first *X-layer* hits. This is done in several steps:

1. Selects all hits from the `hit_index` inside the coordinates (min_x, max_x, x0_layer,x0_layer, 0, 1). This will return all hits in the first layer of the station performing the reconstruction, both in the upper and lower part.

2. For each of the gathered first layer (*X0-layer*) hits, referred as *Hit*:

   a) Predict $x$ positions using basic line projections in the *U-V-X1* layers. Similarly to the *Hybrid Seeding* approach, a projection is performed simulating the hit belongs to a track coming from the (0,0) coordinates.

   $$m = Z_{X_0} \ / \ Hit_X$$

   $$n = Z_{X_0} \ - \ (m \ * \ Hit_X \ )$$

   $$PredX_U = (Z_U - n) \ / \ m$$

   $$PredX_V = (Z_V - n) \ / \ m$$

   $$PredX_{X_1} = (Z_{X_1} - n) \ / \ m$$

   As can be seen in fig. 8.5, the difference between the predicted and the real do not differ by much. Based on this premise, creating a ANN for this only purpose would be overkill, as it would increase execution time without not much benefit.

   b) Once the predicted positions are computed, now we access the `hits_index` to gather all the possible *U-V-X1* layer hits. It is done by using the `intersection` method, previously described (sec. 8.2.1), with the following coordinates:

      i. ( $PredX_U$ - limit, $PredX_U$ + limit, $U_{planecode}$, $U_{planecode}$, $Hit_{Zone}$, $Hit_{Zone}$)
      ii. ( $PredX_V$ - limit, $PredX_V$ + limit, $U_{planecode}$, $U_{planecode}$, $Hit_{Zone}$, $Hit_{Zone}$)
      iii. ( $PredX_{X_1}$ - limit, $PredX_{X_1}$ + limit, $X_{1planecode}$, $X_{1planecode}$, $Hit_{Zone}$, $Hit_{Zone}$)

   As can be seen, the $x$ dimension for each of the coordinates is introduced as a range. This is done to overcome the error the prediction might produce. The width of the search window may be changed dynamically, but the best value for a balance between performance and ghost rate has been found to be 200 *mm*.

   c) The next step is the creation of combinations for each *U-V-X1* layer hits producing station seeds. It is performed with the `selectValidCombination` routine:

- If all three `hits_index` queries above have returned possible hits, a three level *loop* indent is performed in order to compute the height of the seed using the `predictHeight` method. It computes the height in the same way as described in the *stereo hits* (sec. 5.4) of the *Hybrid Seeding* algorithm.

  All the ones with a height difference between the first *X0-U* combination and the second *V-X1* greater than 50 *mm* are discarded. This value has been selected based on the computed standard deviation from fig. 8.4.

- For the cases where no hits where found in any of the three layers, the seed is created with missing hits, with the id and the $x$ position to zero.

d) After all the possible hit combinations have been created, the `seed_station_validation` ANN model is used to gather the accuracy of each combination. Only the ones with a probability of 20% or more are introduced into `seed_index`. This low probability is selected in order to remove most unlikely seeds.

This `seedStationReconstruction` routine is executed three times, one for each station. This step can be independently executed for each station. As a result, a parallel execution is performed to decrease execution time by using a *Thread pool* of 3 processes. Furthermore, access to shared variables such as `hits_index` and `seeds_index` is controlled by the usage of *Locks*, to synchronise access to the shared resources.

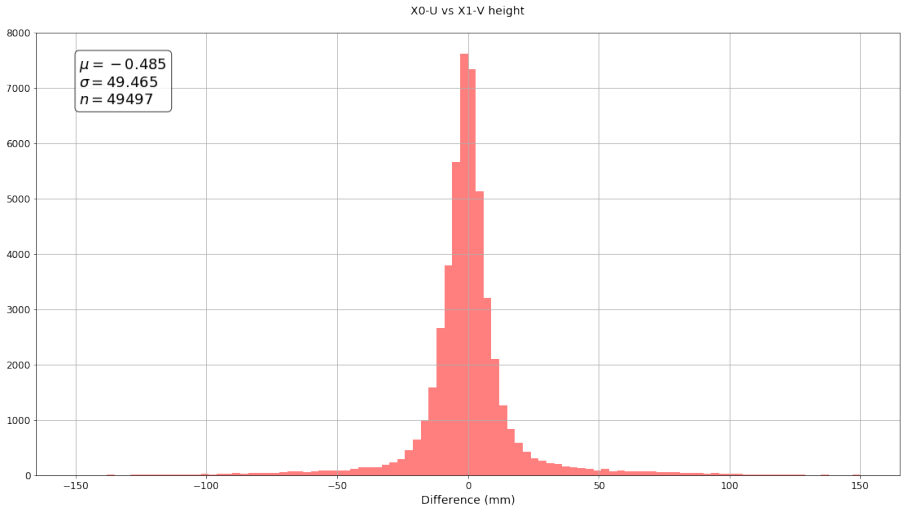



Figure 8.4: Difference between the predicted positions of *X0-U* and *X1-V*.

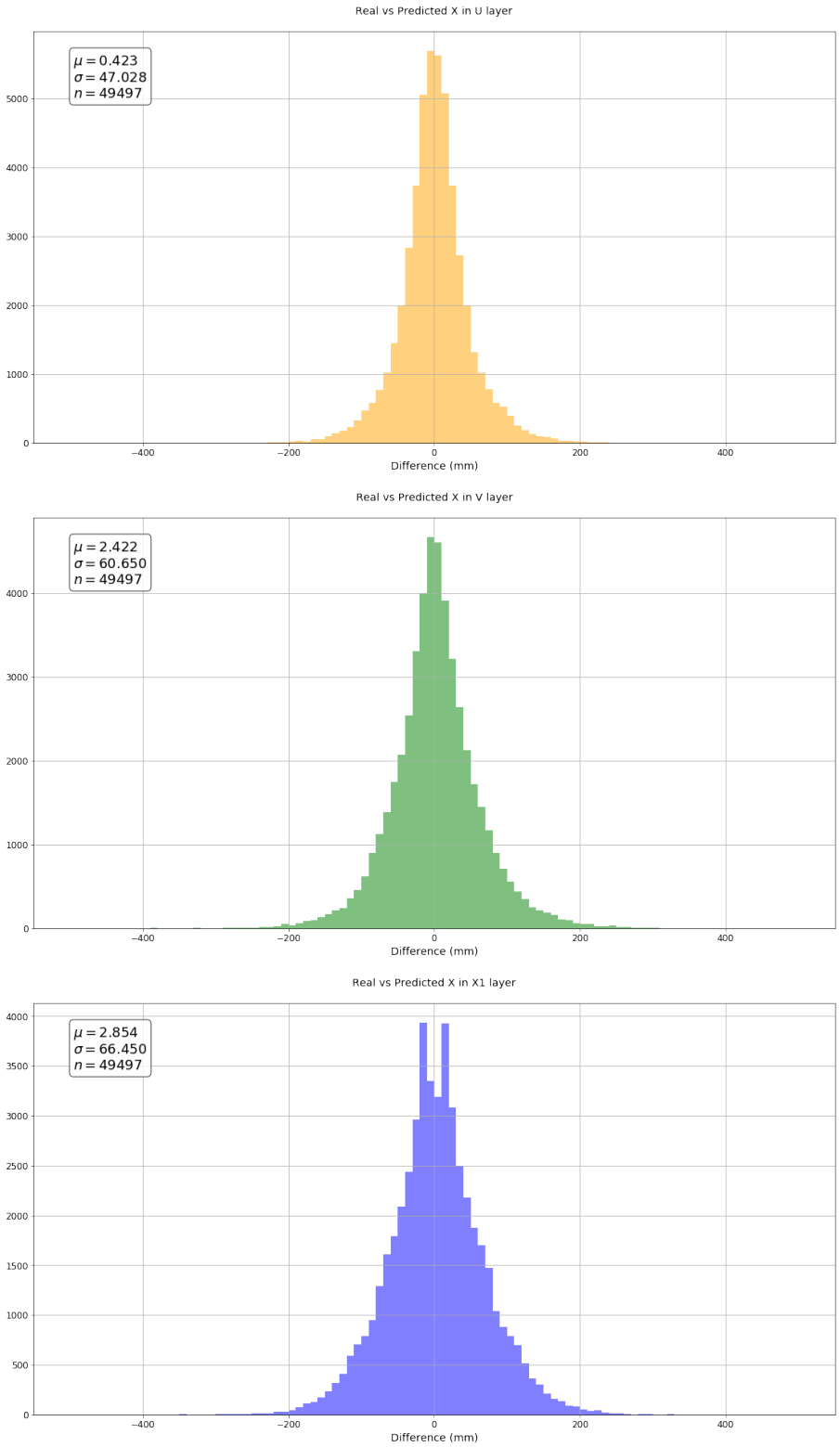



Figure 8.5: Difference histograms of the real position versus the predicted ones using line projections for *U-V-X1* layers respectively. $n$ represents the number of particles used

## 8.5 Unification of stations seeds

After the simultaneous execution of the three *Threads*, the `seeds_index` contains all the possible seeds for each station. So this step is responsible of unifying them by $x$ and $y$ positions. The logic is similar to the `seedStationReconstruction` routine, starting from the first station it constructs tracks by matching seeds with the contiguous station. A more detailed description is illustrated below.

1. Gathering all possible seeds in the first station by performing a `intersection` on the `seeds_index` by the following coordinates: ((min_x, max_x, min_y, max_y, 0, 0)). A more comprehensive description of this method along with the coordinate system can be found in 8.2.1 and 8.2.2 respectively.

2. For each seed in the first station:

   a) If more than 25% of the seed hits have been previously used by other constructed tracks, the seed is discarded. This reduces the number of *clone* and *ghost* tracks significantly. In addition, by performing this check at the beginning, the execution time is also reduced.

   b) Predict the possible position of the next seed from the contiguous station by performing line projection. Similar as when constructing seeds, the projection is used when accessing the `seeds_index` by the `intersection` method to gather all the possible seeds within a range.

      The coordinates are: ($PredX_{S1}$ −100, $PredX_{S1}$ + 100, mean - 150, mean + 150, 1, 1). The values $PredX_{S1}$ and *mean* are computed:

      - $PredX_{S1}$: Using the $z$ and $x$ positions of the first station seed, the predicted position at the first layer of the next station is performed as follows:

        $$m = (Z_{X1} - Z_{X0})/(X_{X1} - X_{X0})$$

        $$n = Z_{X0} - (m * X_{X0})$$

        $$PredX_{S1} = (NextZ_{X0} - n) \ / \ m$$

        Where $NextZ_{X0}$ represents the $z$ position from the first layer of the next station.

      - *mean*: Represents the mean value of the two heights the first layer seed has:

        $$mean = (Y_{X0-U} + Y_{X1-V})/2$$

   c) From all the seeds gathered from the previous *r-tree* query, only the one with the highest accuracy is selected by using the `two_seeds_validation` ANN model.

d) If the previous selection does not have an accuracy higher than 50% is discarded and no further search is perform using the current first station seed. In addition, the seed hits are also checked in order to ensure that less than 25% of hits have already been selected.

e) Similar as before, a line projection is performed using the resulting seed hits towards the first layer of the last station.

The coordinates are: ($PredX_{S2}$ −150, $PredX_{S2}$ + 150, mean - 150, mean + 150, 2, 2). The values $PredX_{S2}$ and *mean* are computed:

- $PredX_{S2}$: Using the $z$ and $x$ positions of the recently found second station seed, the predicted position at the first layer of the next station is performed as follows:

$$m = (Z_{X1} - Z_{X0})/(X_{X1} - X_{X0})$$

$$n = Z_{X0} - (m * X_{X0})$$

$$PredX_{S2} = (NextZ_{X0} - n) \ / \ m$$

  Where $NextZ_{X0}$ represents the $z$ position from the first layer of the next station.

- *mean*: Represents the mean value of the two heights the second layer seed has:

$$mean = (Y_{X0-U} + Y_{X1-V})/2$$

f) Again, from all the seeds gathered from the last station inside the search window only the one with the highest accuracy using the `complete_seeds_validation` is selected.

g) In case the later does not have an accuracy higher than 50%, it is discarded. Moreover, if more than 25% of the hits of this selected seed have already been used by others, it is also discarded.

h) Finally, if the combination of seeds comply with all the previous requirements, the track is stored and all the hits from each seed are tagged as used, to prevent duplicate tracks.

## 8.6 Performance

In this section, the results of the algorithm's performance are going to be shown. In order to generate the following values, 100 event batches have been used performing the mean between all the performance indicators from each event. The results are separated in three main sections: *All particles*, *Particles with hits in all layers* and *Particles with at least one missing hit.*

| Missing Hits | Fraction |
|---|---|
| None | 63.32 % |
| 1 | 24.36 % |
| 2 | 9.17 % |
| 3 | 2.13 % |
| 4 or more | 1.00% |

Table 8.1: Proportion of particles with missing hits in the SciFi simulated data.

The main reason to this decision is to show how much impact has in the algorithm's performance the fact that particles do not always produce hits in all layers, approximately about 36% of them, as seen in table 8.1.

The tables shown below have the same structure as in the *Hybrid seeding algorithm* performance section 5.7 with the addition of the *Seed reconstruction* column. It has been added in order to prove, as seen in the following results, that the algorithm's weak spot is the unification of the constructed seeds rather than the creation of them. The value is computed as the mean between the three station seed reconstructions efficiency's.

### 8.6.1 All particles

The results obtained can be directly compared with the *Hybrid seeding algorithm*, with a clear disadvantage in the overall efficiency in all the different types of tracks, as seen in table 8.2. Nevertheless, the *ghost rate* parameter is reduced in comparison, which is mainly produced by three reasons:

- Get only the combinations with the highest accuracy from the ANNs.
- Not considering valid the ones that didn't reach a given accuracy threshold.
- Discarding the entire track if one of the seeds has more than 25% of hits which have been already selected.

Moreover, with figure 8.6, it is possible to conclude also that an increase in the number of hits per event will directly imply a lower efficiency and higher ghost rate. On the other hand, the seed reconstruction efficiency is not affected by much, as seen in the results from the figure.

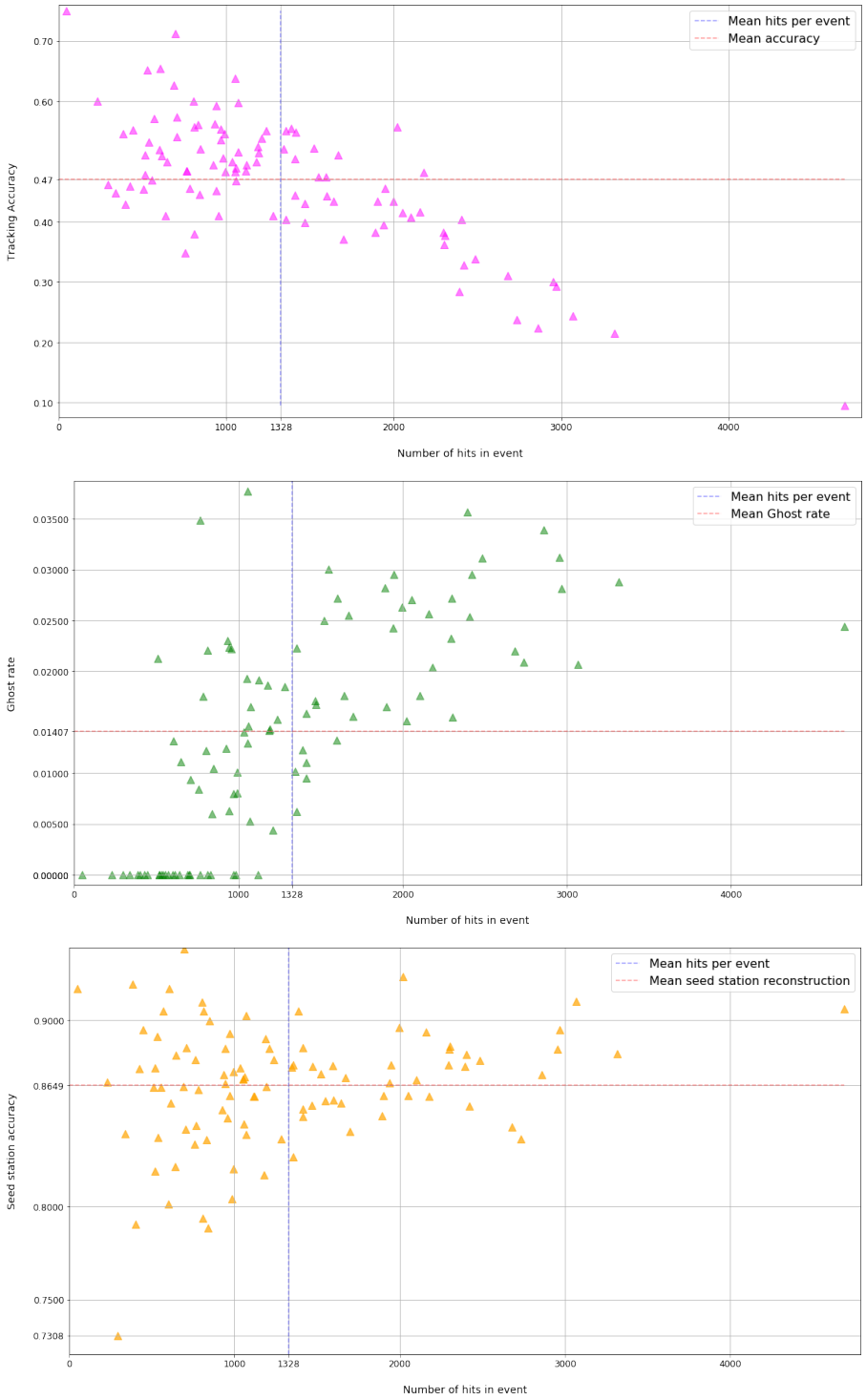



Figure 8.6: Scatter plots containing the *Track reconstruction efficiency*, *Ghost rate* and *Seed reconstruction efficiency* versus the number of hits per event.

| Track Type | Efficiency | Ghost Rate | Seed reconstruction |
|---|---|---|---|
| hasT | (47.3 ± 2.1) % | (1.1 ± 0.2) % | (86.5 ± 0.1) % |
| long | (61.4 ± 2.5) % | (0.7 ± 0.2) % | (89.8 ± 0.1) % |
| long P >5 GeV/c | (66.0 ± 2.6) % | (0.6 ± 0.1) % | (92.3 ± 0.1) % |
| long from B | (60.5 ± 2.6) % | (0.02 ± 0.05) % | (92.3 ± 0.1) % |
| long from B P>5 GeV/c | (62.1 ± 2.8) % | (0.02 ± 0.05) % | (92.1 ± 0.1) % |
| UT + SciFi strange | (65.5 ± 2.6) % | (0.01 ± 0.04) % | (92.1 ± 0.1) % |
| UT + SciFi strange P >5 GeV/c | (67.4 ± 3.0) % | (0.02 ± 0.05) % | (93.6 ± 0.1)% |
| noVELO + UT + SciFi strange | (59.8 ± 2.8) % | 0.0 % | (92.4 ± 0.1) % |
| noVELO + UT + SciFi strange P >5 GeV/c | (67.1 ± 3.3) % | 0.0 % | (94.1 ± 0.1) % |

Table 8.2: Efficiency, *ghost rate* and seed reconstruction efficiency for all particles.

### 8.6.2 Particles with hits in all layers

Regarding tracking reconstruction of particles with hits in all layers, the results are clearly different, showing an increase of 27.77% from the base efficiency, as seen in table 8.3. It is clear that the algorithm performance is clearly affected by the missing hits, resulting a key feature to be solved in future versions.

### 8.6.3 Particles with one missing hit maximum

This last section shows the efficiency of track reconstruction with particles having hits in all or almost all layers (one missing). The addition of these particles in the reconstruction severely affects the efficiency, resulting in a decrease of 21% in comparison to the reconstruction of particles with hits in all layers.

| Track Type | Efficiency | Ghost Rate | Seed reconstruction |
|---|---|---|---|
| hasT | (74.8 ± 2.8 % | (0.7 ± 0.2)% | (96.64 ± 0.1) % |
| long | (89.8 ± 3.1) % | (0.4 ± 0.1) % | (98.97 ± 0.5) % |
| long P >5 GeV/c | (93.4 ± 3.2) % | (0.2 ± 0.1) % | (99.4 ± 0.5) % |
| long from B | (89.3 ± 3.4) % | 0.0 % | (99.6 ± 0.5) % |
| long from B P>5 GeV/c | (90.5 ± 3.6) % | 0.0 % | (99.8 ± 0.2) % |
| UT + SciFi strange | (91.0 ± 3.2) % | 0.0 % | (99.4 ± 0.2) % |
| UT + SciFi strange P >5 GeV/c | (94.2 ± 3.7) % | 0.0 % | (99.9 ± 0.2) % |
| noVELO + UT + SciFi strange | (87.5 ± 3.5) % | 0.0 % | (99.3 ± 0.5) % |
| noVELO + UT + SciFi strange P >5 GeV/c | (92.8 ± 3.1) % | 0.0 % | (99.9 ± 0.1) % |

Table 8.3: Efficiency, *ghost rate* and *seed reconstruction* efficiency with particles having hits in all layers.

| Track Type | Efficiency | Ghost Rate | Seed reconstruction |
|---|---|---|---|
| hasT | (53.2 ± 2.6) % | (1.2 ± 0.1) % | (90.4 ± 0.1) % |
| long | (66.8 ± 2.8) % | (0.73 ± 0.02) % | (92.9 ± 0.2) % |
| long P >5 GeV/c | (71.8 ± 2.6) % | (0.70 ± 0.05) % | (94.5 ± 0.1) % |
| long from B | (66.9 ± 2.8) % | (0.01 ± 0.02) % | (95.1 ± 0.2) % |
| long from B P>5 GeV/c | (68.4 ± 3.0) % | (0.02 ± 0.06) % | (95.5 ± 0.1) % |
| UT + SciFi strange | (68.4 ± 2.8) % | (0.01 ± 0.03) % | (94.1 ± 0.2) % |
| UT + SciFi strange P >5 GeV/c | (72.1 ± 3.1) % | (0.02 ± 0.05) % | (95.3 ± 0.2) % |
| noVELO + UT + SciFi strange | (65.4 ± 2.9) % | 0.0 % | (94.3 ± 0.1) % |
| noVELO + UT + SciFi strange P >5 GeV/c | (71.5 ± 3.5) % | 0.0% | (96.5 ± 0.1) % |

Table 8.4: Efficiency, *ghost rate* and seed reconstruction efficiency with particles having hits in all layers or missing at most one hit.

# Chapter 9

# Conclusions and time distribution

## 9.1 Conclusions

The LHCb detector will be upgraded to cope with the planned increase of luminosity in *Run 3*, from the front-end electronics and sensitive elements of the sub-detectors to a new 40 Mhz full software trigger system. As a result, many improvements have to be made in the actual tracking and identification systems to maintain the efficiency as good as possible.

In particular, this project aims at improving the track reconstruction efficiency for the new Scintillating Fibre Tracker with the use of alternative techniques not exploited in the current *Hybrid seeding* reconstruction algorithm, such as Artificial Neural Networks and spatial indexing data structures.

Results have shown that reconstruction efficiency is 20% below the *Hybrid seeding* performance. Nevertheless it gains in *ghost rate* rate reduction, with a decrease from (7.9 $\pm$ 0.1) % to (1.1 $\pm$ 0.2) %.

The efficiency degradation is mainly caused in the seed unification step rather the actual seed creation. This is thought to be produced in the process of joining seeds with wrongly formed hits. Since these seeds have an empty slot or even a wrongly assigned hit, the probability computed by the three-seed validation ANN is lower than for combinations where the seed missing a hit is replaced for a nearby seed having all hits belonging to the same particle. As a consequence, when selecting the combination with the highest probability the correct combination is rarely selected.

This statement is supported by the efficiency results obtained from the algorithm execution when using particles with hits in all layers, which clearly surpasses the algorithm's base efficiency by $\sim$28%, going from 47% to 75%.

It can also be seen higher number of hits per event directly implies lower efficiency. This is mainly due to the usage of fixed ranges when querying into the hit and seed indexes introducing an increase in the number of combinations, together with a higher probability of matching incorrectly.

The above statements also show how this algorithm can be improved: 1) Implementation of a better system for joining seeds, utilising different heuristics for selecting seed combinations; 2) Variable range for the hit and seed indexes, e.g. increasing the range for the outer region of the sub-detector and decreasing it for the inner one; 3) perform an exhaustive research aim at improving the ANN models, as they have a high impact on the track reconstruction efficiency.

On top of that, it is also necessary to compare the execution time with the *Hybrid seeding* algorithm, porting the current `Python` implementation into efficient C++ code and making use of more memory- and CPU-efficient libraries.

At a personal level, the development of this project has brought invaluable experience both at the academic and personal level. From the general understanding of an incredible project such as CERN, to the opportunity to develop a tracking algorithm with a real scientific purpose rather than for an exclusive academic purpose.

## 9.2 Time management

With a total time dedication of 510 hours, the time distribution of the project is shown in the figure below.

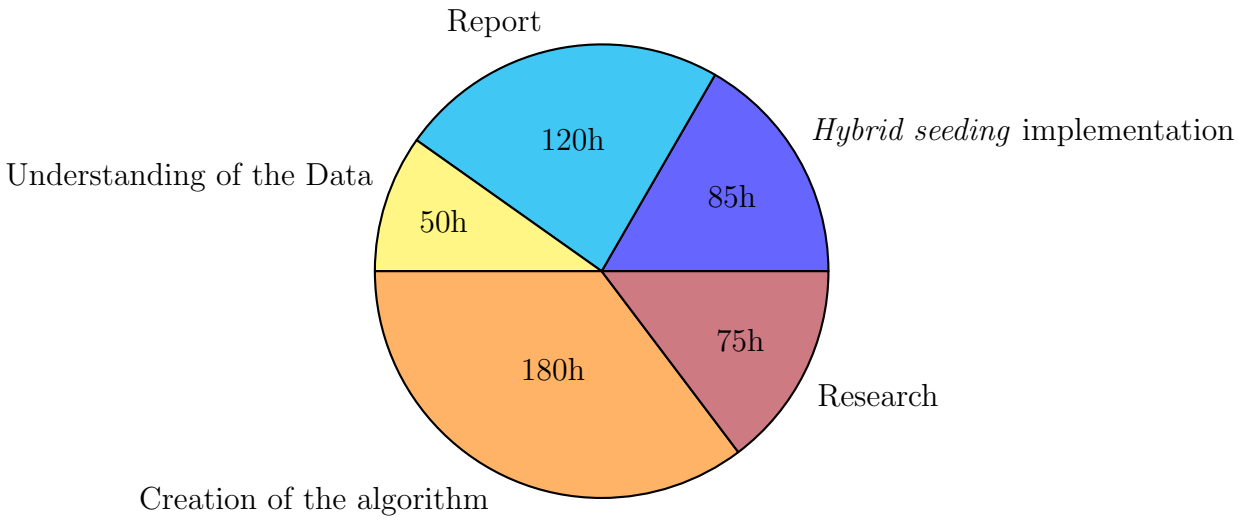



Figure 9.1: Representation with the form of a circular diagram the time distribution of the project

# Bibliography


[1] CERNd. *About CERN*. Tech. rep. 2015. URL: https://home.cern/about.

[2] LHCb Collaboration. *LHCb Tracker Upgrade Technical Design Report*. Tech. rep. CERN-LHCC-2014-001. LHCB-TDR-015. Feb. 2014. URL: https://cds.cern.ch/record/1647400.

[3] G. Corti. *Overview of Monte Carlo simulation(s) in LHCb*. Oct. 2009. URL: https://lhcb-comp.web.cern.ch/lhcb-comp/Simulation/Tutorial/01.GCorti-MCinLHCb-20091013.pdf.

[4] GITTA. *R-Trees*. URL: http://www.gitta.info/SpatPartitio/en/html/ObjOriDecomp_learningObject2.html.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[6] Antonin Guttman. "R Trees: A Dynamic Index Structure for Spatial Searching". In: vol. 14. Jan. 1984, pp. 47–57. DOI: 10.1145/971697.602266.

[7] Marios Hadjieleftheriou. *libspatialindex*. URL: https://libspatialindex.org/.

[8] Sean Gilles. Howard Butler Brent Pedersen. *Rtree: Spatial indexing for Python*. URL: http://toblerity.org/rtree/.

[9] IBM. *R-Tree Index Structure*. URL: https://www.ibm.com/support/knowledgecenter/en/SSGU8G_11.50.0/com.ibm.rtree.doc/sii-overview-27706.htm.

[10] INRIA and others. *scikit-learn*. Tech. rep. URL: https://scikit-learn.org/stable/index.html.

[11] Keras. *Keras: The Python Deep Learning library*. Tech. rep. URL: https://keras.io.

[12] Michael A. Nielsen. *Neural Networks and Deep Learning*. 2015. URL: http://neuralnetworksanddeeplearning.com/.

[13] Renato Quagliani. *Tracking for LHCb upgrade, using a full software trigger at 30MHz*. May 2018. URL: https://cds.cern.ch/record/2643444/files/Poster-2018-657.pdf.

[14] Renato Quagliani, Patrick Robbe, and Jonas Rademacker. "Study of double charm B decays with the LHCb experiment at CERN and track reconstruction for the LHCb upgrade". Presented 06 Oct 2017. Oct. 2017. URL: https://cds.cern.ch/record/2296404.

[15] T. Sjöstrand. *Monte Carlo Event Generation for LHC*. Tech. rep. Oct. 1991. URL: https://www.hep.phy.cam.ac.uk/theory/webber/MCEGforLHC.pdf.

[16] Google Brain team. *Tensorflow*. Tech. rep. URL: https://www.tensorflow.org/.

[17] F. Polci Y. Amhis O. Callot. *The Seeding tracking algorithm for a scintillating fibre detector at LHCb*. Mar. 2014. URL: https://cds.cern.ch/record/2260415/files/LHCb-PUB-2017-016.3.pdff.