Department of Physics and Astronomy
Heidelberg University

Master Thesis in Physics
submitted by

Sachin Gupta

born in Haldwani (India)

2024

Track Reconstruction Using Cellular Automata for the High Luminosity LHC

This Master Thesis has been completed by

Sachin Gupta

at the Physikalisches Institut, Heidelberg University

under the supervision of

Prof. Dr. André Schöning

Dedicated to

Maa (Rani Gupta), Papa (Yogendra Kumar Gupta),

and my sister Neelam Gupta

Abstract

The Large Hadron Collider (LHC) is set to reach unprecedented luminosity in the High-Luminosity LHC (HL-LHC) phase. This increase in luminosity will result in a substantial rise in the volume of data recorded by detectors. The surge in data will significantly increase the complexity of track reconstruction, especially for the central silicon-based tracker of general-purpose detectors like ATLAS and CMS. The TrackML dataset was introduced as a machine learning challenge on the Kaggle platform, generating a silicon-based tracker dataset that mimics conditions expected at the HL-LHC. Particle tracking, which involves grouping 3D space points into track candidates, becomes extremely challenging under high particle densities. The current state-of-the-art algorithm for tracking, the combinatorial Kalman filter, operates iteratively and dominantly relies on CPUs, though GPU-based implementation also exist.

This thesis explores the performance of a track-finding algorithm using cellular automata (CA), where triplets are employed as cells. The implementation has been carried out on CPUs, without GPU acceleration. The CA-based algorithm utilizes the Triplet Fit and Global Triplet Track Fit algorithms to assess the track quality at two stages, filtering out unwanted triplets and track candidates. Both the CA approach and the fitting algorithms are parallelizable, making them well-suited for acceleration on Graphics Processing Units (GPUs).

For this study, the barrel region of the TrackML dataset is selected, focusing on particles originating near the luminous region that deposit hits across all barrel layers, defined as signal tracks. The algorithm achieves an average perfect matching signal efficiency of $\approx 96\%$, with an impressive signal purity of $\approx 99\%$. These results demonstrate the potential of this parallelizable local tracking algorithm for handling the high pile-up conditions expected at the HL-LHC.

Zusammenfassung

Der Large Hadron Collider (LHC) wird in der High-Luminosity-Phase (LH-LHC) eine bisher unerreichte Luminosität erzielen. Diese Steigerung führt zu einem erheblichen Anstieg des Datenvolumens und trägt somit maßgeblich zur wachsenden Komplexität der Spurrekonstruktion bei, insbesondere in den inneren, siliziumbasierten Tracker von Universaldetektors wie ATLAS und CMS. Der TrackML-Datensatz wurde als Machine-Learning-Herausforderung auf der Kaggle-Plattform eingeführt und simuliert die Anforderungen an das Spurtracking, die am HL-LHC erwartet werden. Herkömmliches Tracking, bei dem 3D Raumkoordinaten zu Spur-Kandidaten zusammengefasst werden, gestalte sich aufgrund der hohen Teilchendichte als Herausforderung. Der derzeit führende Algorithmus zur Spurrekonstruktion, der kombinatorische Kalman-Filter, arbeitet iterative und ist überwiegend auf CPUs angewiesen, wenngleich auch GPU-basierte Implementierungen existieren.

Diese Arbeit untersucht die Leistungsfähigkeit eines Spurfindungsalgorithmus auf Basis von zellulären Automaten (CA), bei dem Triplets als Zellen verwendet werden. Die Implementierung erfolgt auf CPUs ohne Einsatz von GPU-Beschleunigung. Der Algorithmus verwendet einen lokalen Triplet Fit und einen globalen Triplet Track Fit, um die Qualität der Spuren auf zwei Stufen zu bewerten und unerwünschten Triplets, sowie Track-Kandidaten herausfiltern zu können. Sowohl der CA-Ansatz als auch die Fitting-Algorithmen sind parallelisierbar und daher gut für die Beschleunigung auf Grafikprozessoren (GPUs) geeignet.

Im Rahmen dieser Arbeit wurde die Barrel Region des TrackML Datensatzes genauer untersucht, wobei der Fokus auf sogenannten Signalspuren liegt. Diese umfassen Teilchen, die aus dem luminösen Bereich stammen und Treffer in sämtlichen Barrel-Lagen hinterlassen. Der Algorithmus erreicht eine durchschnittliche, perfekte Zuordnungseffizienz von ≈ 96 %, bei einer optimalen Reinheit von ≈ 99 %. Dieses Ergebnis demonstriert das Potential dieses parallelisierbaren lokalen Spurfindungsalgorithmus, zur Bewältigung der hohen Pile-up Bedingungen am HL-LHC.

Contents

Chapter 1

Introduction

1.1 High Luminosity LHC upgarde

The Large Hadron Collider (LHC) [1] at CERN is the largest scientific machine ever built to study the fundamental building blocks of our universe and their interactions. Major experiments at the LHC, such as ALICE [2], ATLAS [3], CMS [4], and LHCb [5], have been designed to investigate different aspects of high-energy particle collisions and probe the nature of matter at unprecedented scales. The LHC is currently preparing for the upgrade High-Luminosity LHC (HL-LHC) phase [6, 7] to achieve its full potential. The goal of this upgrade is to achieve unprecedented luminosity, increasing it by 5 to 7.5 times the current nominal value. Luminosity is a measure of the number of potential collisions per bunch crossing. For ATLAS and CMS experiments the average pileup is expected to reach around $\mu = 200$ for the HL-LHC phase. This means that for each bunch crossing, there will be an average of 200 proton-proton interactions occurring simultaneously. This high pileup environment will enable physicists to study phenomena with low probabilities, as it will provide the large data necessary for various rare processes in particle physics.

The timeline for LHC operation is shown in figure 1.1. During the HL-LHC phase, the total integrated luminosity is anticipated to be 4000 fb^{-1}. To give a sense of scale, 1 fb^{-1} corresponds to approximately 100 million million collisions. Foreseeing this, experiments involved in LHC are also going through significant upgrades [8]. Currently (October 2024), the Large Hadron Collider (LHC) is amid Run 3 [9], which began in July 2022. This run is expected to continue until 2026. In parallel with Run 3 operations, preparations for the High-Luminosity LHC (HL-LHC) upgrade are underway. The HL-LHC is expected to begin operations around 2029, following the completion of major upgrades to the accelerator complex and detectors [10].

1.2 Tracking Detectors

Tracking detectors are integral components of high-energy physics experiments, designed to record the trajectories of charged particles produced during collision. A range of technologies exists for tracking that employs gaseous, semiconductor, and fiber detectors [11]. They are capable of precisely measuring a particle position in the form of "hits". Multiple detector layers are used that constitute a tracking system that allows us to measure trajectory in the magnetic field which further helps to obtain the following information about the charged particle :

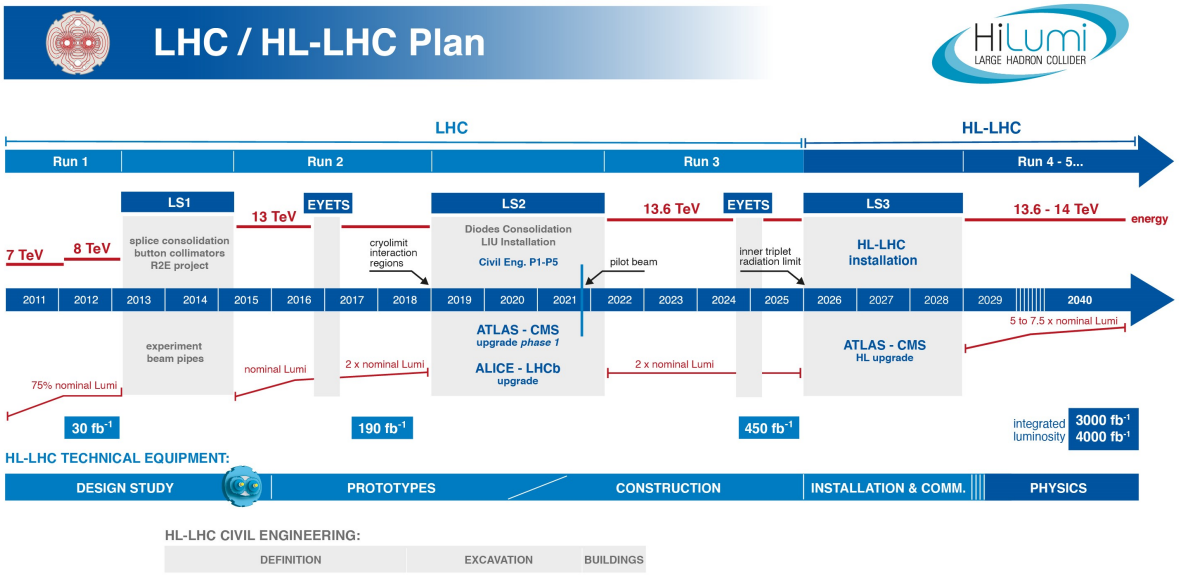


Figure 1.1: LHC timeline showing collision energy and luminosity evolution, including major upgrades during long shutdowns (LS2 and LS3) leading to the high-luminosity era. This plot is taken from [6]

1. **Measurement of Particle Kinematics**: Tracking detectors enable highly precise measurements of particle momenta and other kinematic properties, which are crucial for studying particle physics processes.

2. **Particle Identification**: The bending and the momentum help us to identify the particle type. For example in a process where a particle decays into a particle-anti-particle pair, one will have a positive bending and vice-versa.

3. **Vertex Reconstruction**: These detectors also allow us to reconstruct both primary and secondary vertices that play a crucial role in studying short-lived particles and decay processes.

This thesis focuses on track finding for silicon-based trackers, specifically, those employed in the central region of the ATLAS [3] and CMS [4] experiments. The analysis presented in this thesis is based on the TrackML detector [12] which was designed for the conditions expected during the Hl-LHC phase. The TrackML dataset [13] contains the simulated hits produced after collisions (Chapter 4).

1.3 Track Reconstruction

Track reconstruction is a crucial step in analyzing data from tracking detectors. It is a two-step process. The first step, Track Finding, groups the hits into track candidates and is known as a combinatorial problem. This step utilizes a dedicated pattern recognition algorithm. The second step, Track Fitting, estimates the parameters of each track candidate. This step employs statistical techniques that fit a theoretical model to the measured data, considering both hit positions and their uncertainties.

Tracking becomes particularly challenging in the high multiplicity environment expected during the HL-LHC phase. The most commonly used track reconstruction algorithm in HEP experiments is the combinatorial Kalman filter [14] which has an iterative nature, making it well-suited for implementation on CPUs, though it may not benefit much from being ported to GPUs. Given the potential of parallel computing hardware like GPUs [15] to improve computational efficiency, it is crucial to explore algorithms that can effectively leverage this hardware for better performance in the HL-LHC phase. This thesis explores a local tracking algorithm, utilizing methods that are both parallelizable and suitable for implementation on GPUs. The track-fitting approach is primarily used for quality metrics, while the final performance is analyzed specifically for the track-finding component based on Cellular Automata (CA). This thesis presents the physics performance of the track-finding algorithm and explores its performance for the barrel region of the TrackML dataset [12].

The structure of the thesis is organized as follows: Chapters 2 and 3 provide a theoretical overview of the algorithms for track finding and track fitting, respectively. Chapter 4 offers a brief review of the TrackML dataset and outlines the implementation roadmap for the track-finding algorithm. Chapters 5 and 6 present the performance of the algorithms at the triplet and the final track level. The discussion and outlook of the thesis are detailed in Chapter 7.

Chapter 2

Track Finding

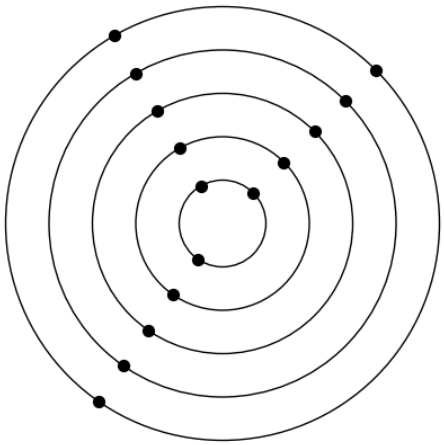
Track finding is a combinatorial problem that groups 3d space points i.e. hits into track candidates. Generating all possible combinations of hits becomes unfeasible due to a large number of hits recorded during a collision event. For this, dedicated pattern recognition algorithms are used that utilize information related to the detector geometry and physical information (particle's bending, particle's origin, etc.). There are various algorithms for this task: Hough transform, Legendre transform, Graph neural network, cellular automaton, Combinatorial Kalman Filter [11]. The Hough and Legendre transform are robust for shape detection but struggle with computational costs and simple patterns [11]. Graph neural networks [16] capture complex relationships yet demand extensive datasets and iterative message-passing steps. The Combinatorial Kalman Filter is also computationally intensive with iterative state estimation [14]. In contrast, cellular automata provide efficiency and flexibility with straightforward rules, making them well-suited for complex, noisy environments. This chapter discusses how we can utilise CA for track-finding step.

In an event [1] consisting of N hits, denoted as $\{h_1, h_2, \ldots, h_N\}$, the process of track finding attempts to identify sets of hits that correspond to distinct particle trajectories. After applying the track finding algorithm, the hits are grouped into subsets, each representing potential track candidates. These subsets can be written as $\{h_{i1}, h_{i2}, ..., h_{ip}\}$, $\{h_{j1}, h_{j2}, ..., h_{jq}\}$,$\{h_{k1}, h_{k2}, ..., h_{kr}\}$, ..., where subscript $i, j, k...$ corresponds to track candidate and p, q, r, corresponds to the number of associated hits within the track candidate. This procedure can also be visualized with the help of Figure 2.1. In this thesis, the cellular automata-based track finder is designed to find the longest track chain in the barrel region of the TrackML dataset [12].

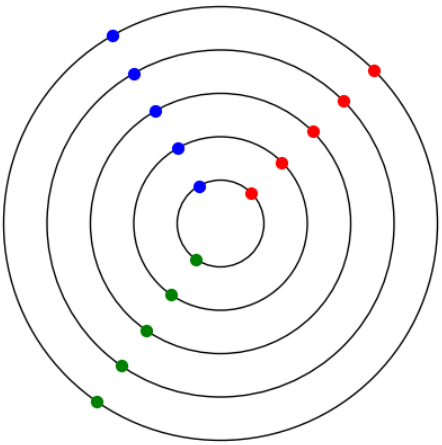
2.1 Cellular automata (CA)

Cellular automata (CA) were first introduced by John von Neumann in the 1940s [17] as a mathematical model for self-reproduction in biological systems. The concept was popularized by the work of John Conway's Game of Life in 1970 [18], which demonstrated how simple rules could lead to complex and dynamic patterns. Cellular Automata (CA) are simple mathematical models describing how a one or two-dimensional "grid" structure evolves in discrete time steps. The smaller grid unit is called a "cell" that carries a numeric value. In a single time step, all cells in the grid change their values simultaneously. The value of a particular cell at time step $t+1$ is determined by a fixed local rule that the entire

[1] An event is a collection of measurements of particles generated during a single bunch crossing.



(a) Recorded hits after an event. All hits are uni color that specifies they are not grouped yet.



(b) Hits after the track finding step. The hits are color-coded based on the track candidates to which they belong to.

Figure 2.1: Toy representation of track finding step. The detector has 5 layers. (a) shows the recorded "ungrouped" hits from three particles, while (b) displays "grouped" hits that belong to three different track candidates.

grid follows during the evolution. This local rule is determined by the 'local neighbors' of the cell. For example, given a cell i with value $c_i(t)$, the evolution is given as:

$$c_i(t+1) = f\left(c_i(t), \mathcal{N}(c_i)\right) \tag{2.1}$$

where $\mathcal{N}(c_i)$ denotes the set of local neighbors of the cell i and their values at time t. The neighbors and the law of evolution remain unchanged throughout time iterations. CA is constructed by defining a cell, its neighbors, and the local rule of evolution. The entire CA construction is problem-specific and gives full freedom to specify its structure and evolution. This is the reason why this simplest architecture has found applications in various fields, including Computer Science, Physics, Chemistry, and biology.

2.1.1 Example: The Glider Gun

The example presented here illustrates one of the early implementations of cellular automata and demonstrates how a simple model can lead to compelx results. Gliders are a fascinating phenomenon in cellular automata, particularly in Conway's Game of Life. They are patterns that move across the grid, demonstrating the ability of simple rules to produce dynamic and self-replicating behavior.

The glider gun, discovered by Bill Gosper in 1970 [19], was a groundbreaking pattern in the study of cellular automata. It was the first known finite pattern with unbounded growth, proving that the Game of Life could support infinite growth from a finite starting configuration. Following the definition of cellular automata discussed earlier, the glider gun setup is described below:

1. **Grid Initialization**: The grid is theoretically infinite, but in practice, it is represented by a large enough two-dimensional array. Each cell in the grid can exist in

one of two states: alive (usually represented by 1 or a filled square) or dead (0 or an empty square). In a 2D grid, each cell is surrounded by 8 neighboring cells that act as its neighbors.

2. **The Rule**: The glider gun follows Conway's game of life rules which are applied to each cell in the grid simultaneously:

   - Any live cell with fewer than two live neighbors dies (underpopulation).
   - Any live cell with two or three live neighbors lives on to the next generation.
   - Any live cell with more than three live neighbors dies (overpopulation).
   - Any dead cell with exactly three live neighbors becomes a live cell (reproduction).

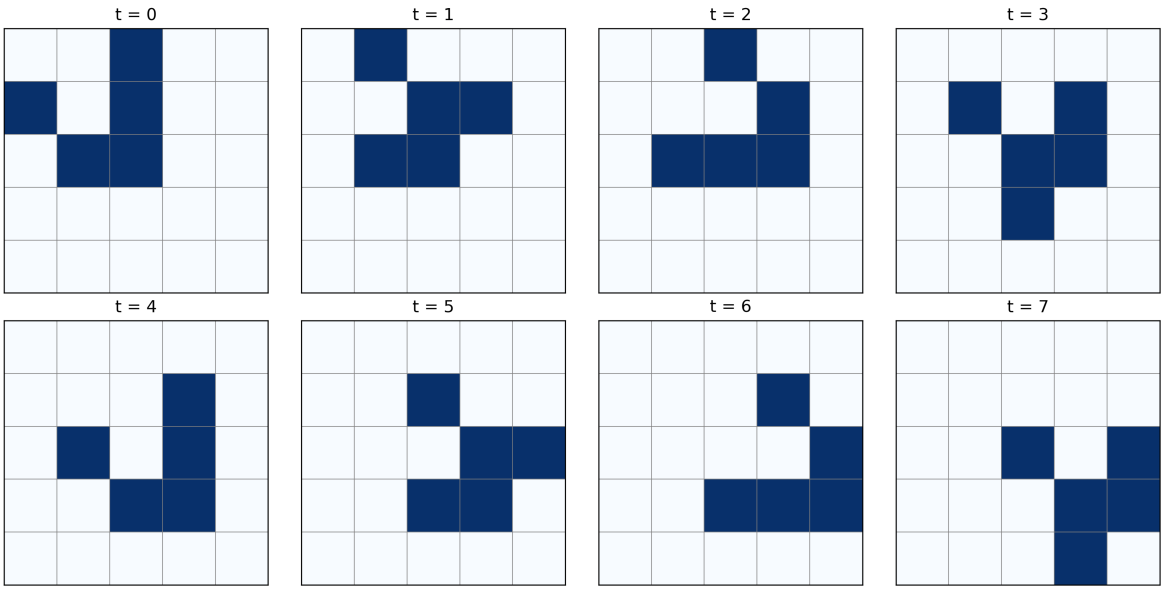


Figure 2.2: Evolution of the glider in Conway's Game of Life from $t = 0$ to $t = 7$. The grid is usually infinite, but for demonstration 5 x 5 grid is shown. The figure illustrates how the glider pattern moves diagonally and replicates itself over time after every four iteration.

The evolution in the Figure 2.2 shows the glider moves diagonally across the grid. It replicates itself after every four iterations, and this evolution continues indefinitely. This demonstrates how a simple pattern can evolve into a complex, self-sustaining structure.

2.2 Track Finding with Cellular Automata

The local nature of CA, and its simple implementation makes it suitable for particle tracking. In addition to their simplicity, CA algorithms can be efficiently accelerated using GPUs[20]. Earlier applications of CA in high-energy physics tracking were inspired by Conway's Game of Life, utilizing CA to filter out noisy hits. For instance, in the ARES experiment [21], detector hits were represented as cells, and the evolution process allowed

for the removal of noisy hits and the recovery of missing hits. Different versions of CA-based track finders exist for various high-energy physics experiments like ALICE [2], CMS [22] etc. The CA algorithm in this thesis is inspired by the CATS (cellular automaton for tracking in silicon detector) algorithm implemented for the HERA-B vertex detector [23]. The algorithm has also been extensively studied for the CBM experiment [24].
The CATS algorithm [23] performs track finding by considering track segments as a cell. A track segment is formed by grouping hits. In track reconstruction, the most commonly used track segments are doublets (a pair of two hits in consecutive layers) and triplets (three hits in consecutive layers). In the example presented below, hits are used as cells CA[2]. However, this does not introduce any fundamental difference, in the end, the track is formed by connecting hits or segments. The ultimate goal remains clear: "grouping of hits and forming track candidates". The example with five detector planes along with some hit configuration is shown in Figure 2.3. Specific to this detector geometry, the CATS algorithm forms track candidates as follows:

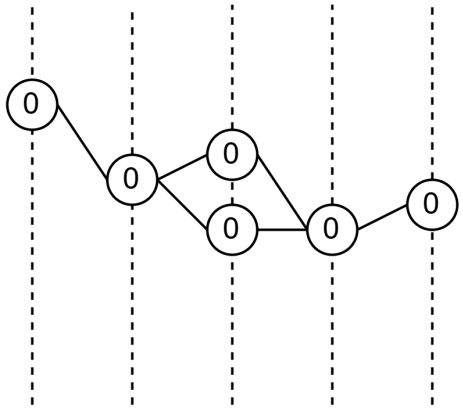


Figure 2.3: An example of five detector planes with a specific hit configuration is shown. The particle's direction is assumed to be from left to right, with each hit represented as a cell initialized to zero. Neighboring cells are connected by edges.

1. **Grid Initialization**:
   - **Cell**: Single detector hit is described as a cell.
   - **Neighbors**: All hits lying in the previous layer i.e. towards "left" plane. According to this definition, the first plane's detector hits will have no neighbors.
   - **Cell's value**: Each cell contains an integer value, and at the beginning of the grid, all cells are initialized to zero.

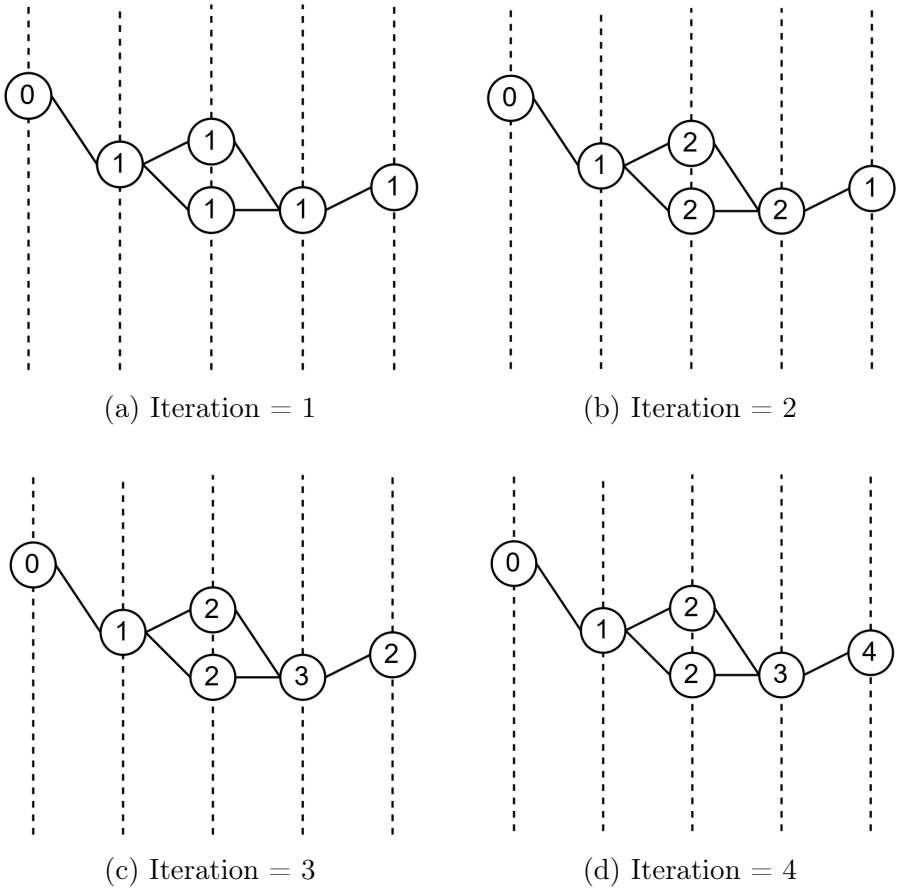
   The grid initialization is shown in Figure 2.3
2. **The Rule**: The evolution rule implemented in the CATS algorithm, serves to find the longest track chain with minimum scattering angle for the specific detector geometry. In this example, the longest track chain will consist of five hits. In each iterative step, a cell (c_i) examines its left neighbors and identifies the cell with the

[2]In this thesis, triplets have been used as a cell

maximum value [3] ($c_j^{\max}$). The new value for c_i is then obtained by incrementing this maximum value by one:

$$c_i^{new} = c_j^{max} + 1 \tag{2.2}$$

3. **Evolution and Track Collection**: The evolution of the CA is depicted in the Figure 2.4. The evolution process stops when no further updates are possible (Figure 2.4d). Track collection begins from the final configuration of the CA. In the final column of the CA grid, a cell with the highest value (in this case, 4) is selected. From this cell, a backward propagation step is implemented that collects all possible cells in the descending order $4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 0$. In this simple example, there are **two possible paths** and thus **two track candidates**.



(a) Iteration = 1 (b) Iteration = 2

(c) Iteration = 3 (d) Iteration = 4

Figure 2.4: CA evolution and cell values at different iterations. The evolution remains unchanged after the 4th iteration

We have successfully achieved our goal of grouping hits using cellular automata. However, further track selection is needed to finalize the best possible track candidates within an event. In this process, the CATS algorithm is complemented by a track fitting step, which includes the calculation of the χ^2 for the track candidates and selecting the track based on some threshold χ^2 value. There are also cases when track candidates may share hits. Such cases require additional treatment and are discussed in Chapter 6.

[3] This ensures the longest track chain is preferred

2.3 Triplet as a Cell

The example discussed in the previous section uses hits as cells in CA grid. This approach might not be suited for the high particle density environment where the total number of hits are expected to be $\mathcal{O}(10^5)$ per event [12]. Handling CA with many cells and neighbor connections will become computationally challenging and memory-expensive. Instead, we can utilize the flexibility of CA by redefining the “cell" and using “triplet of hits" as a definition for a cell. Three consecutive layer hits are combined to form a triplet. In the example given above, triplets will be formed between layers$(1, 2, 3)$, layers $(2, 3, 4)$, and layers $(3, 4, 5)$. The track will be formed by linking these three triplets as shown in Figure 2.5.

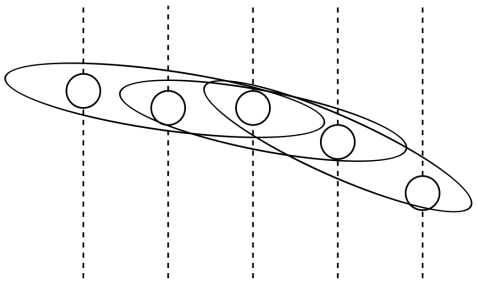


Figure 2.5: Track comprising of 5 hits and three triplets. Two consecutive triplets share one hit in common.

The chapter 5 discusses how triplets are generated with the physical cuts that allow certain configurations of triplets based on detector geometry and experimental setup. These triplets are further filtered by the quality metric χ^2 as discussed in the Section 3.2.1. The fitting algorithm is also parallelizable [25] and implemented at two levels: triplet level and track level. The next chapter discusses briefly the fitting procedure and fit quality χ^2 calculation.

Chapter 3

Track Fitting

Track fitting is the process of estimating a particle's parameters, covariance matrix, and track quality. This is a complex task that requires proper statistical treatment. The complexity arises due to factors such as the detector geometry, hit position uncertainties, multiple Coulomb scattering, and the influence of the inhomogeneous magnetic field. While the traditional Kalman filter, an iterative algorithm, is commonly used for track fitting, this chapter provides a qualitative overview of a non-iterative approach based on triplets is presented. In this thesis, the track fitting algorithm is used to assess the quality of a fit, represented by the χ^2, which is evaluated at two levels: triplet and track candiadate.

3.1 Genereal Idea

The general idea behind track fitting is to extract track parameters from the recorded n measurements $\mathbf{m_1}, \mathbf{m_2}, ..., \mathbf{m_n}$. The fitted track model $\mathbf{f(p)} = \mathbf{f_1}, \mathbf{f_2}, ...\mathbf{f_n}$ are obtained from the track parameters $\mathbf{p}$. The distance to be minimized is also known as χ^2 which is given by :

$$\chi^2 = (\mathbf{m} - \mathbf{f(p)})^T V^{-1} (\mathbf{m} - \mathbf{f(p)}) \tag{3.1}$$

Where the matrix V denotes the covariance matrix of the measurements. The χ^2 is also used as a quality metric. The χ^2 minimization method is not feasible for the final track fit due to several limitations. First, there is no closed-form solution for the track model due to the material interactions. The iterative nature of χ^2 minimization requires repeated matrix inversions, which become computationally intensive as the number of measurements increases. In contrast, the Kalman filter [11], archives the same minimization iteratively and is used widely for track-fitting [11]. The next section discusses a non-iterative algorithm that calculates χ^2 for triplets and track candidates, allowing for efficient processing suitable for implementation on GPUs.

3.2 Track Fitting

The track fitting implemented in this thesis is based on the algorithm presented in [25]. The track fit aims to fit particle momentum p and the hit position shift ($\delta\vec{x}_k = \vec{x}_{\text{fit},k} - \vec{x}_{\text{meas},k}$). These quantities carry full information about the particle's trajectory. The algorithm is divided into two parts: **Triplet Fit** and **Global Triplet Track fit**

(GTTF). The triplet fit is applied after building triplets with preselection cut (Chapter 5). The GTTF is applied to the track candidates generated by the cellular automata algorithm. The next two sections describe the qualitative description of the fit quality (χ^2) calculation.

3.2.1 Triplet Fit

Triplet fit is performed by minimizing the multiple scattering angles and hit uncertainty. These two terms are included in the triplet fit quality χ^2 whose formula is given by :

$$\chi^2(p;\vec{\boldsymbol{\delta}}) = \frac{\Delta\Theta_{MS}(p;\vec{\boldsymbol{\delta}})}{\sigma^2_{MS}(p)} + \sin^2\vartheta\frac{\Delta\Phi_{MS}(p;\vec{\boldsymbol{\delta}})}{\sigma^2_{MS}(p)} + \sum_{k=j}^{j+2}\vec{\delta x}_k^{\,t}{V_k}^{-1}\vec{\delta x}_k \tag{3.2}$$

Where $\vec{\boldsymbol{\delta}}$ represents hit position shifts of all hits in a triplet. The variables Θ_{MS} and Φ_{MS} are the polar and azimuthal scattering kink angles of a triplet calculated in their respective planes. The uncertainty in the scattering angle σ_{MS} is given by the Highland formula [26]. The azimuthal scattering term also contains an additional $\sin^2\vartheta$ arising from the transformation from cartesian to polar coordinates. The matrix V_k is a 3x3 covariance matrix that contains the spatial uncertainty of each k^{th} hit. The third term is a sum that runs over hits from layer j to $j+2$. The fit solution is achieved by minimizing Equation 3.2 with respect to momentum p[1] and the hit position shifts $\vec{\boldsymbol{\delta}}$.

The fitting approach described in [25] is based on linearization around a known circular solution that shows no multiple scattering kink angle in the transverse plane i.e. $\Delta\Phi = 0$. For the sake of brevity, the detailed calculation is omitted. The linearisation is performed for both kink angles, by also taking the effect of hit position shifts. The hit position shifts are also calculated in the local frame, which makes the error matrix V_k diagonal. The fitting method calculates the triplet parameters (p and $\vec{\boldsymbol{\delta}}$) and track quality χ^2.

3.2.2 Global Triplet Track Fit (GTTF)

The GTTF algorithm is applied to all track candidates obtained by the final configuration of the CA evolution. As mentioned in Section 2.3, a track can be assumed as joining consecutive triplets. The triplet parameters obtained from the Triplet fit algorithm, are utilized here to find the track parameters (p and $\vec{\boldsymbol{\delta}}$) and the fit quality χ^2. A track candidate with N hits can be visualized as $N-2$ linked triplets. The fit quality of a track is given by:

$$\chi^2 = \sum_{\text{triplet } j=1}^{N-2}\left(\frac{\Delta\Theta_{MS,j}(p;\vec{\boldsymbol{\delta_j}})}{\sigma^2_{MS,j}(p)} + \sin^2\vartheta_j\frac{\Delta\Phi_{MS,j}(p;\vec{\boldsymbol{\delta_j}})}{\sigma^2_{MS,j}(p)}\right) + \sum_{k=1}^{N}\vec{\delta x}_k^{\,t}{V_k}^{-1}\vec{\delta x}_k \tag{3.3}$$

The first sum with subscript j runs over all triplets and minimizes multiple scattering kink angle for the entire track. The second term with subscript k runs over all the hits in a track. We minimize the same terms as we did for the triplet fit. The triplet fit parameters are used in the minimization of the track's χ^2. These parameters are organized into a matrix, and the final minimized χ^2 value is obtained through a matrix multiplication, as outlined in [25].

[1] In the main paper [25] the minimization of χ^2 is done with respect to curvature $\kappa = \frac{qB}{p}$

Chapter 4

CA for the TrackML Dataset

4.1 TrackML dataset

The tracking machine learning challenge (TrackML [12]) was set up on the Kaggle [27] platform which hosts a variety of datasets for various problems in data science and machine learning. The dataset was simulated by LHC physicists to reach out to computer scientists to explore new approaches for tracking. The detector design was chosen independently of ATLAS and CMS upgrade tracker designs [[28],[29]]. The central region of the detector has a barrel-like geometry and the forward region has a disk-like geometry.

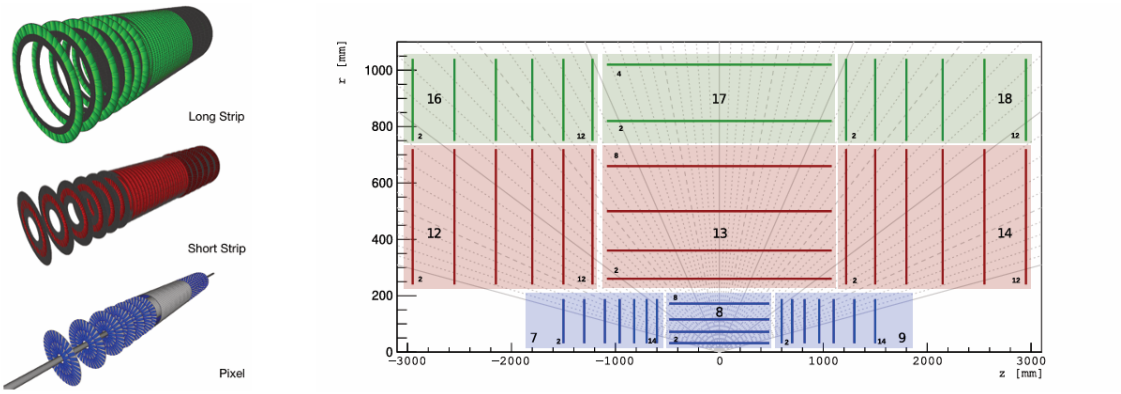


Figure 4.1: Detector layout of the TrackML detector. The left side displays the three main sub-detectors: pixel, short strips, and long strips. The right diagram illustrates the full layout and its coverage in the radial, longitudinal, and η directions. Different colors denote the various sub-detectors, with marked numbers indicating the internal volume and layer identifiers.

The full detector geometry is shown in Figure 4.1. The detector is split into three separate sub-detectors that differ in spatial resolution and material budget. The innermost sub-detector is a pixel detector with a spatial resolution of 50 µm x 50 µm. Surrounding it are two types of strip detectors: short strips with a resolution of 80 µm x 1200 µm and long strips with a resolution of 0.12 mm x 10.8 mm. The particle content of the collision was generated using the Pythia 8 event generator [30]. A hard QCD interaction that produces a $t\bar{t}$ pair serves as the signal. Additionally, 200 soft QCD interactions are overlaid to simulate the expected pile-up conditions at the HL-LHC. Generated charged

particles were propagated using fast simulation (FATRAS) of the ACTS software [31] by considering the magnetic field and material interaction. The fast simulation step does not include hadronic interactions. The TrackML dataset is challenging due to the high track multiplicity, with around $10,000$ charged particles producing $\mathcal{O}(10^5)$ [13] hits per event. This dense environment leads to significant overlap among particle trajectories, which complicates the track finding process.

The dataset contains of spatial coordinates (x, y, z) of hits in the global coordinates. Each hit has several identifiers for example volume id, layer id, particle id, hit id etc. that allow easy handling of the dataset and grouping hits. The particle id help us to identify weather a track candidate belong to a particle or not. If all hits within a track candidate have same particle id, that means a track candidate is an actual particle track.

4.2 Target Signal

The CA based algorithm implemented in this thesis is implemented for the **barrel region** of the TrackML dataset (volume $8, 13, 17$ in Figure 4.1) and for **specific tracks**. The algorithm is designed to track particles that originate near the luminous region and deposit hits in all 10 **barrel layers**. These tracks will be called **signal tracks** throughout this thesis. The signal tracks also satisfied cuts based on the particle's transverse momentum $p_T = \sqrt{p_x^2 + p_y^2}$ and origin (v_x, v_y, v_z). Both quantities are given in the dataset and cuts are selected to ensure they originate near the luminous region. The particle origin in the transverse $(x - y)$ plane is given by $r_0 = \sqrt{v_x^2 + v_y^2}$. The signal track cuts are listed in Table 4.1 where r_1 is the radius of the first barrel layer[1]. The performance of the CA algorithm, as discussed in Chapters (5 and 6) are evaluated concerning these signal tracks.

Parameter	**Constraint**
p_T	$p_T \geq 1\,\text{GeV}$
z_0	$\lvert v_z \rvert \leq 200\,\text{mm}$
r_0	$r_0 < r_1 = 32.30\,\text{mm}$

Table 4.1: Parameter Constraints for Signal Tracks.

The barrel region contains more than 50% of hits ($\approx$ 60,000) of an event. The average number of the signal tracks we are trying to reconstruct is around 250 per event constituting around $\approx 3,000$ hits. This explains the complexity of reconstruction. Not only do we have to filter out signal hits ($\approx 3,000$ hits from $\approx 60,000$ hits) but later group them into track candidates as well.

4.3 Track Finding with CA

The track-finding algorithm based on Cellular automata as described in Chapter 2 is implemented in the barrel region of the TrackML dataset [12]. The algorithm is also complemented by the track fitting (Chapter 3) algorithm that allows only a certain combination

[1]TrackML dataset [12] does not provide radii of the barrel layers explicitly. The radii for the barrel layers are calculated implicitly by the hits belonging to a particular barrel layer and taking the average.

of hits at triplet and track level by applying χ^2 cut. The track finding is performed sequentially as described by the following steps :

1. **Data selection**: The TrackML dataset is filtered for the barrel region by selecting hits belonging to the barrel layers which are given by the volume identifier: $[8, 13, 17]$ as described in Figure 4.1.

2. **Doublet Formation**: With the barrel dataset doublets of hits are formed with consecutive layers and preselection cuts. The procedure for doublet formation is described in Section 5.1.

3. **Triplet Generation**: Two consecutive doublets that share a hit are joined together to form a triplet that satisfies some preselection cuts. Triplet formation is described in Section 5.5.

4. **Triplet Fit**: Triplet fit is performed to all triplets and χ^2 cut is applied.

5. **Cellular automata**: Triplets that satisfy the χ^2 cut processed further to generate CA grid and track candidates are formed after the evolution.

6. **Track Selection**: The GTTF algorithm (Section 3.2.2) is applied to track candidates and χ^2 is calculated. The tracks are further selected by the χ^2 cut and ambiguity resolution (Section 6.3).

4.4 Software Environment and Libraries

The track-finding algorithm utilizing cellular automata is implemented in Python, leveraging various libraries to enhance performance and data handling. The following libraries and their versions are employed in this implementation:

Python: version 3.10.9 [32]. NumPy: version 1.23.5 [33], used for efficient numerical computations and array operations. Pandas: version 1.5.3 [34], which simplifies the manipulation and analysis of datasets, including the TrackML dataset provided in CSV format. Matplotlib: version 3.7.0 [35], utilized for data visualization and graphical representation of results. The use of these libraries facilitates efficient processing and analysis of the dataset, allowing for an effective implementation of the track-finding algorithm.

The tracking pipeline is available at the following web-link :
`https://gitlab.cern.ch/guptasa/cellular_automata`

Chapter 5

Triplet Generation

Triplets are track segments consisting of hits belonging to three consecutive layers. Before generating triplets, doublets of two hits are formed that belong to adjacent layers. Triplets are then built by joining two doublets that share a hit in the middle layer. Both track segments are formed by certain physical cuts that are discussed in great details in this chapter. After this step, the triplet fit algorithm calculates the track quality χ^2 for each triplet that further helps to reduce the number of reconstructed triplets.

5.1 Doublet Generation

Doublets are formed by pairing hits of two consecutive layers as shown in Figure 5.1. The TrackML dataset [12] has 10 barrel layers, thus doublets will be formed between layers $1-2, 2-3, ...9-10$. If the total number of hits in each layer are given $(N_1, N_2, ...N_{10})$, one can calculate the total number of possible doublets for the barrel region as :

$$\#\text{doublets} = N_1N_2 + N_2N_3 + ... + N_9N_{10} \tag{5.1}$$

To have a rough estimate, the number of hits in each barrel layer is expected to be of the order of $\mathcal{O}(10^4)$. Thus for one event, the total number of doublets will be around a **billion**, which is a lot. However, we do not need to create all possible doublets. Physical information such as detector geometry, magnetic field, and beamline constraints are used to only allow doublet with certain types. The next two sections discuss "physcical cuts" which are implemented in this thesis.

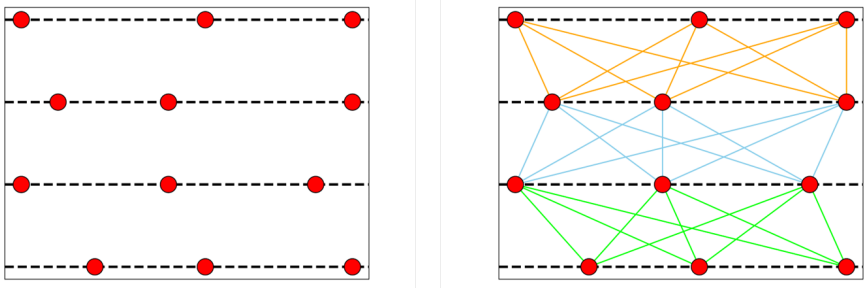


Figure 5.1: An example of four parallel detector planes with 3 hits each. The doublets are formed between layers and represented by an edge drawn between the hits. The total number of doublets in this example is 27.

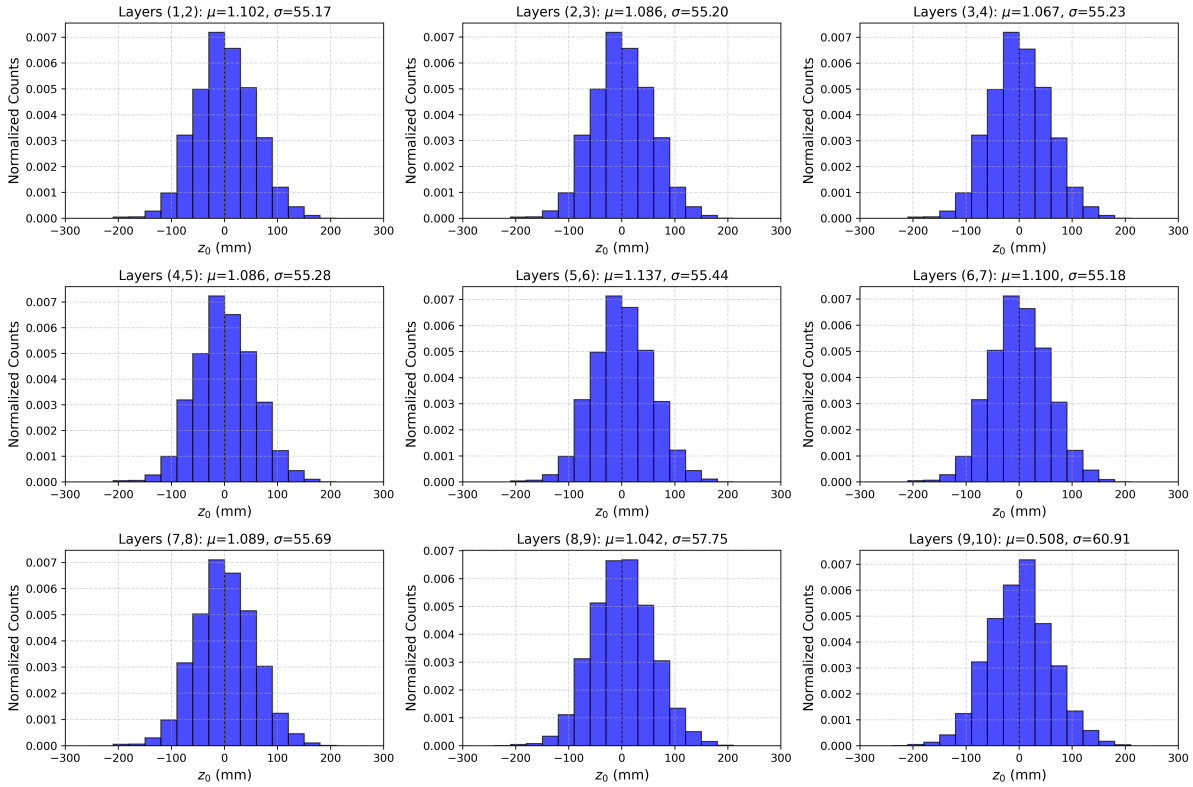


Figure 5.2: The normalized z_0 distribution for the signal tracks are shown for different layer combinations $(i, i+1)$. The distribution mean (μ) and standard deviation (σ) are displayed above each plot, alongside the corresponding layer numbers. For instance, "Layers (1,2)" indicates that the distribution corresponds to the true doublet between layer 1 and layer 2.

5.1.1 The z_0 based cut

The quantity z_0 defines the luminous region along the beam axis. Primary particles are produced along the beam axis from $-z_0$ to z_0. The beam axis is parallel to the z-axis. Since we are aiming to reconstruct the longest track chain in the barrel region as described in Section 4.2, a data-driven approach is utilized in this thesis. There is no bending in the longitudinal plane. The quantity z_0 is calculated for true doublets[1] as an intercept in the $(r-z)$ plane. For a true doublet, having hit coordinates (r_1, z_1) and (r_2, z_2), the z_0 is given by:

$$z_0 = \frac{r_1 z_2 - r_2 z_1}{r_1 - r_2} \tag{5.2}$$

The z_0 is calculated from true doublets and for each pair of layers the distributions are shown in Figure 5.2. From the distribution, Gaussian is fitted and (μ, σ) are obtained. This distribution is layer-dependent due to the cumulative effect of multiple coulomb scattering as charged particle traverses through barrel layers. These distributions eventually help to create a selection region while creating doublets between any two layers. The strategy implemented in this thesis is based on the selection of an integer n_{z_0} which essentially specifies layer-wise $(i, i+1)$ selection window $[\mu - n_{z_0}\sigma, \mu + n_{z_0}\sigma]_{i,i+1}$.

[1]True doublets are pairs of hits that correspond to an actual signal track (Section 4.2).

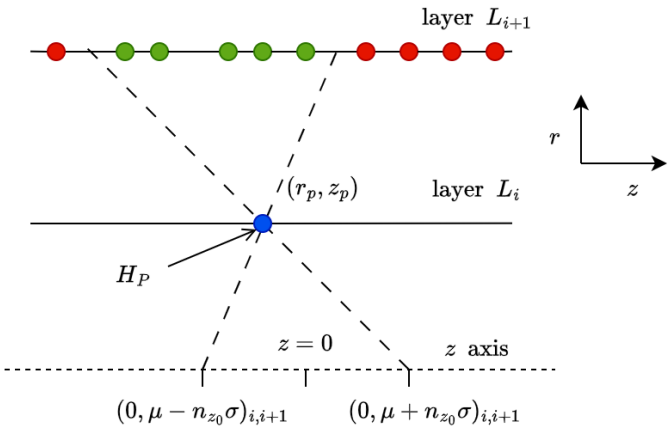


Figure 5.3: The longitudinal plane (r, z) view of two barrel layers L_i, L_{i+1}. The selection region for a particular hit H_p with coordinates (r_p, z_p) is shown by extending two lines from the z axis. The green hits in the next layer L_{i+1} satisfy the selection cut and proceed for further analysis. The red hits do not satisfy the selection cut and therefore do not form doublet with the hit H_p.

Now let us understand how this selection window helps to form doublets between two layers. Suppose we are at layer L_i and have a hit H_p with coordinates (r_p, z_p). We create a selection window in the next layer L_{i+1} by drawing two straight lines. The first line passes through the point $(0, \mu - n_{z_0}\sigma)_{i,i+1}$ and (r_p, z_p) extended to the next layers. Similarly, the second line is drawn through points $(0, \mu + n_{z_0}\sigma)_{i,i+1}$ and (r_p, z_p) and extended to the next layer as shown in Figure 5.3. The subscript $(i, i+1)$ specifies the layer-dependent cuts. The z coordinate of the intersection points of these two lines and the layer L_{i+1} is calculated. Once the selection window is mapped to the next layers only hits having z coordinates within this region are processed for further analysis. The next section discusses a second physical cut that is applied to the hits that are selected from the current step (Green hits in Figure 5.3).

5.1.2 The transverse momentum (p_T) cut

The final preselection cut implemented for doublet formation is based on the transverse momentum p_{Tcut}. This cut utilizes information from **beamline constraints** and **magnetic field**. In an ideal world, primary vertices will be along the origin of the transverse $(x - y)$ plane. The TrackML dataset is simulated where the magnetic field is along the beam direction (z axis) and has a central field strength of 2 Tesla [36]. Particles produced during the collision event traverse through the magnetic field which bends their trajectory. This bending is visible in the transverse plane $(x - y)$.

We aim to utilize the p_{Tcut} and find the angular section of layer L_{i+1} which contains hits that will form doublet with the hit H_p as discussed in the previous step. We draw two circles passing through the points $(0, 0)$ and the hit H_p. One circle represents the trajectory of a positively charged particle, while the other corresponds to a negatively charged particle. Both circles have the same radius, which is given by:

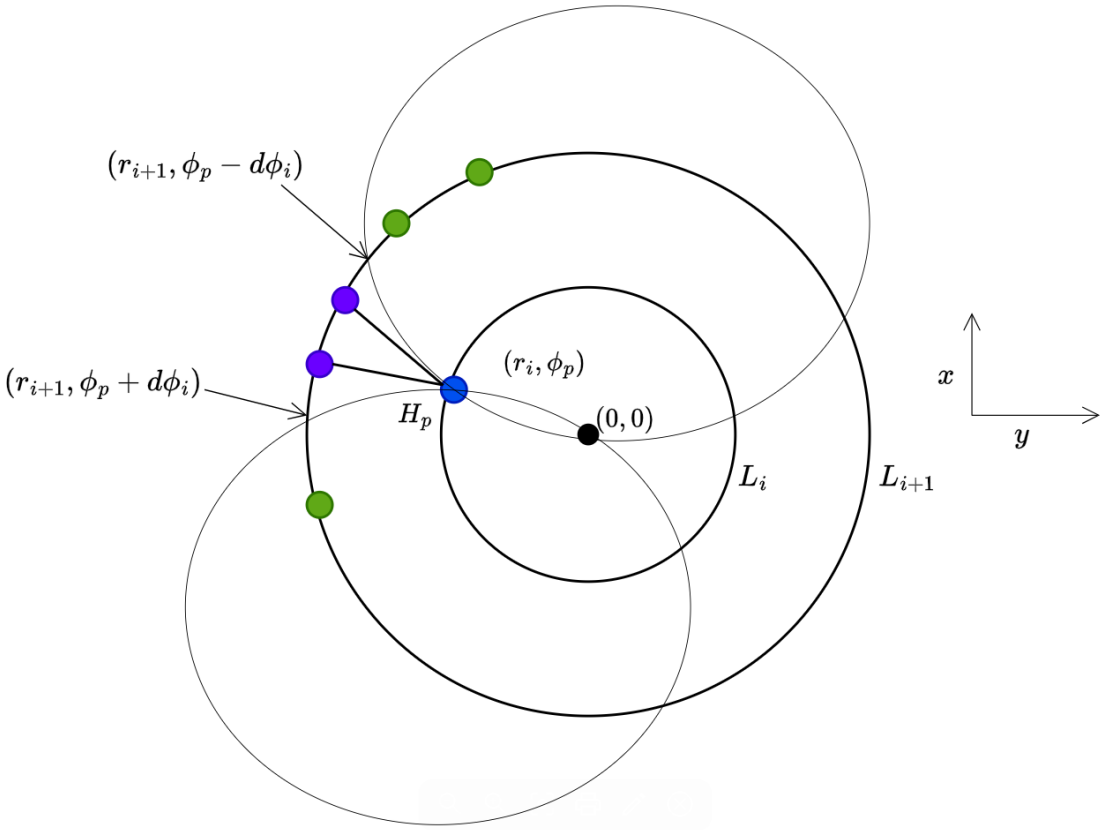


Figure 5.4: Representation of p_{Tcut} in the transverse (x, y) plane, barrel layers L_i and L_{i+1} are shown as two concentric circles. Two circles passing through the hit H_p and the origin $(0, 0)$ are shown along with their intersection with the layer L_{i+1}. Only two hits pass the cut from the green hits selected from Section 5.1.1. The chosen purple hits ultimately form a doublet with the hit H_p, represented by an edge connecting the selected hits and H_p.

$$p_{Tcut}[\text{GeV}] = 0.3B[\text{T}]r[\text{m}] \tag{5.3}$$

Where p_{Tcut} is given in GeV, magnetic field B in Tesla, and r is in meters. These two circles are extended to the next layer and two points in the next layer L_{i+1} eventually give the desirable section as shown in Figure 5.4. If the hit H_p has polar coordinates (r_i, ϕ_p), the selection window in the next layer will be from $(r_{i+1}, \phi_p - d\phi)$ to $(r_{i+1}, \phi_p + d\phi)$. Where r_i and r_{i+1} are radii of the barrel layers respectively. Note that in the TrackML dataset, the radii for the barrel layers are not stated specifically. For this, specific barrel layers are selected and then the radius is calculated by taking the average radius calculated from each hit coordinates (x, y, z) by the formula $r = \sqrt{x^2 + y^2}$

Note that, unlike the previous cut, this cut is automatically layer-specific. And we do not need to calculate its value from the signal tracks.

5.2 Efficiency and Purity

When building any segments i.e. doublets, triplets, or track candidates we need to keep a score of whether all **signal segments** are reconstructed or not. Efficiency and purity

are the key quantities that quantify the performance of any tracking algorithm. **Note that in this thesis only these scoring metrics are evaluated unlike the scoring scheme mentioned in the TrackML challenge [12]**. Typically, these two metrics are calculated concerning segments belonging to any track. However, in this thesis, they are calculated concerning segments that belong to **signal tracks**[2] (Section 4.2). This **efficiency** is the ratio of reconstructed signal segments to total signal segments.

(a) Truth Space

(b) Reconstructed Space

Figure 5.5: An example of segments (here doublets) in two cases. The green segments represent signal segments, while the blue ones correspond to non-signal segments. In (a) all segments are signal which means both efficiency and purity are 100% while in (b) efficiency can be 100% but not purity due to the presence of non-signal segments.

$$\text{Efficiency} = \frac{\#\text{reconstructed signal segments}}{\#\text{total signal segments}} \tag{5.4}$$

100% efficiency means that our algorithm captures all possible signal segments. While creating segments, certain segments may not belong to signal tracks as shown in Figure 5.5b. The goal is to reconstruct signal segments and avoid non-signal segments as much as possible. This is quantified by the **purity** that calculates how many reconstructed segments are signal and given by the ratio:

$$\text{Purity} = \frac{\#\text{reconstructed signal segments}}{\#\text{total reconstructed segments}} \tag{5.5}$$

The goal of any track reconstruction algorithm is to keep the reconstructed segment space as efficient and as pure as possible. Note that if we generate all possible combinations then the efficiency will be 100% but purity will be minimum.

5.3 Doublet cut selection n_{z_0}, p_{Tcut}

The cuts discussed in the previous sections are implemented for the barrel region of the TrackML dataset. Only muon (μ^+ and μ^-) tracks are used to select the optimal cut values n_{z_0} and p_T for doublet reconstruction. Once the target signal is obtained as described in Section 4.2, muon tracks are filtered out by selecting particles belonging to pdg id $[-13, 13]$. The idea behind this is to achieve 100% reconstruction efficiency for the muon tracks as they are less affected by the material interaction.

[2]Essentially, we calculate **signal efficiency** and **signal purity**, which will be used synonymously with efficiency and purity throughout this thesis.

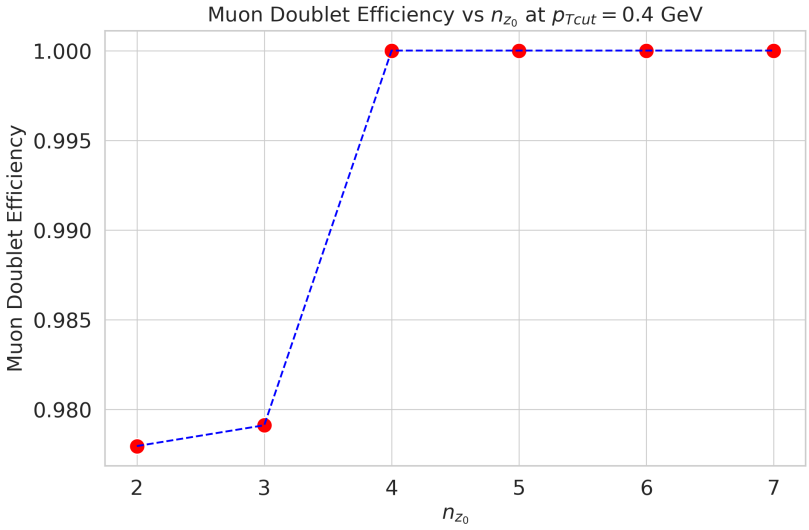


Figure 5.6: Muon reconstruction efficiency for the total of 862 signal muon doublets with $p_{Tcut} = 0.4$GeV. The layer-wise selection window $[\mu - n_{z_0}\sigma, \mu + n_{z_0}\sigma]_{i,i+1}$ expands as n_{z_0} increases, resulting in a corresponding increase in efficiency.

For the analysis purpose, we have chosen 90 events of the TrackML dataset. First, we select muon tracks that satisfy the signal definition (Section 4.2). A total of **61 muon tracks** are obtained that has **862 muon signal doublets** [3]. We aim to choose the cuts such that we are able to capture all of these doublets. To select the cuts, a differential approach has been carried out. First doublet formation is made almost independent of one variable and the efficiency metric (Eq. 5.4) is obtained by varying the second variable and vice-versa. The efficiency plot shown in Figure 5.6 is obtained by fixing $p_{Tcut} = 0.4$ GeV. This makes the angular selection region larger (Figure 5.4) and doublet formation is independent of p_{Tcut}. The figure shows we reach 100% efficiency when $n_{z_0} = 4$.

In the second case, we fix $n_{z_0} = 7$ and plot the efficiency by varying the p_{Tcut} as shown in Figure 5.7. We achieve 100% efficiency for $p_{Tcut} = 0.6$GeV. The efficiency starts decreasing after that value.

5.4 Doublet Metric

From the differential study of the previous section, by setting up the doublet cuts to $n_{z_0} = 7$ and $p_{Tcut} = 0.6$ GeV we get 100% efficiency for muon doublets i.e. we successfully reconstruct 862 muon signal doublets. We now utilize these cuts to form doublets in the barrel region of the TrackML dataset and the signal efficiency and purity are calculated for each event separately. The metric now is calculated with respect to true doublets of all particle type. The metrics are displayed as box plots for 90 events in Figure 5.8, accompanied by their mean values. These plots provide a visual representation of the range, showing where 50% of the data values fall, as well as the minimum and maximum values represented as *whiskers*. Additionally, total signal doublets and total reconstructed doublets per event are included in the same figure. The doublet cuts generate doublets with an average efficiency of 99.9% and an average purity of 0.5%. The mean number of

[3] One would expect this number to be $61 \times 9 = 549$. But a particle may deposit more than one hit in one layer giving rise to more number of doublets.

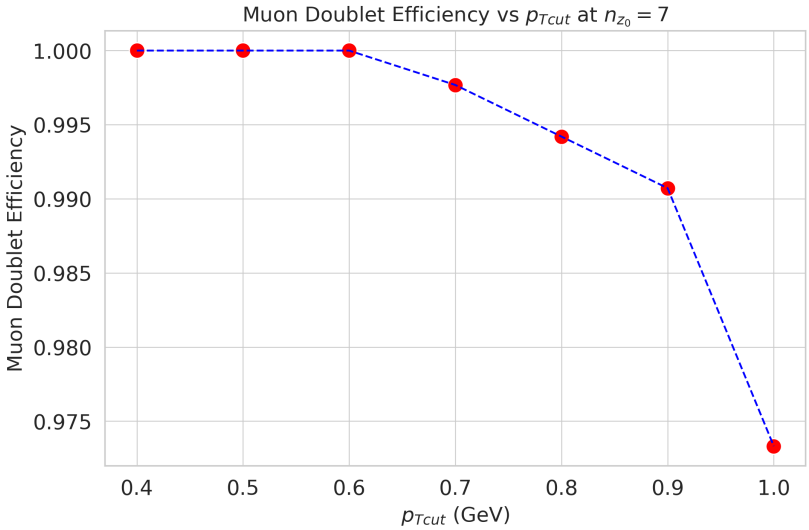


Figure 5.7: Muon doublet efficiency vs p_{Tcut} at the $n_{z_0} = 7$. The increase in p_{Tcut} results in a smaller angular window (Figure 5.4) and the efficiency decreases.

generated doublets ($\approx$ 600,000) is quite higher than the mean number of true or signal doublets ($\approx$ 3,000). Since this is the very first stage of building segments, we prefer having higher efficiency. These formed doublets are now processed to form triplets in the barrel region.

5.5 Triplet Formation

After creating doublets between all the layers with the optimal cut values, the next step is to join two consecutive doublets and form a triplet. A triplet is comprised of two doublets sharing a hit as shown in Figure 5.9. For ten barrel layers of the TrackML dataset, triplets are formed with hits belonging to layers: $L_1L_2L_3, L_2L_3L_4, ..., L_8L_9L_{10}$ the total number of possible triplets are given by :

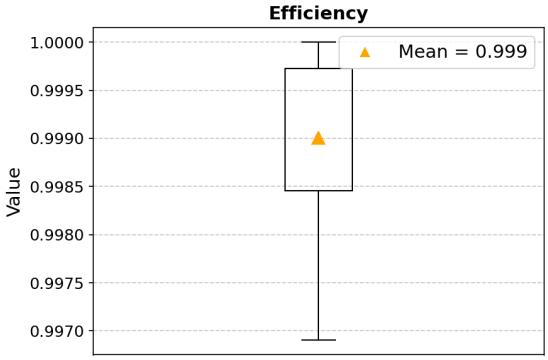
$$\#\text{triplets} = N_1N_2N_3 + N_2N_3N_4 + ... + N_8N_9N_{10} \tag{5.6}$$

where N_i is the number of hits belonging to the layer L_i. The total number of possible triplets for the barrel region of the TrackML dataset is of the order of $\mathcal{O}(10^{13})$ per event, given that the number of hits belonging to one barrel layer is $\mathcal{O}(10^4)$. Therefore, we apply two preselection cuts to a pair of consecutive doublets that share a hit in the middle layer, which allows only certain geometrical configurations.

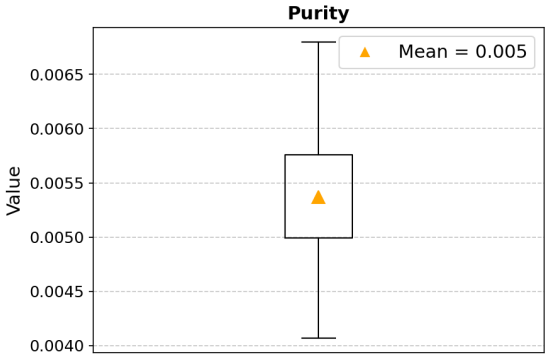
5.5.1 The Polar Angle Difference $d\theta$

This cut utilizes the information from the longitudinal plane $(r - z)$, where particle's trajecotries are straight line. The polar angle is defined as the angle made by a doublet with respect to z axis. A triplet that is formed by two doublets d_1, d_2 with polar angles θ_1, θ_2, the polar angle difference is defined as $d\theta = \theta_1 - \theta_2$. By following the similar approach as described in Section 5.1.1, this quantity is calculated for **true triplets**[4] in

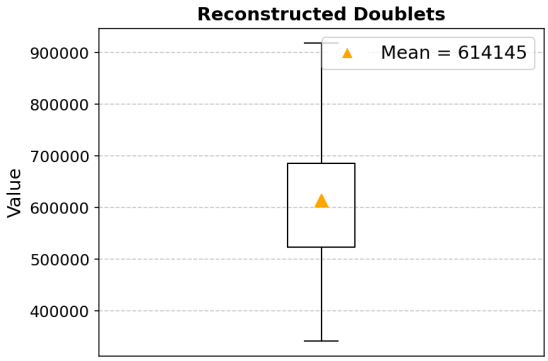
[4]A true triplet comprised of hits that belong to a signal track (Section 4.2).



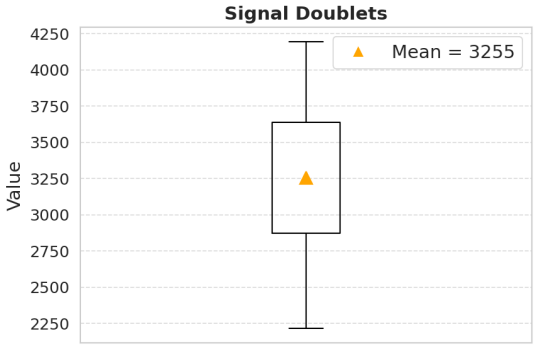
(a) Efficiency



(b) Purity



(c) Total Reconstructed Doublets



(d) Total signal doublets

Figure 5.8: Doublet metric and total reconstructed and signal doublets with doublet cuts $n_{z_0} = 7$ and $p_{Tcut} = 0.6$ GeV. Each figure is a box plot obtained for 90 events for the barrel region of the TrackML dataset. The rectangular box in each plot shows the region where 50% of the values fall. The upper and lower whiskers represent the maximum and minimum value obtained.

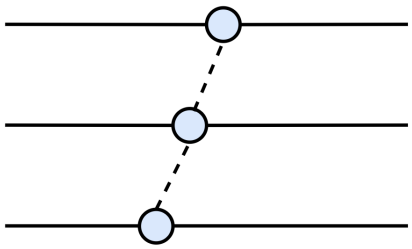


Figure 5.9: Visual representation of a triplet formed by two doublets sharing one hit in the middle layer.

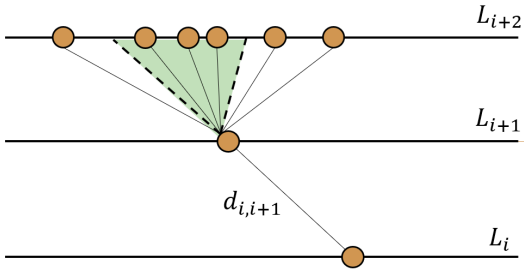


Figure 5.10: Triplet selection cut visualization with three detector layers in the longitudinal plane $(r - z)$. Doublet d_{i+1} shares a hit with the other 5 doublets. Only three of them satisfy the selection criteria $d\theta < [\mu - n_{d\theta}\sigma, \mu - n_{d\theta}\sigma]_{i+1}$ and are processed for the next selection cut.

a layer-specific manner and the distributions are shown in Figure 5.12. The TrackML dataset's barrel layers comprise of different material budgets [12] resulting in different amounts of multiple scattering across various layer combinations. The amount of multiple scattering (σ) increases as we move towards the end of the barrel layers.

The polar angle difference cut is implemented after selecting a positive integer $n_{d\theta}$ which gives a layer-wise selection window. Now let us try to understand how this cut filters out doublets. This can be understood with an easy example. Say we have three layers L_i, L_{i+1}, L_{i+2}. The doublet $d_{i,i+1}$ is a doublet between layer L_i, L_{i+1}. Doublets in the next two layers L_{i+1}, L_{i+2} sharing one hit with the doublet $d_{i,i+1}$ and polar angle difference $d\theta < [\mu - n_{d\theta}\sigma, \mu - n_{d\theta}\sigma]_{i+1}$ are selected for the next step. The selection can also be understood as a shaded angular region in between the layers as depicted in Figure 5.10. If no such doublets exist in the next two layers then no triplets are formed stemming from the doublet $d_{i,i+1}$.

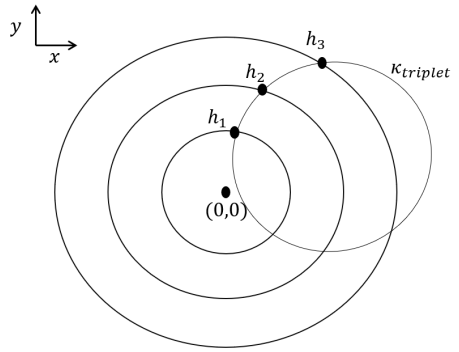
5.5.2 The Curvature Difference $d\kappa$

A pair of doublets passing through the criteria described above are then processed for a different cut. This cut is based on the transverse plane and uses beamline constraints and magnetic field bending information together. To calculate a curvature, a minimum of three points in the space are required. The curvature of a circle passing through three points in space $\vec{r_1}, \vec{r_2}, \vec{r_3}$ is given by the formula :

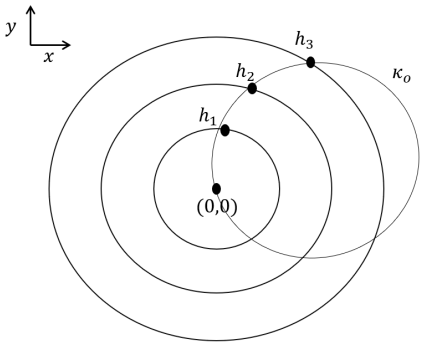
$$\kappa = \frac{2\,|\mathbf{r}_2 - \mathbf{r}_1| \times |\mathbf{r}_3 - \mathbf{r}_1|}{|\mathbf{r}_2 - \mathbf{r}_1|\,|\mathbf{r}_3 - \mathbf{r}_1|\,|\mathbf{r}_3 - \mathbf{r}_2|} \tag{5.7}$$

The curvature of a triplet $(\kappa_{triplet})$ can be calculated from the spatial coordinates of three hits. The second curvature incorporates the beamline constraint and is represented by $\kappa_{\mathcal{O}}$. The origin of the transverse plane and the first and third hit of the triplet are used to calculate $\kappa_{\mathcal{O}}$. The visualization of both curvatures and their respective circles are shown in Figure 5.11. The quantity that eventually help us to create a selection window in the transverse plane is the **curvature difference** [37] which is given by:

$$d\kappa = \kappa_{\mathcal{O}} - \kappa_{triplet} \tag{5.8}$$



(a) Calculation of $\kappa_{triplet}$ from the three hits h_1, h_2, h_3



(b) Calculation of $\kappa_{\mathcal{O}}$ from the three hits h_1, h_3 and the origin

Figure 5.11: Visualization of the two curvatures $\kappa_{triplet}$, $\kappa_{\mathcal{O}}$ and their respective circles.

Similar to the polar angle difference-based cut, we calculate curvature difference $d\kappa$ for true triplets. The layer-wise distributions are shown in Figure 5.13. Contrary to $d\theta$ distribution (Figure 5.12), $d\kappa$ distributions shrinks as we move farther from the beam axis. This arises because not only $d\kappa$ account for multiple scattering but include beamline-constriant as well. The curvature of triplets lying closer to the beam axis is larger than those triplets lying in the outer section of the barrel layers. That is why the distribution shrinks as we move towards outer region of the barrel. To implement this cut a positive integer $n_{d\kappa}$ is chosen that gives a layer-wise selection window. Thus for a specific layer combination L_i, L_{i+1}, L_{i+2} only those pair of doublets are allowed whose $d\kappa$ lies within the interval $[\mu - n_{d\kappa}\sigma, \mu - n_{d\kappa}\sigma]_{i+1}$ where μ and σ are obtained from the distributions shown in Figure 5.13

5.6 Triplet cut selection $n_{d\theta}$, $n_{d\kappa}$

Once we have generated doublets with cuts $n_{z_0} = 4$, $p_{Tcut} = 0.6$ GeV as mentioned in Section 5.3, they are joint together to form a triplet. For a triplet to be formed between two consecutive doublets, three conditions must be satisfied :

1. Two doublets must share a hit in the middle layer.
2. The polar angle difference $d\theta$ of doublets must lie within a selection window $[\mu - n_{d\theta}\sigma, \mu + n_{d\theta}\sigma]_{i+1}$
3. If the doublets satisfy the $d\theta$ cut, the curvature difference $d\kappa$ is calculated. If $d\kappa$ lies within the selection window $[\mu - n_{d\kappa}\sigma, \mu + n_{d\kappa}\sigma]_{i+1}$, then the triplet is formed between two doublets.

Thus with two numbers $n_{d\theta}$ and $n_{d\kappa}$ we can formulate triplets from doublets for the entire barrel region of the TrackML dataset. These cuts are selected by a differential approach and with the muon signal triplets, similar to the doublet cut selection (Section 5.3). There are a total of **963 muon signal triplets**. We select cuts such that we successfully reconstruct these 963 muon signal triplets. Figure 5.14 shows the efficiency of muon signal triplets for the different values of the cut $n_{d\theta}$ by fixing the second cut at

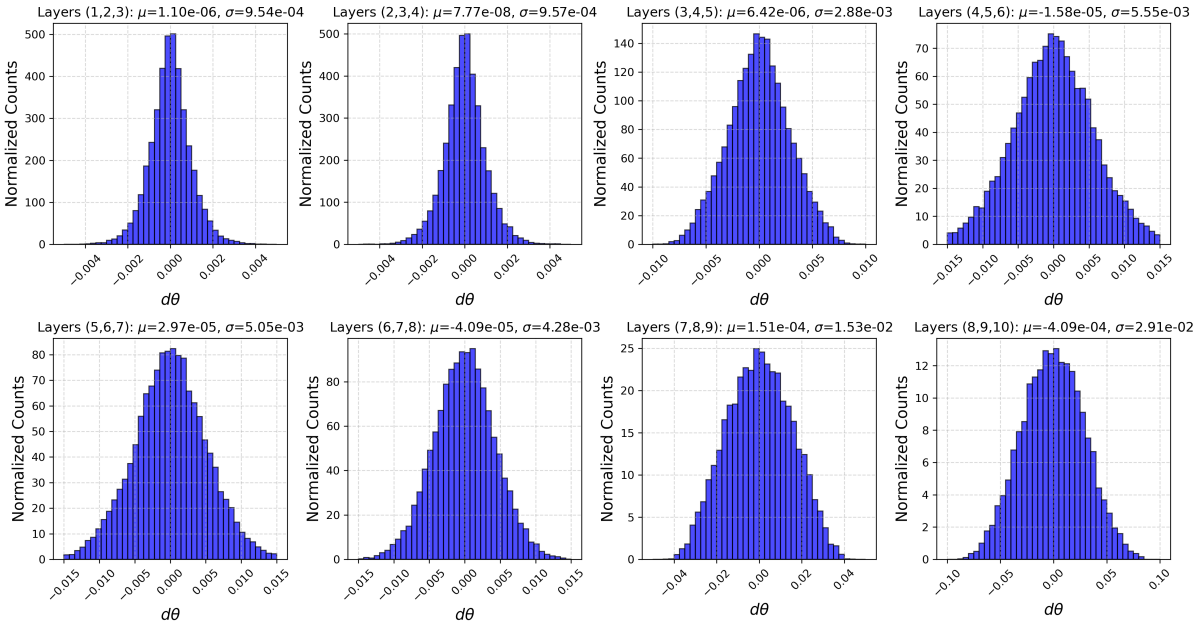


Figure 5.12: The layer-wise distribution of $d\theta$ for signal triplets in the TrackML dataset, with the mean and standard deviation displayed in each subplot title.

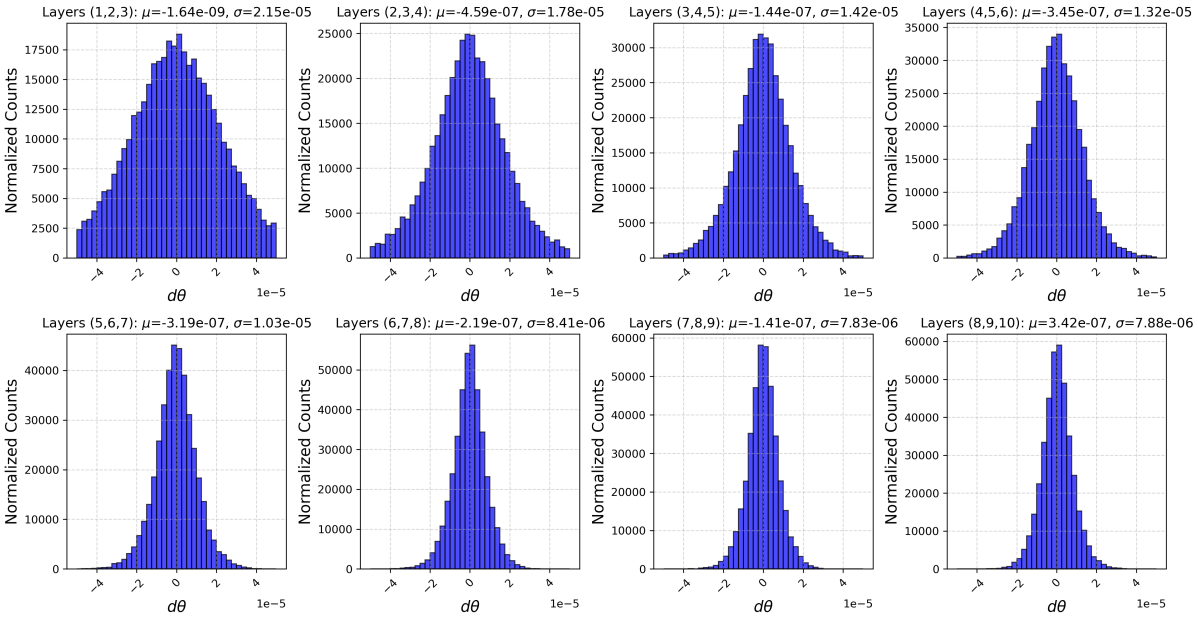


Figure 5.13: The layer-wise distribution of $d\kappa$ for signal triplets in the TrackML dataset, with the mean and standard deviation displayed in each subplot title.

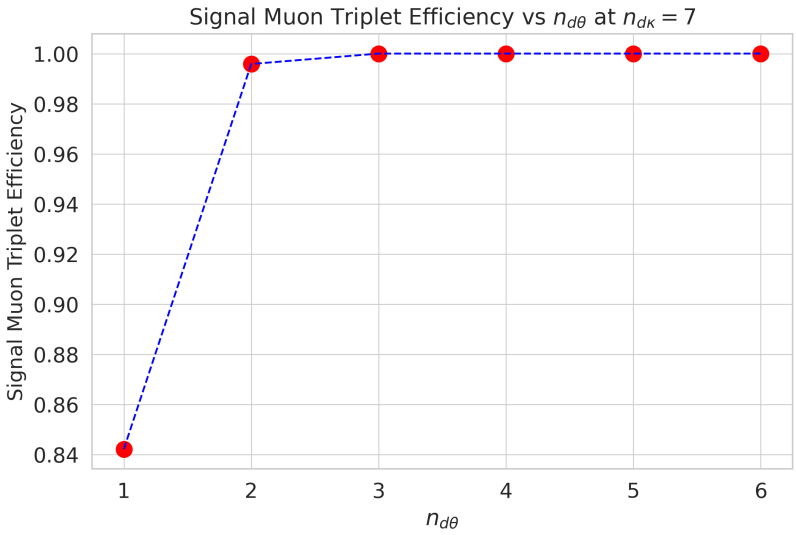


Figure 5.14: Muon signal triplets efficiency at different values of $n_{d\theta}$. The second cut is set large enough ($n_{d\kappa} = 7$) that makes the triplet formation independent of it.

higher value $n_{d\kappa} = 7$. The 100% muon signal efficiency at triplet level is achieved from values $n_{d\theta} \geq 3$.
The effect of the second cut $n_{d\kappa}$ at fixed $n_{d\theta} = 7$ is shown in Figure 5.15. In this case, 100% muon efficiency is achieved at $n_{d\kappa} \geq 6$. The cut values $n_{d\theta} = 3, n_{d\kappa} = 6$ gives 100% muon signal efficiency at triplet level and thus selected for generating triplets for all events of the TrackML dataset.

5.7 Triplet Metric and χ^2 cut

The differential study carried out for muon signals segments (doublets and triplets) in Section 5.3 and 5.6, gives the following values of four selection cuts for doublet and triplet formation :

Segment	Preselection Cut	Value
Doublet	n_{z_0}	4
Doublet	$p_{T cut}$	0.6 GeV
Triplet	$n_{d\theta}$	3
Triplet	$n_{d\kappa}$	6

Table 5.1: Preselection cuts for triplet formation.

These four cuts eventually help us to build and validate doublets which later are used to form triplets. The performance metric for triplets generation is also evaluated by the efficiency and purity as discussed in Section 5.2. For each event of the TrackML dataset, we build triplets in the barrel region and calculate signal efficiency and purity with respect to all signal triplets. The metrics are shown as a box plot in Figure 5.16. The average number of reconstructed triplets is around $\approx 154,000$ per event. These triplets are further processed to the fitting algorithm as discussed in Section 3.2.1 which calculates fit quality χ^2 for their early validation. The normalized χ^2 distribution for signal and non-signal

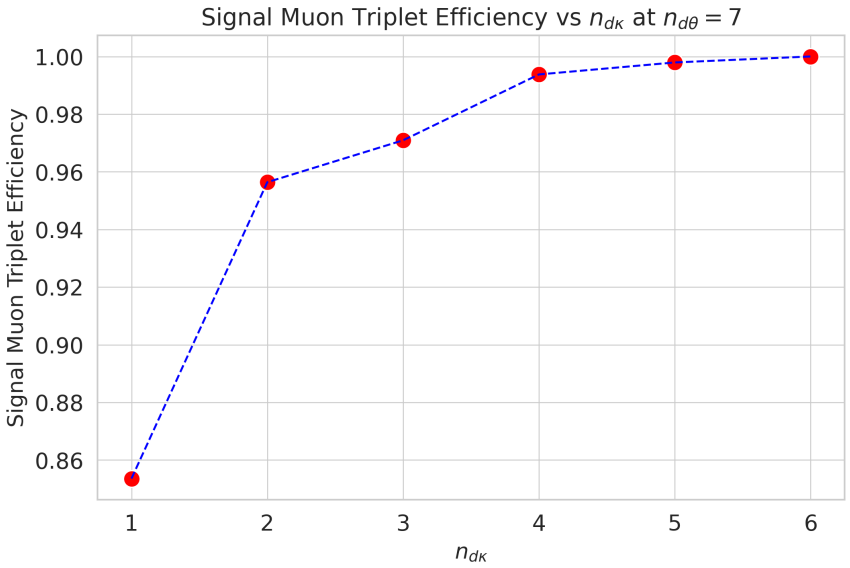


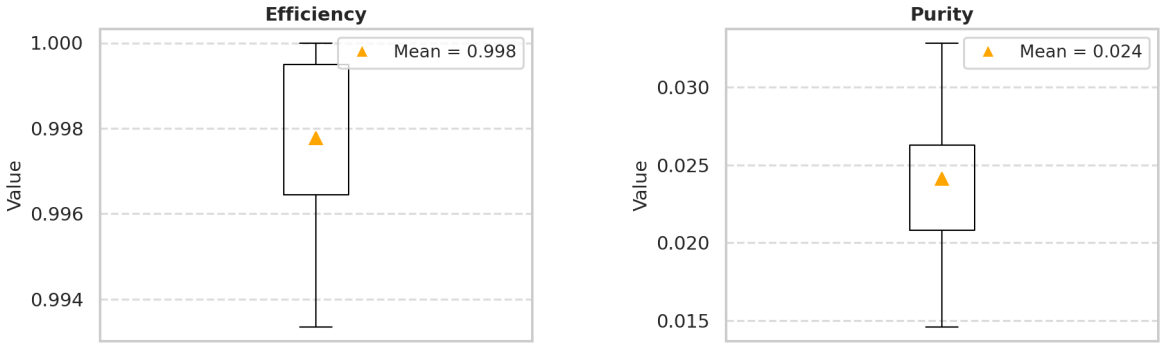
Figure 5.15: Muon signal triplets efficiency at different values of $n_{d\kappa}$. The first cut is set large enough ($n_{d\theta} = 7$) that makes the triplet formation independent of it.

triplets are shown in Figure 5.17. For a triplet, the **degrees of freedom** of χ^2 is **1**. Based on the distribution, the $\chi^2_{cut} = 6$ is selected, and triplets whose χ^2 is less than this threshold value proceed to track building stage with the cellular automata algorithm (Section 2.1).
The power of the χ^2 cut can be observed in Figure 5.18. The benefit of this step is that we keep the signal triplets as much as possible while discarding the unwanted triplets. This is why average efficiency remains almost the same and the average purity increases from 2.4% to 10.5%. Without applying the χ^2 cut, we would have had to process, on average, $\approx 154,000$ triplets per event. However, after the cut, this number is reduced to an average of about $\approx 34,000$ triplets ($\approx 80\%$ reduction) per event that will proceed to the next CA track-finding stage. The following table summaries the average number of generated segments and signal segments with preselection and χ^2 cut :

Segment	**# Signal Segments**	**# Reconstructed Segments**	**Average Efficiency**	**Average Purity**
Doublet	3,255	614,145	99.9%	0.5%
Triplet (before χ^2 cut)	3,526	153,953	99.8%	2.4%
Triplet (after χ^2 cut)	3,526	34,036	99.6%	10.5%

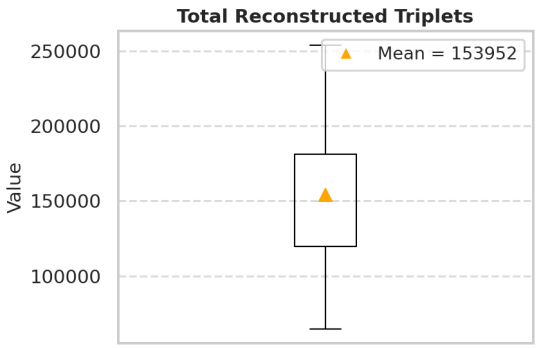
Table 5.2: Metric summary for different segments. The values are obtained by taking the average of 90 events of the TrackML dataset.

Triplets that pass successfully the χ^2_{cut} are used to create the CA- grid and evolution is done similarly to the process discussed in Section 2.1. The next chapter discusses how the tracks are collected and how do we get the best track by resolving ambiguity.

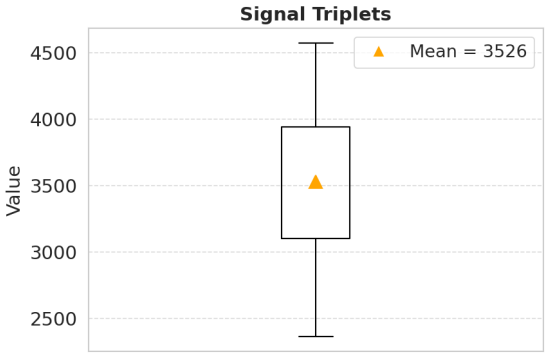


(a) Efficiency

(b) Purity



(c) Total Reconstructed Triplets



(d) Total Signal Triplets

Figure 5.16: Triplet metric along with reconstructed and signal triplets, calculated for the barrel region of the TrackML dataset. The triplets are formed with the preselection cuts described in Table 5.1. Each plot shows the ranges of values obtained for 90 events of the TrackML dataset.

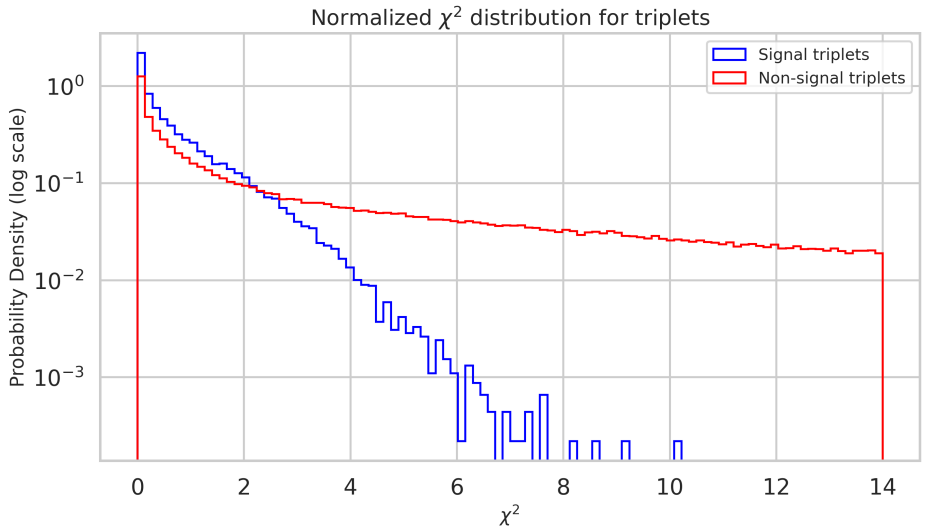
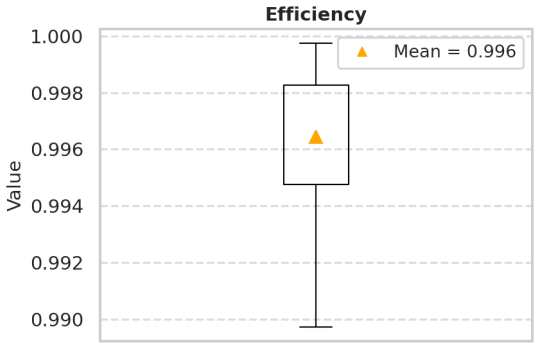
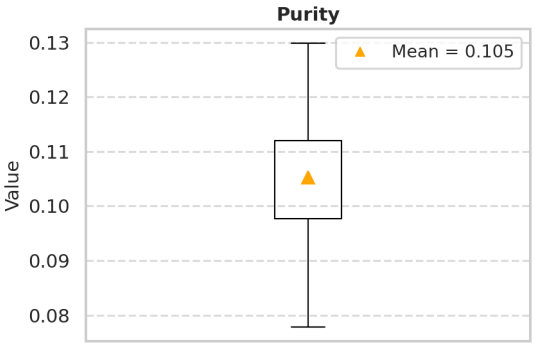


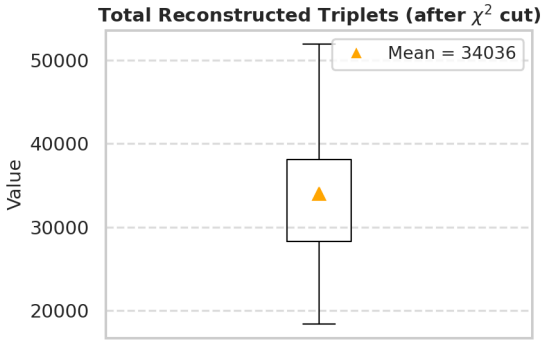
Figure 5.17: The normalized χ^2 distribution for signal and non-signal triplets, with the y-axis on a logarithmic scale. The peak of non-signal triplets around zero occurs because we also construct triplets with p_T less than 1 GeV due to the momentum cut on doublets (Section 5.1.2). Hence non-signal triplet may belong to an actual physical track but not signal tracks as described in Section 4.2.



(a) Efficiency



(b) Purity



(c) Reconstructed Triplets after χ^2_{cut}

Figure 5.18: Triplet metric after applying $\chi^2_{cut} = 6$. The efficiency remains almost similar as before applying χ^2 cut (Figure 5.16a) and the mean purity increase from 2.4% to 10.5%. The average number of triplets reduces to $\approx 34,000$ per event after applying the cut.

Chapter 6

Track Collection and Results

Triplets obtained after χ^2 cut are used to build CA grid. After the evolution, all possible track candidates are formed. This chapter discusses how we collect tracks from the final configuration of CA and select the best tracks based on χ^2, fitted momentum, and ambiguity resolution. Ultimately, we get a collection of track candidates that do not share any hits.

6.1 Track Collection

Once the CA evolution with triplets is performed and the final configuration is obtained for an event, a track collection step is applied that groups triplets into track candidates. The process is similar to the one discussed in Section 2.2. In the track collection step, a triplet is selected with a cell value[1] of 8 belonging to barrel layers $(8, 9, 10)$. From this selected triplet (say J), its neighbors are collected in descending order of cell values, and the tree-like structure is obtained as shown in Figure 6.1.

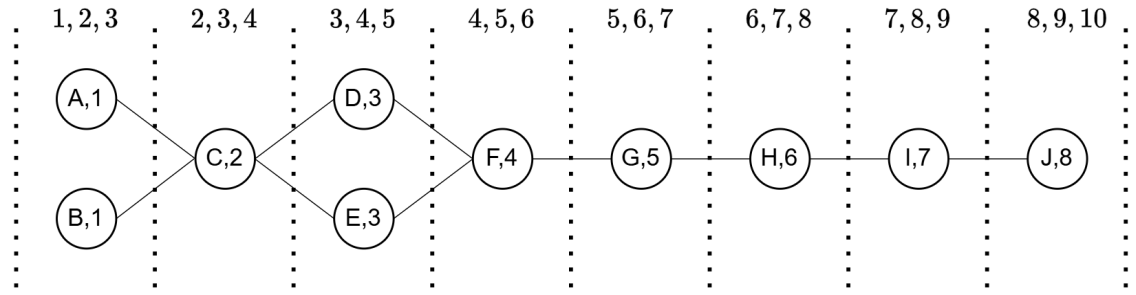


Figure 6.1: Tree structure originating from a triplet J in the layers (8,9,10). The circle represents a cell i.e. triplet with their letter identifier and the cell value. Above each cell, their respective layer combinations are shown.

There are a total of 4 track candidates that originate from triplet (J). The four possible tracks can be arranged in a 2-dimensional matrix where each row represents a track candidate. The obtained tracks from the configuration shown in Figure 6.1 is given by the following matrix :

[1]The maximum value of a cell belonging to layers $(8, 9, 10)$ can be 8 and we intend to find only the longest track chain.

$$\begin{bmatrix} J & I & H & G & F & D & C & A \\ J & I & H & G & F & D & C & B \\ J & I & H & G & F & E & C & A \\ J & I & H & G & F & E & C & B \end{bmatrix} \tag{6.1}$$

Note that the track candidates are now constructed with triplet identifiers. They can also be converted to their respective hit identifiers as well. The same procedure is done for all the triplets that has a cell value of 8 and belong to layers $(8, 9, 10)$. Once all possible track candidates from an event are generated the χ^2 for all track candidates are calculated using the algorithm described in Section 3.2.2. The χ^2 further helps to filter out the best track among track candidates that share one or more hits/triplets.

6.2 CA Performance Evaluation

Once all possible track candidates from the CA algorithm are reconstructed, the algorithm performance is evaluated by signal efficiency and signal purity as discussed in Section 5.2. The goal is to reconstruct as many signal tracks (Section 4.2) as possible. To analyze the performance two matching schemes have been used in this thesis :

1. 50% **Matching**: If within a reconstructed track candidate, more than 50% of hits match to a signal track, then it is counted as a successfully **reconstructed signal track**. If two or more track candidates correspond to the same signal track, the reconstructed track is counted only once, avoiding any double-counting.

2. **Perfect Matching**: If within a reconstructed track candidate all hits (100 %) correspond to a signal track, then only it is counted as a successfully **reconstructed signal track**.

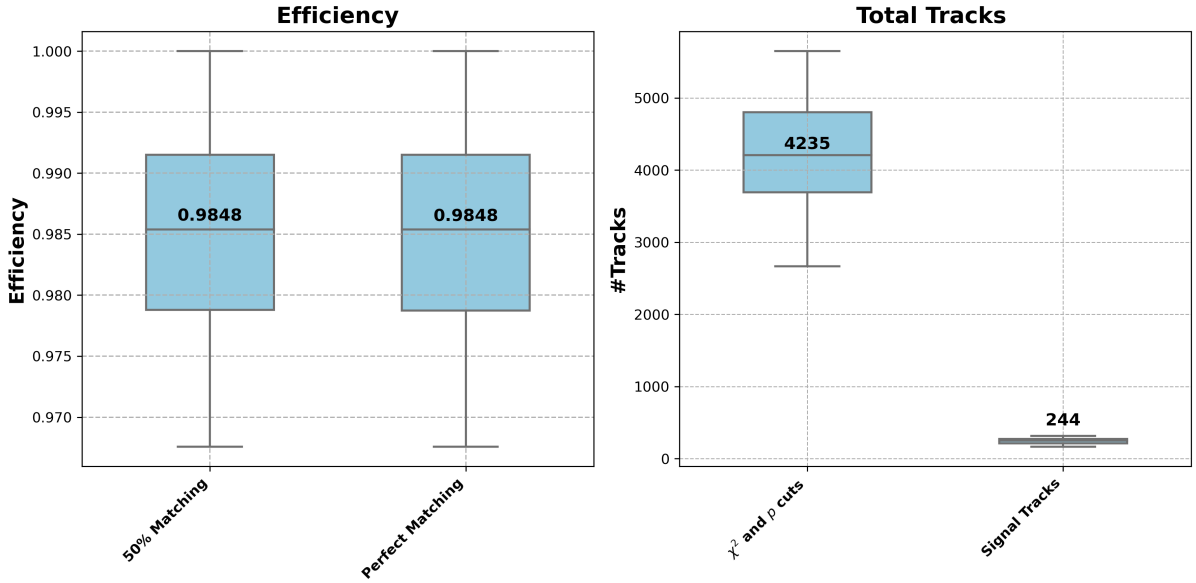


Figure 6.2: Performance metric of CA algorithm implemented on triplets satisfying χ^2 cut (Figure 5.18), along with their mean value for 90 events of the TrackML dataset. The efficiency is shown for two matching i.e. 50% and perfect matching. The second plot shows the total tracks reconstructed by CA and the actual signal tracks.

Therefore both efficiency and purity will be calculated for two matching criteria. In both definitions only the idea of when a track candidate is called “signal" changes. The efficiency for an event is calculated by the following ratio:

$$\text{Track Efficiency} = \frac{\#\text{reconstructed signal tracks}}{\#\text{total signal tracks}} \tag{6.2}$$

At the early stage of performance evaluation of CA, the quantity **purity** can be misleading due to the presence of tracks sharing one or multiple hits and belonging to one signal track, it is calculated during the final steps when reconstructed track candidates do not share any hits among them. The purity is determined using a similar formula as shown in Equation 5.5 :

$$\text{Track Purity} = \frac{\#\text{reconstructed signal tracks}}{\#\text{total reconstructed tracks}} \tag{6.3}$$

For the 90 events of the TrackML dataset, the performance plots are shown in Figure 6.2. The average efficiency is 98.5% which is same to both matchings. This shows the capability of CA algorithm to reconstruct a significant amount of signal tracks. However, the number of reconstructed tracks per event is almost 18 times higher than the actual signal tracks. This discrepancy arises because tracks share hits or triplets. The next section discusses how to filter the best tracks from the reconstructed ones without significantly impacting the efficiency.

6.3 Ambiguity Resolution

The reconstructed tracks by CA are further analyzed to reduce the combinatorics. For this, the track fit algorithm GTTF (Section 3.2.2) is implemented for all tracks and their χ^2 and fitted momentum p is calculated. Based on these values the ambiguity resolution is performed. The next two sections describe them briefly.

6.3.1 χ^2 and Momentum p cut

The normalized χ^2 distribution for signal tracks and all tracks (obtained after CA) is shown in Figure 6.3. The GTTF algorithm also gives the fitted momentum value after minimization of χ^2 [25]. The degrees of freedom for track fitting in 10 barrel layers are **8**. The momentum p distribution is shown in Figure 6.4. Based on the signal track distribution the following cuts are applied to the tracks obtained from CA.

$\chi^2 < 45$
$p \geq 1\,\text{GeV}/c$

Table 6.1: Selection cuts for track candidates

Only tracks satisfying these cuts are processed for further analysis. The track metric is calculated for the selected tracks. The performance of the selection cut is shown in Figure 6.5. The mean value of efficiency for both 50% and 100% matching, reduces by the small amount. The average total number of track candidates after applying the cuts is nearly halved. The selected tracks are now proceed for further analysis.

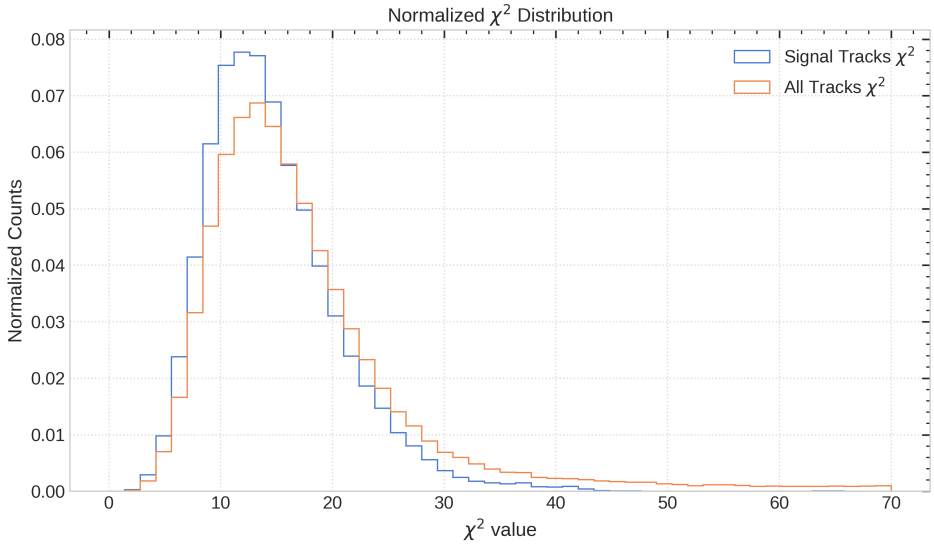


Figure 6.3: Normalized χ^2 distribution (degrees of freedom = 8) for signal tracks and all tracks (signal+non-signal tracks) obtained by implementing GTTF (Section 3.3) on the tracks reconstructed by the CA algorithm (Fig. 6.2).

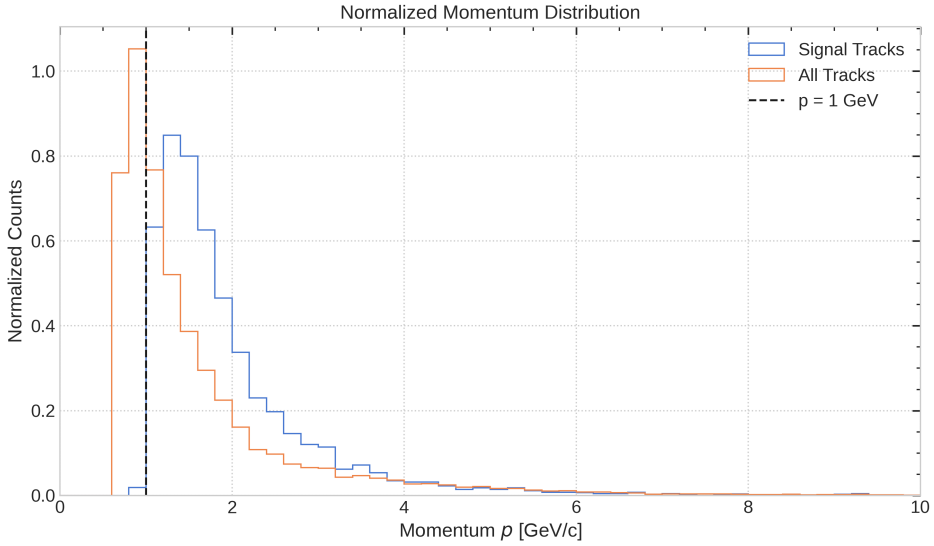


Figure 6.4: Normalized momentum p distribution for signal tracks and all tracks (signal+non-signal tracks) obtained by implementing GTTF (Section 3.3) on the tracks reconstructed by the CA algorithm (Fig. 6.2).

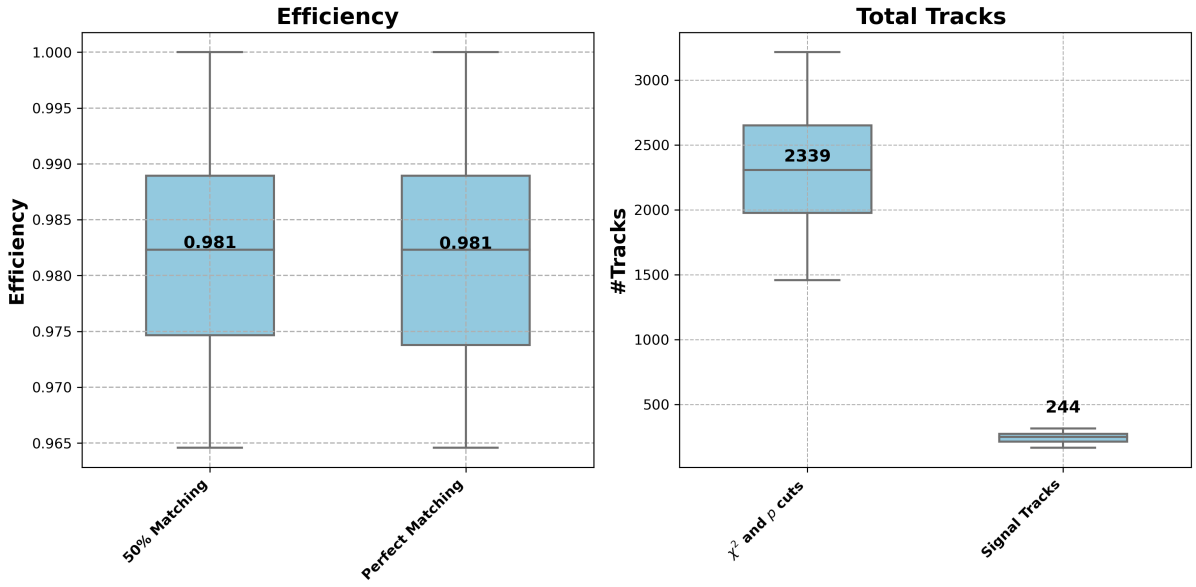


Figure 6.5: Track metric after applying χ^2 and momentum cuts described in Table 6.1

6.3.2 Track Trimming

After applying selection cuts, we address the problem of track candidates sharing multiple triplets among them as described in Matrix 6.1. The tree structure emerges because the CA algorithm (CATS [23]) implemented in this thesis lacks terminal points at both ends of the grid. The trimming step refers to a **selection of one best track from tree-like structure** as shown in Figure 6.1. The example shown has the total number of four possible tracks, we select the track that has minimum χ^2. We perform the same step for all the triplets lying in the last region of the barrel (layers $(8, 9, 10)$). After applying this we get the track candidates that do not share any triplet in the outermost region of the barrel. These filtered tracks are further processed to perform similar trimming from the other end of the barrel i.e. Layers $(1, 2, 3)$. After this step, the filtered track candidates do not share any triplets from both ends. The trimming step for the track candidates can be performed in two ways as listed below :

1. Inside-out: The track candidates obtained after selection cuts (Table 6.1) are first trimmed from the innermost region of the barrel i.e. layers $(1, 2, 3)$. The filtered tracks are then trimmed from the outermost region of the barrel (Layers $(8, 9, 10)$). The track metric is shown in Figure 6.6.

2. Outside-in: In this approach, tracks are trimmed from the outermost region of the barrel (Layers $(8, 9, 10)$) first and then from the inner region (Layers $(1, 2, 3)$). The metric for this approach is shown in Figure 6.7

Note that **both approaches are symmetric** and almost perform similarly after both trimming steps are implemented. The average 50% matching efficiency remains similar to the metric we get after the selection cuts (Figure 6.5). However in both cases, after the second trimming, the average perfect matching efficiency is 97.6%. Which is almost a 1% reduction compared to CA metric. The average number of tracks at the final stage in both cases is around $\approx$ 300 which is closer to the number of average signal tracks $\approx$ 243. Thus we have reduced the total number of track candidates without affecting both efficiencies significantly.

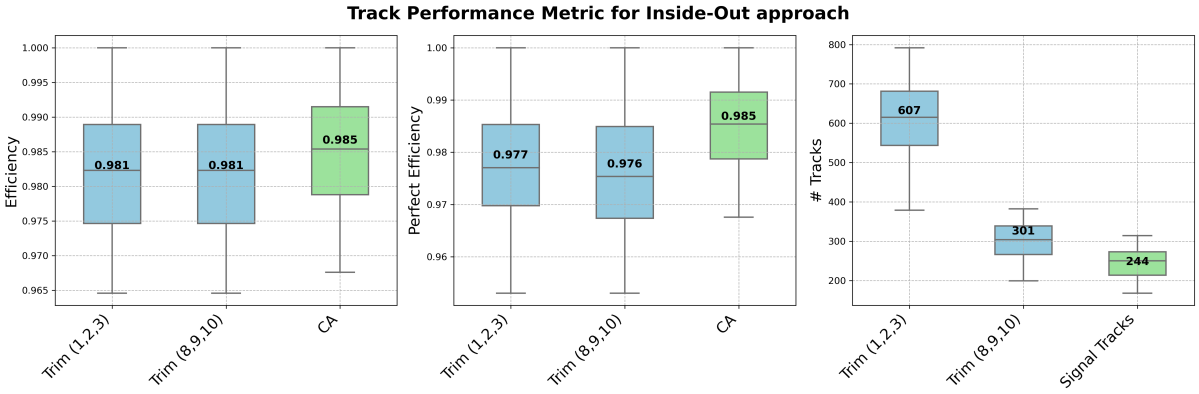


Figure 6.6: Performance metric for Inside-out approach after applying selection cuts (Figure 6.5) along with their mean values. The plots show the sequential effect of trimming performed first from layers (1,2,3) and then layers (8,9,10). The light green box in the first two efficiencies plots corresponds to the CA metric (Figure 6.2) while in the third plot it corresponds to the signal tracks we want to reconstruct.

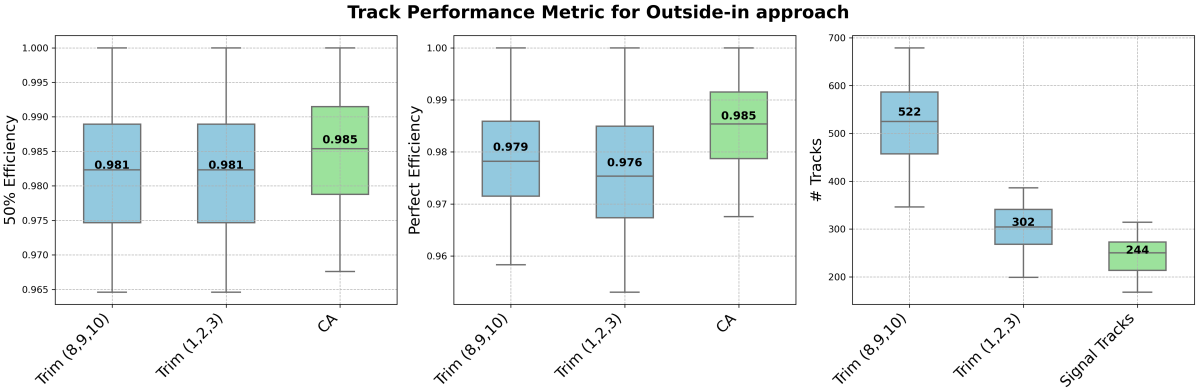


Figure 6.7: Performance metric for Outise-In approach after the selection cuts (6.5)) along with their mean values. The plots show the sequential effect of trimming performed first from layers (8,9,10) and then layers (1,2,3). The light green box plot in the first two efficiencies plots corresponds to the CA metric (Figure 6.2) while in the third plot it corresponds to signal tracks we want to reconstruct.

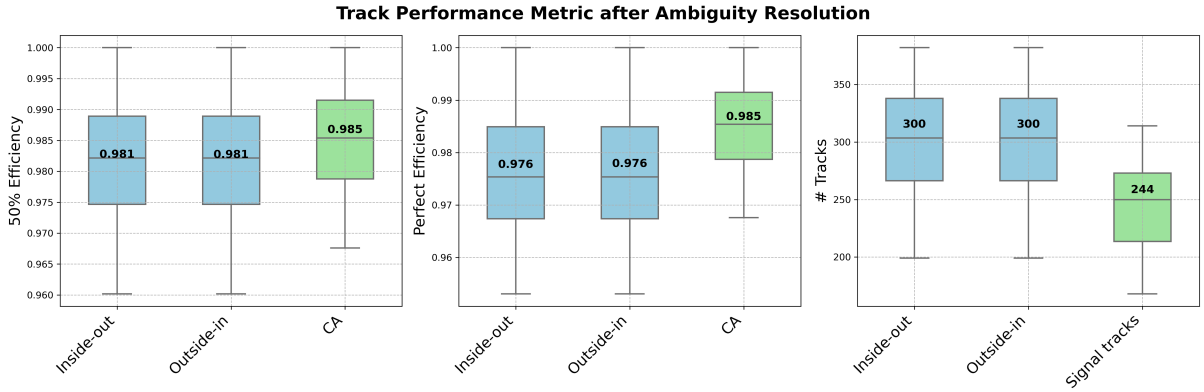


Figure 6.8: Track performance metric after sequentially applying steps to tracks reconstructed by CA: selection cuts $(\chi^2, p) \rightarrow$ trimming $\rightarrow$ track sharining hits. The two approaches for trimming are shown. The efficiency plots show how much we lose in comparison to CA (Figure 6.2). The third plot shows how many track candidates we get at the end. Both trimming steps give similar efficiency and total tracks at the end.

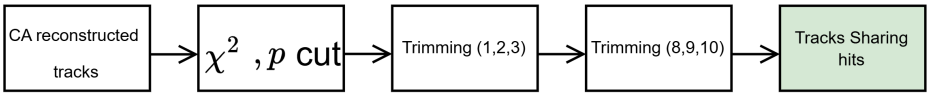


Figure 6.9: Flow diagram of the inside-out approach, with the final box showing the track after all ambiguity resolution steps, color-coded for clarity.

6.3.3 Track Sharing Hits

After applying trimming there may be a possibility that two or more than two tracks share one or more than one hit. Earlier the tracks sharing triplets (Equation 6.1) at the end of the track chain were analyzed by the trimming steps. However, track-sharing hits may still be present after trimming from both ends. For this, the tracks obtained after the trimming step (either from the inside-out or outside-in approach) are converted to their respective hit representation. Then if a hit is common among tracks then the track with minimum χ^2 is selected. The metric obtained from the inside-out and outside-in approach is shown in Figure 6.8. Both efficiencies remain similar to the performance shown in Figure (6.6 and 6.7). The average number of reconstructed tracks after this step does not change significantly.
The three steps of ambiguity resolution are applied to tracks reconstructed by the CA algorithm sequentially. The two approaches inside-out and outside-in are shown in Figure 6.9 and 6.10. Both approaches are almost symmetric and give equal track metrics at the end (Figure 6.8).

6.4 Results

After successfully reducing combinatorics without affecting efficiencies significantly the purity is calculated for both 50% matching and perfect matching. The purity is shown in Figure 6.11, for the final step of ambiguity resolution (Section 6.3.3). The average signal purity and efficiency for both matchings are around $\approx 79\%$ and $\approx 98\%$ respectively.

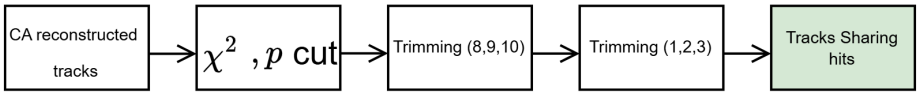


Figure 6.10: Flow diagram of the outside-in approach, with the end box showing the track after all ambiguity resolution, color-coded for clarity.

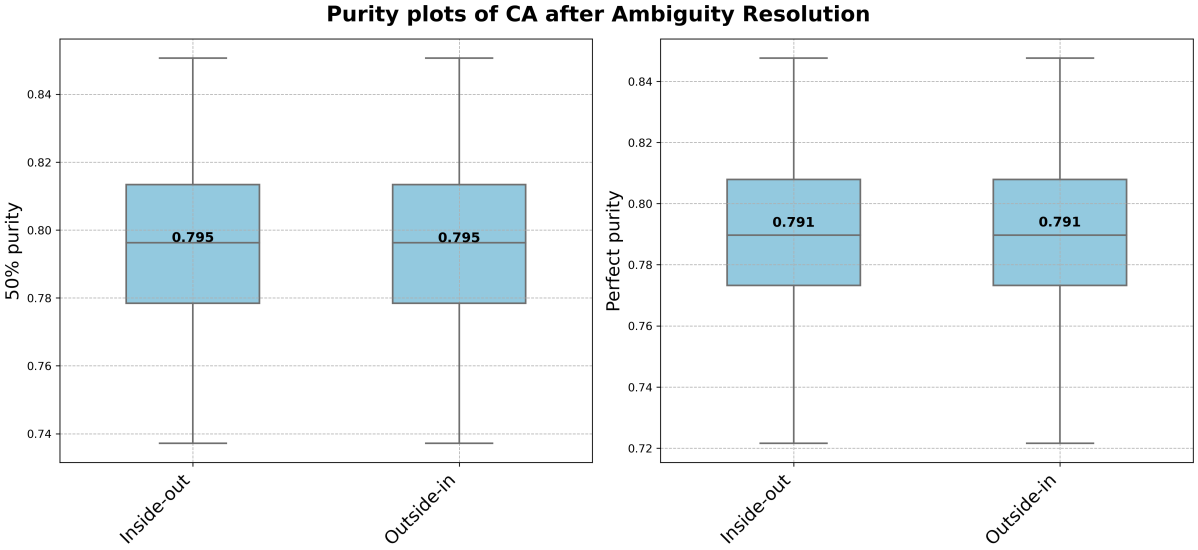


Figure 6.11: Purity plots for both matchings. The plots are shown for the final configurations after applying ambiguity resolution steps sequentially as mentioned in Section 6.3

.

However, it does not mean that the rest of the ≈ 20% particles constructed are **fake tracks**. They may belong to physical tracks as well. They are counted as non-signal because they do not satisfy our **signal track** definition (Section 4.2). This is shown in χ^2 distribution obtained after the inside-out approach [2] in Figure 6.12 that shows **non-signal** tracks have meaningful χ^2. Thus we can not get any benefit from the χ^2 cut for improving signal purity of tracks.

To further improve the purity we can utilize the **fitted transverse momentum** (p_{Tfit}) obtained from the GTTF algorithm. The distribution is shown in Figure 6.13 for the final configuration of tracks obtained from the inside-out approach. The non-signal distribution peaks much higher before $p_T <= 1$GeV which shows that non-signal tracks are essentially the ones that do not satisfy our signal definition.

Thus we pass one more selection cut based on fitted transverse momentum to the track candidates obtained from the final stage of the inside-out (Figure 6.9) approach for every event and then the mean is calculated. For various cut values[3] of p_T the metrics are listed in Table 6.2. The efficiency and purity for $p_T = 1$ GeV gives the maximum value for both matchings (The working point is selected based on F1-score [38]).

[2]Both approaches (Figure 6.9 and 6.10) give similar results thus the analysis in this section is based on inside-out approach.

[3]This cut is different than the one introduced for doublets 5.1.2

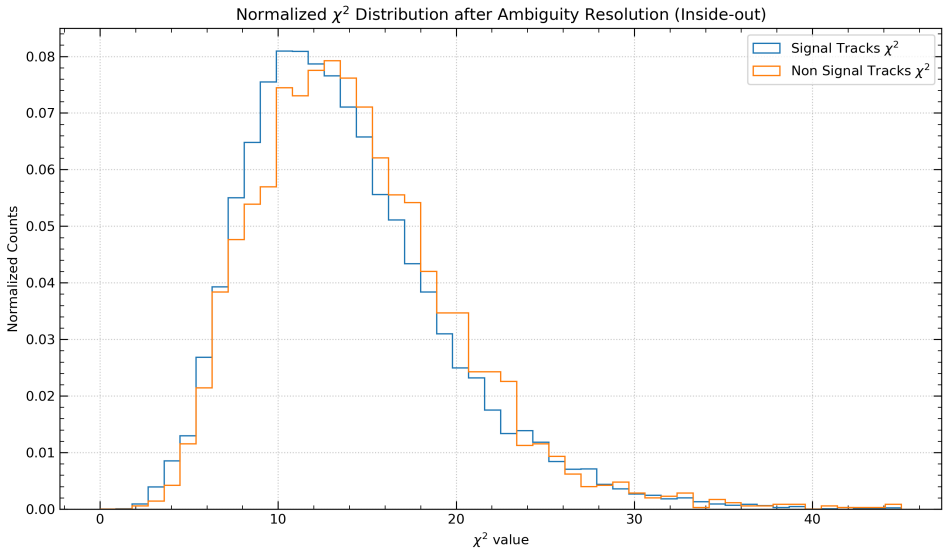


Figure 6.12: Normalized χ^2 distribution (degrees of freedom = 8) for **signal and non-signal** tracks obtained from the final tracks from the inside-out approach (Fig. 6.9).

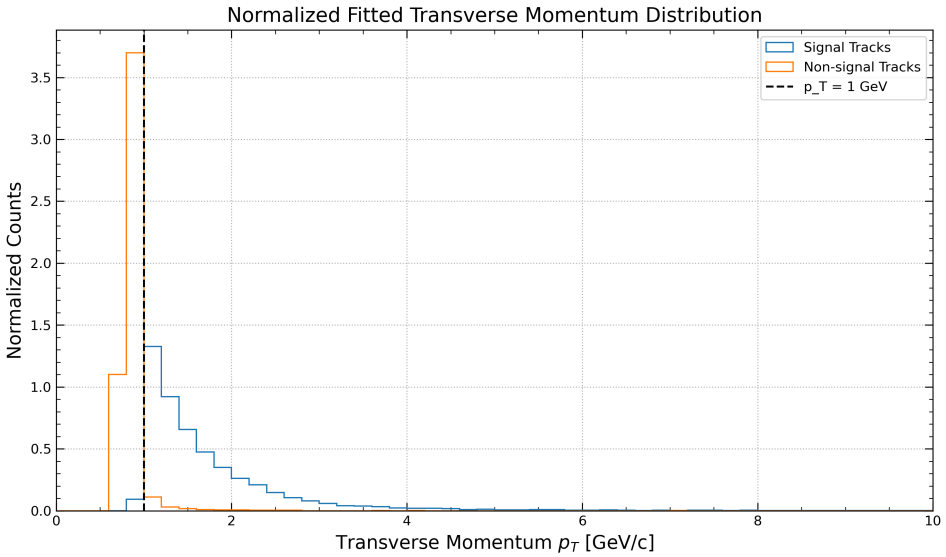


Figure 6.13: Normalized fitted p_T distribution for **signal and non-signal** tracks obtained from the final tracks from the inside-out approach (Fig. 6.9).

p_T cut (GeV/c)	Mean Efficiency		Mean Purity	
	50% Matching	Perfect Matching	50% Matching	Perfect Matching
0.5	0.9811	0.9756	0.7939	0.7894
0.6	0.9811	0.9756	0.7939	0.7894
0.7	0.9811	0.9756	0.7975	0.7930
0.8	0.9811	0.9756	0.8326	0.8280
0.9	0.9811	0.9756	0.8992	0.8941
1.0	0.9627	0.9573	0.9943	0.9887

Table 6.2: Mean Efficiency and Mean Purity per event, for two matching criteria obtained after applying p_T cut to tracks from the final stage of the inside-out approach.

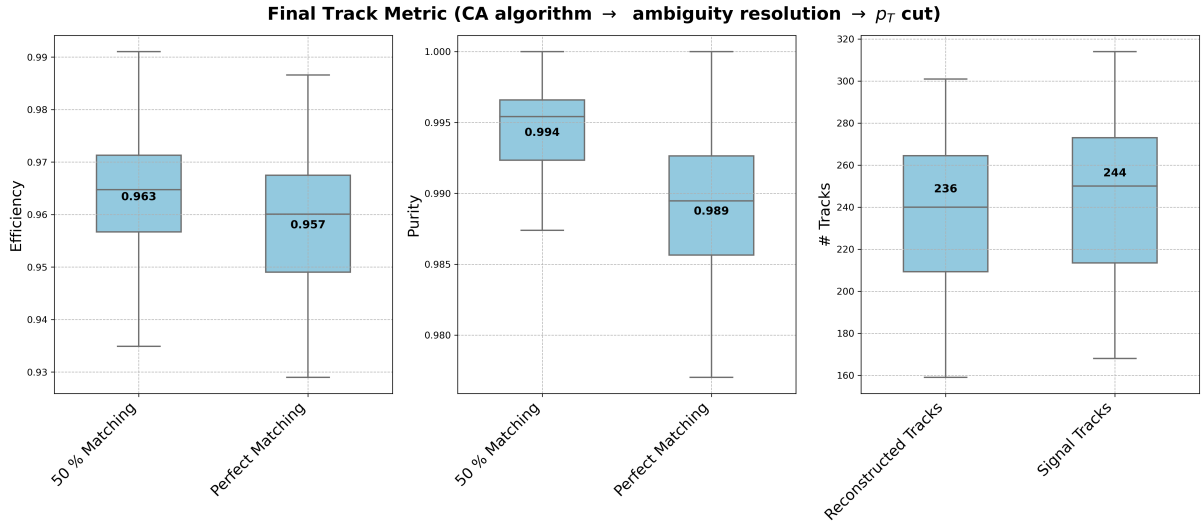


Figure 6.14: Final track performance metric along with their mean obtained after applying steps sequentially - CA Tracks → ambiguity resolution (inside-out) → p_T cut on tracks. The efficiency and purity are shown for two matchings. The rightmost plot describes the number of remaining tracks after applying the steps in comparison to an actual number of signal tracks.

With this, the **final metric of the CA algorithm** combining ambiguity resolution (inside-out) plus additional $p_T = 1$ GeV/c cut on resulted tracks are shown in. The average efficiency and purity per event are $\approx 96\%$ and $\approx 99\%$ for both matchings respectively. The average number of total tracks per event has also been reduced to **236** which was originally **4235** obtained from CA (Figure 6.2). Table 6.3 summarizes the sequential effect of the ambiguity resolution along with p_T cut to the tracks reconstructed by CA.

Stage	Mean Efficiency (%)		Mean Purity (%)		Reconstructed Tracks
	50% Matching	Perfect Matching	50% Matching	Perfect Matching	
CA algorithm	98.5	98.5	-	-	4235
χ^2 and p cut	98.1	98.1	-	-	2339
Trim(1,2,3)	98.1	97.7	-	-	607
Trim(8,9,10)	98.1	97.6	-	-	301
Track Sharing hits	98.1	97.6	79.5	79.1	300
p_T cut = 1 GeV	**96.3**	**95.7**	**99.4**	**98.9**	**236**

Table 6.3: The summary table of average track metrics and the average reconstructed tracks at different stages. All stages are applied sequentially (top to bottom) and the inside-out approach is chosen for ambiguity resolution. The entries in purity are only filled when track candidates do not share any hits among them. The average number of present signal tracks is **244**. The last row is highlighted in bold to indicate the **final track** metric.

This **final tracks** (last row of Table 6.3) can be used for further analysis. The p_T distribution of the reconstructed final tracks and the signal tracks is shown in Figure 6.15 which shows the reasonably good agreement between the reconstructed p_T and signal track p_T obtained from the TrackML dataset.

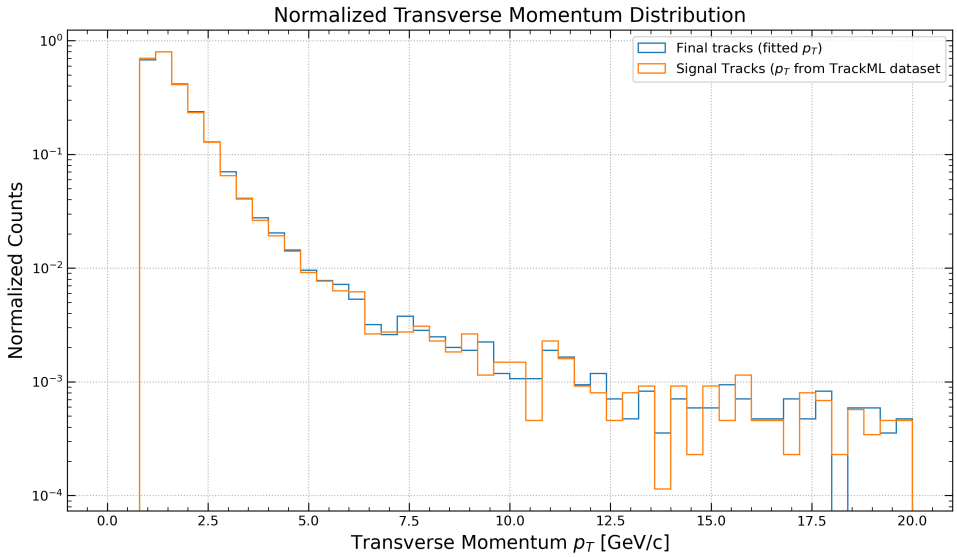


Figure 6.15: The normalized distribution of p_T with y axis is in logarithmic scale. For final tracks, the p_t is reconstructed from the GTTF algorithm. For the signal tracks this is already given in the TrackML dataset.

Chapter 7

Discussion and Outlook

The CA algorithm, fitting methods, and ambiguity resolution techniques give an impressive signal efficiency of $\approx 96\%$ and signal purity of $\approx 99\%$ for perfect matching. This explains the potential of the algorithm for track reconstruction for the HL-LHC phase. This chapter concludes the thesis with a comprehensive discussion of the results with the outlook.

7.1 Discussion

Forming track candidates with CA was inherently simple but the complexity of the algorithm arose in forming triplets with preselection cuts and applying triplet fit to them. The following sections conclude results obtained ion chapter 5 and 6 as follows:

1. **Triplet Formation**: The triplet-building stage along with fitting constitutes the most computationally intensive portion of the algorithm, consuming a substantial amount of the overall processing time. With the four preselection cuts for building doublets and triplets ($n_{z_0} = 4$, $p_{Tcut} = 0.6$, $n_{d\theta} = 3$, $n_{d\kappa} = 6$), the flow table describes how much average metric per event we get -

Segment	**# Signal Segments**	**# Reconstructed Segments**	**Average Efficiency**	**Average Purity**
Doublet	3,255	614,145	99.9%	0.5%
Triplet (before χ^2 cut)	3,526	153,953	99.8%	2.4%
Triplet (after χ^2 cut)	3,526	34,036	99.6%	10.5%

Table 7.1: Metric summary for different segments. The values are obtained by taking the average of 90 events of the TrackML dataset.

   The doublets and triplets were initially formed using loose selection criteria by taking the integer values of the layer-wise cuts, resulting in approximately 600,000 doublets and 154,000 triplets per event. This high segment generation yields good efficiency, capturing all signal segments, with the number of triplets further reduced after applying the χ^2 cut. However, the purity was very low with these cut values. If the algorithm is further targeted for hardware acceleration, this step can be optimized by taking real values for the layer-wise selection cuts ($n_{z_0}, n_{d\theta}, n_{d\kappa}$).

2. **Tracking Performance**: The reconstructed track candidates by CA, were much higher due to the heavy pile-up environment. This step reconstructs signal tracks

with good efficiency but also gives rise to many combinations of tracks. The ambiguity resolution implemented in this thesis was a very robust way of reducing combinatorics. The perfect matching efficiency goes from 98.5 % to 95.7 % (Table 6.3). The performance metric as outlined in Table 6.3, shows the effect of p_T cut to the final track metric. This cut affects both efficiency compared to other stages but also increases purity to 99%. The distribution shown in Figure 6.15 shows the reconstructed tracks and their fitted momentum p_T does agree with the TrackML dataset value.

7.2 Outlook

The implementation of the cellular automata-based track finder in this thesis opens up several avenues for future exploration with this aspect:

1. We devised CA for finding the longest track chain in the barrel region (10 hits), and avoid calling a "track" a signal if it misses a detector hit due to detector inefficiency. The algorithm can also be used for reconstructing track candidates with hits < 10 and accounting for detector inefficiencies. Thus the tracking performance can be calculated concerning all physical tracks.

2. The trimming step of finding a track from a tree was a robust way of constructing tracks which could also benefit by some advanced algorithm for pile up [39].

3. The region of interest in this thesis was only barrel, the CA algorithm could be extended to the other part of the detector and evaluate full tracking performances. When we extend to the other region of the barrel the algorithm will require some additional handling because now the track collection would be complicated having different number of hits. Usually, the longest tracks are found first and then tracks with shorter lengths are built.

4. The CA approach implemented in this thesis was solely on CPUs and further this can benefit from hardware acceleration with GPUs along with fitting algorithms to explore its online tracking capability.

5. The algorithm performance can also be compared to the state-of-the-art combinatorial Kalman filter and studied for ATLAS ITk [40].

Bibliography

[1] L. R. Evans. *The Large Hadron Collider Project*. Technical Report LHC Project Report 53. Geneva: CERN, 1996.

[2] David Rohr et al. "Track Reconstruction in the ALICE TPC using GPUs for LHC Run 3". In: *arXiv preprint arXiv:1811.11481* (2018). arXiv: 1811.11481 [physics.ins-det].

[3] ATLAS Collaboration. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3 (2008), S08003. DOI: 10.1088/1748-0221/3/08/S08003. URL: https://doi.org/10.1088/1748-0221/3/08/S08003.

[4] CMS Collaboration. "The CMS Experiment at the CERN LHC". In: *Journal of Instrumentation* 3 (2008), S08004. DOI: 10.1088/1748-0221/3/08/S08004. URL: https://doi.org/10.1088/1748-0221/3/08/S08004.

[5] LHCb Collaboration. "The LHCb Detector at the LHC". In: *Journal of Instrumentation* 3 (2008), S08005. DOI: 10.1088/1748-0221/3/08/S08005.

[6] Béjar Alonso, I. and Brüning, O. and Fessia, P. and Lamont, M. and Rossi, L. and Tavian, L. and Zerlauth, M. "High-Luminosity Large Hadron Collider (HL-LHC): Technical design report". In: CERN Yellow Reports: Monographs 10 (2020). DOI: 10.23731/CYRM-2020-0010. URL: https://cds.cern.ch/record/2749422.

[7] CERN. *High-Luminosity LHC*. https://hilumilhc.web.cern.ch/. Accessed: 2024-08-23. 2024.

[8] D. Contardo. *Highlights of LHC, ATLAS, CMS and LHCb upgrades*. Séminaire thématique GT01: « Physique des particules », 12-13 Mars 2020, IP2I Lyon. 2020.

[9] CERN. *Run 3: an opportunity to expand the LHC physics programme*. https://www.home.cern/press/2022/run-3. Accessed: August 29, 2024. 2022.

[10] G. Arduini et al. "LHC Upgrades in preparation of Run 3". In: *Journal of Instrumentation* 19.05 (May 2024), P05061. DOI: 10.1088/1748-0221/19/05/P05061. URL: https://dx.doi.org/10.1088/1748-0221/19/05/P05061.

[11] Rudolf Frühwirth et al. *Pattern Recognition, Tracking and Vertex Reconstruction in Particle Detectors*. Particle Acceleration and Detection. Springer, 2013. ISBN: 978-3-642-35099-5.

[12] Moritz Kiehn et al. "The TrackML high-energy physics tracking challenge on Kaggle". In: *EPJ Web Conf.* 214 (2019), p. 06037. DOI: 10.1051/epjconf/201921406037. URL: https://cds.cern.ch/record/2699475.

[13] TrackML. *TrackML Particle Tracking Challenge*. https://www.kaggle.com/c/trackml-particle-identification. Accessed: 2024-10-22. 2018.

[14] R Mankel. "Pattern recognition and event reconstruction in particle physics experiments". In: *Reports on Progress in Physics* 67.4 (Mar. 2004), pp. 553–622. ISSN: 1361-6633. DOI: 10.1088/0034-4885/67/4/r03. URL: http://dx.doi.org/10.1088/0034-4885/67/4/R03.

[15] Wikipedia contributors. *Graphics processing unit*. Accessed: 2024-10-22. 2024. URL: https://en.wikipedia.org/wiki/Graphics_processing_unit.

[16] Javier Duarte and Jean-Roch Vlimant. "Graph Neural Networks for Particle Tracking and Reconstruction". In: *Artificial Intelligence for High Energy Physics*. WORLD SCIENTIFIC, Feb. 2022, pp. 387–436. ISBN: 9789811234033. DOI: 10.1142/9789811234033_0012. URL: http://dx.doi.org/10.1142/9789811234033_0012.

[17] Stephen Wolfram. *Notes on Cellular Automata and Complexity*. Accessed: 2024-08-26. 2023. URL: https://www.wolframscience.com/reference/notes/876b/.

[18] Martin Gardner. "Mathematical Games – The fantastic combinations of John Conway's new solitaire game "life"". In: *Scientific American* 223 (Oct. 1970), pp. 120–123.

[19] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays*. Natick, MA, 2001.

[20] Michael J. Gibson, Edward C. Keedwell, and Dragan A. Savić. "An investigation of the efficient implementation of cellular automata on multi-core CPU and GPU hardware". In: *Journal of Parallel and Distributed Computing* 75 (2015), pp. 1–15. DOI: 10.1016/j.jpdc.2014.10.011.

[21] B. Marchetti et al. "ARES: Accelerator Research Experiment at SINBAD". In: *Proceedings of IPAC15*. JACoW. Richmond, VA, 2015, TUPWA029.

[22] D Funke et al. "Parallel track reconstruction in CMS using the cellular automaton approach". In: *Journal of Physics: Conference Series* 513.5 (2014), p. 052010. DOI: 10.1088/1742-6596/513/5/052010.

[23] I. Abt et al. "CATS: a cellular automaton for tracking in silicon for the HERA-B vertex detector". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 489.1-3 (Aug. 2002), pp. 389–405. DOI: 10.1016/S0168-9002(02)00790-8.

[24] Valentina Akishina. "Four-dimensional event reconstruction in the CBM experiment". PhD thesis. Frankfurt am Main, Germany: Goethe University Frankfurt, 2019.

[25] Andre Schöning. *A General Track Fit based on Triplets*. 2024. arXiv: 2406.05240 [physics.ins-det]. URL: https://arxiv.org/abs/2406.05240.

[26] V. L. Highland. "Some Practical Remarks on Multiple Scattering". In: *Nucl. Instrum. Methods* 129 (1975), p. 497. DOI: 10.1016/0029-554X(75)90743-0.

[27] Kaggle. *Kaggle: Your Home for Data Science*. https://www.kaggle.com. Accessed: 2024-10-21. 2024.

[28] Sebastien Roy-Garand. "ATLAS Inner TracKer Upgrade". In: *11th Edition of the Large Hadron Collider Physics Conference*. On behalf of the ATLAS ITk collaboration. ATLAS ITk collaboration. Belgrade, Serbia, May 2023. URL: https://cds.cern.ch/record/2869708/files/ATL-ITK-PROC-2023-011.pdf.

[29] Simone Paoletti. *The CMS Tracker Upgrade for the High Luminosity LHC*. Tech. rep. Geneva: CERN, 2020. DOI: 10.22323/1.364.0138. URL: https://cds.cern.ch/record/2723307.

[30] T. Sjöstrand, S. Mrenna, and P. Skands. "A brief introduction to PYTHIA 8.1". In: *Computer Physics Communications* 178 (2008), pp. 852–867. DOI: 10.1016/j.cpc.2008.01.036.

[31] Xiaocong Ai et al. "A Common Tracking Software Project". In: *Computing and Software for Big Science* 6.1 (Apr. 2022). ISSN: 2510-2044. DOI: 10.1007/s41781-021-00078-8. URL: http://dx.doi.org/10.1007/s41781-021-00078-8.

[32] Python Software Foundation. *Python*. Accessed: YYYY-MM-DD. 2023. URL: https://www.python.org/.

[33] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[34] Wes McKinney. *Data Analysis in Python*. ISBN: 978-1-449-34889-0. O'Reilly Media, 2010.

[35] J. D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[36] Sabrina Amrouche et al. "The Tracking Machine Learning Challenge: Throughput Phase". In: *Computing and Software for Big Science* 7.1 (2023), p. 1. DOI: 10.1007/s41781-023-00094-w. URL: https://doi.org/10.1007/s41781-023-00094-w.

[37] Tamasi Rameshchandra Kar. "A Triplet Track Trigger for Future High Rate Collider Experiments". PhD thesis. Heidelberg University, 2020. DOI: 10.11588/heidok.00029043. URL: https://archiv.ub.uni-heidelberg.de/volltextserver/29043/.

[38] Roy Van and Patrick Raes. "The F1 score and its application to classification problems". In: *Journal of Machine Learning Research* 16 (2015), pp. 103–117. URL: http://www.jmlr.org/papers/volume16/van15a/van15a.pdf.

[39] Ferenc Siklér. *Dense tracking: for efficient track reconstruction in high multiplicity events*. Talk presented at Connecting The Dots / Intelligent Trackers 2017. Accessed: [Insert access date]. Mar. 2017. URL: https://indico.cern.ch/event/577003/contributions/2415235/attachments/1424172/2183976/sikler_denseTracking_ctdwit17.pdf.

[40] ATLAS Collaboration. "Technical Design Report for the ATLAS Inner Tracker Strip Detector". In: *CERN-LHCC-2017-005* (2017). arXiv: 1707.04669 [physics.ins-det].

Acknowledgments

I would like to express my heartfelt gratitude to my supervisors, Prof. Dr. André Schöning and Dr. Sebastian Dittmeier of the Physikalisches Institute, Heidelberg, for offering me the opportunity to work on this exciting project on Cellular Automata. Their guidance and insightful discussions throughout the project were invaluable, and I thoroughly enjoyed exploring, creating, and learning within such an intellectually stimulating environment. I would also like to thank our group secretary, Claudia Wallenwein, for her assistance with complex travel arrangements, which eased the logistics of my work. The weekly group meetings were instrumental to the progress of this thesis, enhancing my understanding of the algorithms involved. I am sincerely grateful to all the members of the Mu3e and ATLAS groups for their support and camaraderie. Additionally, I would like to thank my past supervisors from DESY, David, Yee, and Dr. Federico Meloni, whose encouragement and support during my summer project provided a strong foundation for this work.

Coming from the foothills of the Himalayas, I have always been drawn to the concept of "SANGATKAAR" in *pahadi* music—a term that embodies the essence of musicians supporting one another in harmony. I feel incredibly blessed to have so many *sangatkaar* in my life, without whom this MSc journey would not have been possible. My deepest thanks go to my siblings and cousins: *Saurabh, Neeru, Laxmi, Hema, Deepak, Shivam, Deepika, Ankur, and Kavita.* My sister, Neelam, played a particularly vital role, throughout this time.

Alongside my family, I am grateful for an amazing set of friends who have stood by me unconditionally. My high school friend, *Rishabh Rathor*, and my BSc friends, *Anuj* and *Sukhveen*, have been pillars of support, offering invaluable guidance and companionship. During this journey, our online musical nights and philosophical discussions about life were a breath of fresh air.

In Heidelberg, I was fortunate to meet friends like *Shreya*, *David*, and *Kanika*, whose kindness and encouragement helped me through challenging times. The constant support and encouragement from *Shreya* and *Kanika* were invaluable throughout this journey.

As a musician, I find that research is quite similar to music where unknown melodies can be created from known notes. The way music encapsulates the breadth of human emotion is a profound inspiration, and I am deeply grateful to the artists whose creations helped me stay focused and connected throughout this thesis. I sincerely thank *Nusrat Fateh Ali Khan, Rahat Fateh Ali Khan, Pt. Hariprasad Chaurasia, Amjad Ali Khan, Pt. Shiv Kumar Sharma, Zakir Hussain, Rakesh Chaurasia, Anoushka Shankar, Coke Studio Pakistan, Hans Zimmer, Satinder Sartaaj, Ludovico Einaudi, and Hadiqa Kiani.*

In closing, I would like to share a final lesson that guided me through this journey:

"Take care of small things, and the big things will take care of themselves."