University College London

Department of Physics, HEP

CDT-DIS

Novel Methodologies for Pattern Recognition of Charged Particle Trajectories in the ATLAS Detector

Author:
Charlie Pitman Donaldson

Supervisors:
Nikos Konstantinidis &
Dmitry Emeliyanov

July 26, 2022

Declaration

I, Charlie Pitman Donaldson, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

By 2029, the Large Hadron Collider will enter its High Luminosity phase (HL-LHC) in order to achieve an unprecedented capacity for discovery. As this phase is entered, it is essential for many physics analyses that the efficiency of the reconstruction of charged particle trajectories in the ATLAS detector is maintained. With levels of pile-up expected to reach $<\mu> = 200$, the number of track candidates that must be processed will increase exponentially in the current pattern matching regime. In this thesis, a novel method for charged particle pattern recognition is developed based on the popular computer vision technique known as the Hough Transform (HT). Our method differs from previous attempts to use the HT for tracking in its data-driven choice of track parameterisation using Principal Component Analysis (PCA), and the division of the detector space in to very narrow tunnels known as sectors. This results in well-separated Hough images across the layers of the detector and relatively little noise from pile-up. Additionally, we show that the memory requirements for a pattern-based track finding algorithm can be reduced by approximately a factor of 5 through a two-stage compression process, without sacrificing any significant track finding efficiency. The new tracking algorithm is compared with an existing pattern matching algorithm, which consists of matching detector hits to a collection of pre-defined patterns of hits generated from simulated muon tracks. The performance of our algorithm is shown to achieve similar track finding efficiency while reducing the number of track candidates per event.

Impact Statement

To continue building on our understanding of fundamental physics, it is crucial that we exploit the huge quantities of data from the LHC to make both precision measurements of the Standard Model and search for Beyond Standard Model processes. Therefore, the ATLAS detector requires frequent and significant upgrades to both its hardware and its software to make these discoveries possible. The work in this thesis directly contributes to the ongoing efforts to develop faster and more efficient algorithms to reconstruct event data to be used in many different physics analyses.

Outside of high energy physics and, indeed, outside of academia, techniques such as the Hough Transform are employed for a wide range of imaging tasks. The novel concept of applying Principal Component Analysis to parameterise objects for the Hough Transform could be leveraged to improve detection capabilities in such tasks, and contribute to the field of computer vision overall.

Acknowledgements

First and foremost, I would like express my deepest gratitude to Dmitry Emeliyanov and Nikos Konstantinidis, who have provided so much guidance and support, technical or otherwise, and have made the whole achievement possible. If not for the constant assistance and understanding from you both, I'd be thanking no one at all.

To my peers in the CDT, I am grateful to have met you all and the experience would not have been the same without you. I would like to thank, in particular, Lucas Borgna for being my husband and for helping me through the worst months the past few years have had to offer.

To Ana, Ma, Phoebe and George, I cannot thank you enough for the endless love and support you have all provided.

Contents

Acronyms

ALICE A Large Ion Collider Experiment.

ATLAS A Toroidal LHC AppratuS.

ATMP Associative Memory Tracking Processors.

BSM Beyond Standard Model.

CERN European Council for Nuclear Research.

CMS Compact Muon Solenoid.

DC Don't Care.

DF Drift Tube.

DoG Difference of Gaussian.

DoH Determinant of Hessian.

EF Event Filter.

EM Electromagnetic.

FTK Fast TracKer.

GHT Generalised Hough Transform.

HL-LHC High Luminosity Large Hadron Collider.

HLT High Level Trigger.

HT Hough Transform.

HTT Hardware Tracking for the Trigger.

HTTIF Hardware Tracking for the Trigger Interface Units.

IBL Insertable B-Layer.

ID Inner Detector.

ITk Inner Tracker.

LEP Large Electron-Positron Collider.

LHC Large Hadron Collider.

LoG Lapacian of Gaussian.

MC Monte Carlo.

PCA Principal Component Analysis.

QCD Quantum ChromoDynamics.

RoI Regions of Interest.

SCT Semiconductor Tracker.

SM Standard Model.

SS Superstrip.

SSID Superstrip Identifier.

SSTP Second-Stage Tracking Processors.

TDAQ Trigger and Data Acquisition.

TDR Technical Design Report.

TFM Track Fitting Mezzanine.

TRT Transition Radiation Tracker.

Chapter 1

Introduction

Many of the experiments at the Large Hadron Collider (LHC) face a common problem - in order to continue making new discoveries and further precision measurements, the luminosity must continue to reach unprecedented magnitudes. While an increased luminosity allows for a greater capacity to detect statistically significant levels of very rare processes, it is accompanied by an even greater rise in the number of pile-up collisions. In the High Luminosity LHC (HL-LHC) phase starting in 2029, the average number of pile-up collisions will reach $<\mu> = 200$, which poses a serious data processing challenge, especially for track reconstruction in the trigger system. A proposed solution in the ATLAS experiment has been to implement Hardware-based Tracking for the Trigger (HTT) through the storage of pattern banks obtained from Monte Carlo simulations and a pattern matching algorithm. Patterns in these banks consist of eight layers of hits (1 pixel layer, 7 strip layers) generated through MC simulation of single muons traversing the detector. The hits in each layer have variable width allowing groups of very similar tracks to be described by a single pattern. The pattern matching algorithm then attempts to match the hits fired in an event to the template patterns in the bank. If there are multiple hits in each layer, there will be several combinations of clusters for which a χ^2 fit can be computed and a cut applied. The top quality tracks

can then be extrapolated into the remaining layers before a full χ^2 fit is performed, and the track parameters can then be calculated. Although development of the hardware-based solution has stopped at the time of writing, we propose a novel approach that uses the pattern bank data structures for track finding.

The approach presented in this thesis consists of two main steps. The first step involves the compression and dimensionality reduction of the HTT pattern banks using Principal Component Analysis (PCA). The number of components necessary to explain the variance in the patterns and produce a comparable pattern matching efficiency will be investigated. The second step of our proposed method employs a well-known feature extraction technique from the field of computer vision known as the Hough Transform (HT). The HT has been used for decades to detect objects in images and is not new to high energy physics experiments. There are several examples, including from both ATLAS and CMS, of attempts to use the HT for charged particle track reconstruction. However, these attempts fail to systematically justify the choice of track parameterisation which is essential for effective track finding with the HT. Instead, they appear more determined to use the HT with some ad-hoc parameterisation and then assess its suitability afterwards. In this thesis, we will use the HTT pattern banks as training data for a PCA-based compression and hence develop a data-driven approach to using the HT. In addition to this, the HT tracking algorithm is applied independently to narrow tunnels of the detector volume to reduce the pile-up noise in the resulting images.

The thesis will begin with an overview of the ATLAS detector and the Inner Tracker responsible for charged particle tracking, as well as the design of the HTT and simulated pattern banks. Next, a literature review of the HT will be conducted, covering its initial invention and its multiple use cases over the years. After this, we will describe the two-stage compression of the pattern banks and demonstrate that the compressed, low-dimensionality pattern space is both

appropriate and effective for performing the HT. A detailed explanation will then be provided for how a tracking algorithm can be designed in this space, including a description of the additional image processing techniques that are used. Finally, the implementation and performance of our HT-based solution will be presented and discussed, contrasting the results with those obtained by the HTT pattern matching algorithm.

Chapter 2

The LHC, ATLAS and the HTT

To put the research in this thesis in context, this chapter will first describe the LHC machinery and roadmap. This will follow on to the problems that the ATLAS experiments will face due to the HL-LHC phase and we will describe the detector layout, particularly focusing on the Inner Tracker. Finally, we will outline the plans for the HTT architecture and implementation before describing the pattern bank data structures taken from these plans to be used in this work.

2.1 The LHC

The LHC, which began its operations in 2009, is a two-ring-superconducting hadron collider located at the CERN site on the Swiss-French border. With many of the tunnels and infrastructure already built, the LHC was constructed in the 27km underground ring used previously for the LEP (Large Electron-Positron) accelerator [1], and exploits the existing accelerator chain to inject the main ring with high energy hadrons. The broader accelerator complex is shown in Figure 2.1.

There are several different experiments located along the main ring, some of which are dedicated to studying specific sectors of the Standard Model (SM), such as the LHCb [3] experiment designed to study b-physics, or ALICE [4], which

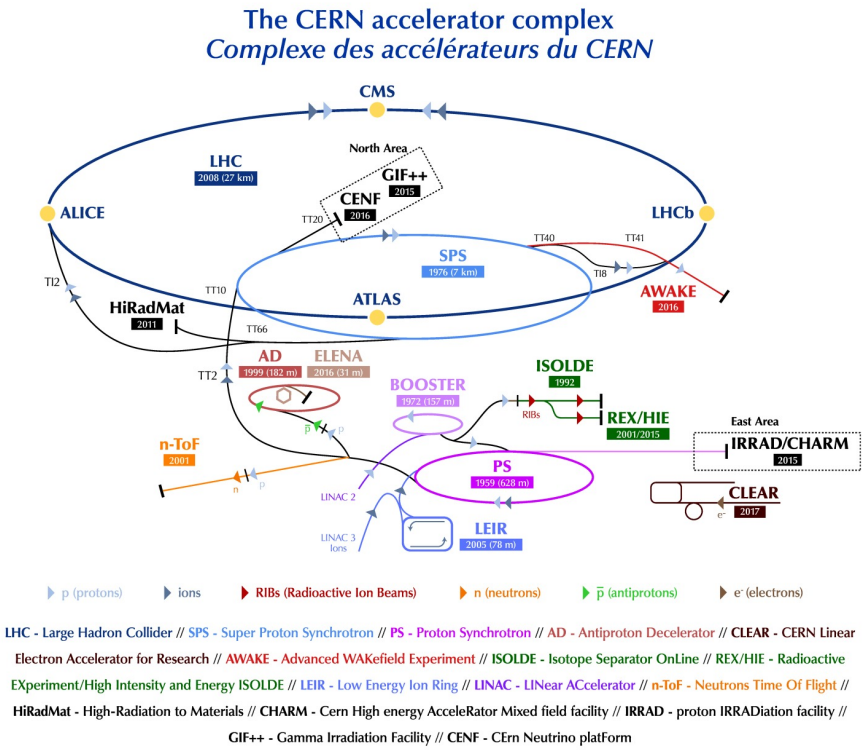


Figure 2.1: **A schematic map of the CERN accelerator complex, showing the different experiments and their relative locations in the site. [2].**

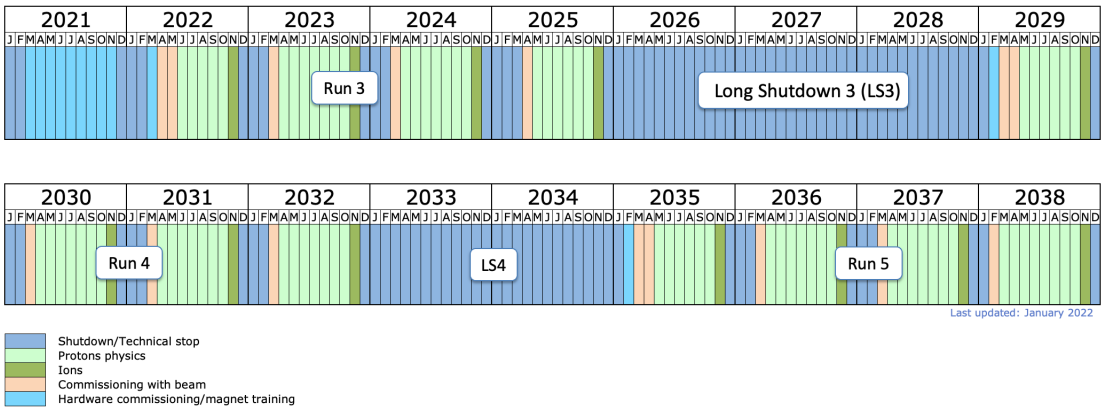


Figure 2.2: **The long term LHC schedule. In January 2022, the schedule was updated with long shutdown 3 (LS3) to start in 2026 and to last for 3 years. [7]**

investigates QCD and the strong interaction, in the extreme conditions of heavy ion collisions. Additionally, there are two general-purpose detectors which are designed to study a wide range of physics phenomena. These are the ATLAS experiment (A Toroidal LHC ApparatuS) and the CMS experiment (Compact Muon Solenoid) [5]. To maintain and upgrade the collider and the detectors, the LHC operates only during scheduled Runs and the upgrades take place during years-long shutdowns. In Run 1, the collider achieved centre-of-mass energies of $\sqrt{s} = 7$ TeV with Run 2 increasing to $\sqrt{s} = 13$ TeV. This is accomplished using helium-cooled superconducting magnets producing a magnetic field with strength up to 8T [6]. At the time of writing, the Run 3 phase has begun and will achieve a maximum centre-of-mass energy of $\sqrt{s} = 14$ TeV.

The LHC roadmap for the next 16 years is shown in Figure 2.2. By 2029, the HL-LHC phase is expected to begin, during which the instantaneous luminosity will be 5-7 times greater than the value the LHC was originally designed for. The change in integrated luminosity over time, which represents the accumulation of collected data and hence statistical power in analyses, is shown in Figure 2.3. The ultimate integrated luminosity by the end of the HL-LHC is around 4000 fb^{-1} and would deliver an unprecedented potential for new physics discoveries and incredi-

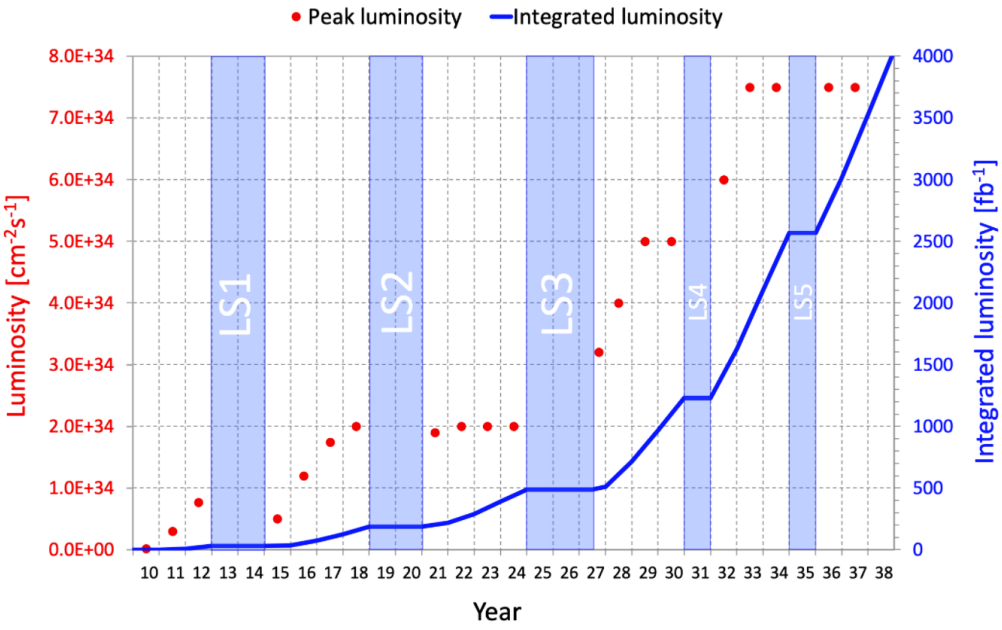


Figure 2.3: **Expected luminosity of the LHC / HL-LHC over the next two decades, showing the ultimate HL-LHC luminosity of 4000fb^{-1}. Note that the timeline in this plot is not in line with that from Figure 2.2, but the luminosity targets are roughly the same. [7].**

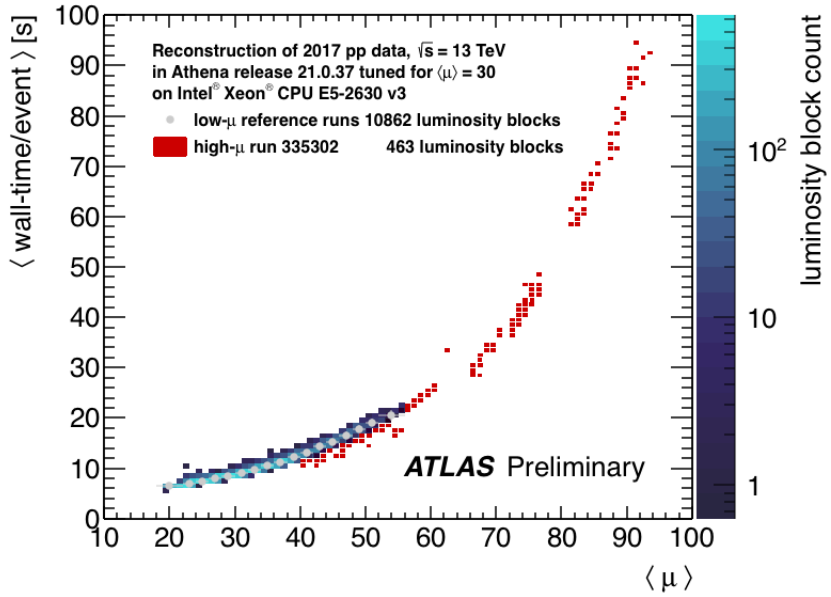


Figure 2.4: **The dependency of reconstruction wall time per event on the pileup. The luminosity block count represents short intervals of data taking, in which the instantaneous luminosity is estimated and, from this, the integrated luminosity derived. [8]**

bly high-precision SM measurements. This will be discussed further in Section 2.4. However, the promising plan for the future is accompanied by many technical challenges that each of the experiments face. The increased luminosity results in many more pp collisions per bunch crossing, referred to collectively as pile-up, which results in drastically higher detector occupancy and radiation levels. In the case of ATLAS, Figure 2.4 shows how the reconstruction time per event increases with pile-up. The increase in time is exponential, and the expected pile-up levels during the HL-LHC are around $<\mu> = 200$, demonstrating the need for upgrades to both the detector and algorithmic designs in ATLAS.

2.2 The ATLAS Experiment

2.2.1 ATLAS Detector

The ATLAS experiment is making SM precision measurements and testing Beyond Standard Model (BSM) theories using the unprecedented centre of mass energy of $\sqrt{s} = 14$ TeV at the LHC. A diagram of the detector is shown in Figure 2.5. In total, the detector is a 44m long cylinder with a diameter of 25m and weighs over 7000 tonnes. It consists of a central barrel and two end-caps to ensure forward physics coverage and hermeticity. The detector's geometry is forward-backward symmetric and, from the inside out, the detector contains the Inner Detector (ID), a name given to several tracking systems designed to measure the momentum of charged particles. This is inside a 2T magnetic field, and beyond this are the calorimeters to measure electromagnetic (EM) radiation and hadrons. Finally, the outer layers of the detector consist of a muon detection system. The coordinate

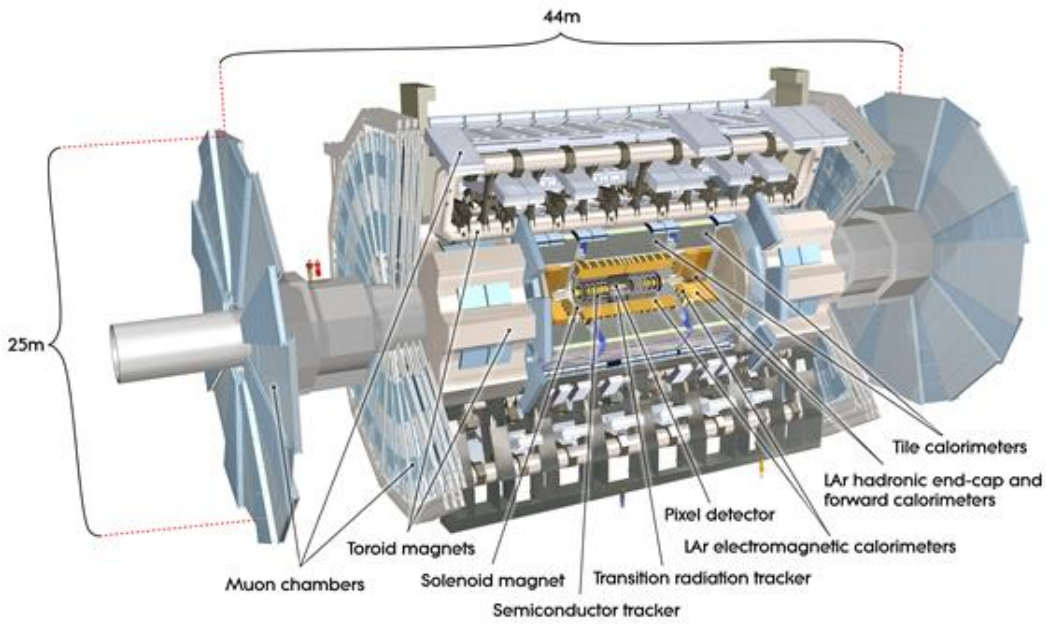


Figure 2.5: **A computer generated image of the ATLAS detector with its main components labelled. [9]**

system used throughout the detector begins with the z-axis which is defined by the beam direction, with the x-y plane being transverse to the beam. The azimuthal angle ϕ is defined in the x-y plane around the beam axis and the polar angle θ is measured from the z-axis [10]. However, a parameter that is often used to express the angle from the beam axis instead is the pseudorapidity, given by:

$$\eta = -\ln \tan\left(\frac{\theta}{2}\right) \tag{2.1}$$

as, at relativistic limits, this is equivalent to the property of rapidity, differences of which are invariant under a Lorentz boost in the z-axis. Additionally, the transverse plane is often used to describe the kinematics of collisions and the transverse momentum is defined as:

$$p_T = \sqrt{(p_x^2 + p_y^2)} \tag{2.2}$$

More comprehensive details of the calorimeters and muon detector can be found in the Technical Design Report (TDR) for the ATLAS detector [10], but since the work in this thesis pertains to tracking, we will turn our attention to the Inner Detector which houses the tracking systems of the ATLAS detector.

2.2.2 The Inner Detector

The innermost layers of the ATLAS detector make up what is known as the Inner Detector (ID). The ID comprises three main parts - the pixel detector and the semiconductor tracker (SCT) and the transition radiation tracker (TRT). A cross-section diagram of the ID is shown in Figure 2.6. The ID offers high efficiency charged particle tracking over the eta range $|\eta| < 2.5$. The components of the ID each contain specialised hardware and contribute towards a full track reconstruction. The innermost pixel detector [11] provides high-precision measurements as

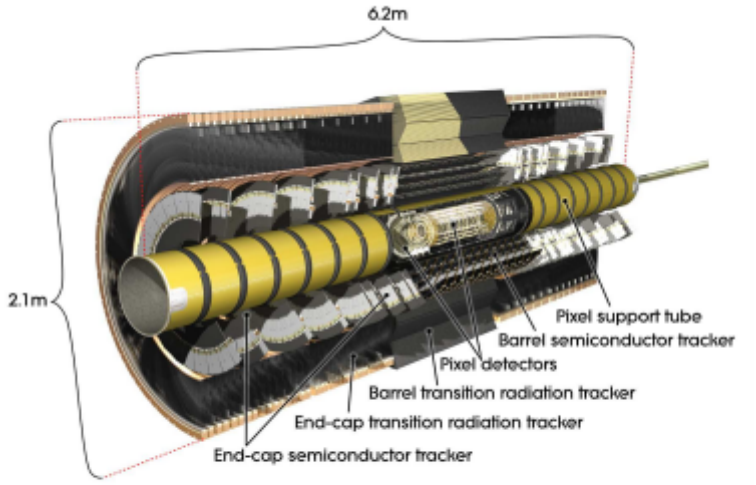


Figure 1. The ATLAS Inner Detector.

Figure 2.6: **Cross-section diagram of the current ATLAS inner detector [11].**

close to the collision point as possible. It consists of 4 barrel layers (including the insertable B-layer (IBL) [12]) as well as three end-cap layers on each side, each providing two-dimensional space-point measurements. The pixel detector was initially constructed with 80 million readout channels, with the IBL providing an additional 12 million [13]. Outside of the pixel detector, the SCT measures charged particles at an intermediate distance from the collision point and improves the determination of vertex position and track momentum. The SCT consists of four barrel layers and nine end-cap layers on each side. In total, these contribute 6 million readout channels to the ID. The outermost section of the ID, the TRT, is used for the identification of charged particles and consists of drift tubes that are filled with a mixture of Xe, CO_2 and O_2, and contain a central gold-plated tungsten wire. When charged particles traverse the TRT, the gas inside the straws is ionised and the free electrons drift towards the wire and are amplified and then read out. In addition, transition radiation provides information on the particle type that passed through the tracker.

The current ID has various limitations that hinder its performance as the LHC machine is upgraded. Radiation damage and high detector occupancy result in the requirement for a full replacement of the ID in the Phase-II upgrade with the new ITk [8] [14]. One significant change in the detector layout is that the ITk will consist only of silicon detectors, replacing the TRT, and extend to a 1m radius, whereas the current SCT outer layer extends only to 60cm. A schematic cross section of the ITk is shown in Figure 2.7. The acceptance of the detector is increased so that the strip detector covers a range of $|\eta| < 2.7$ with the pixel detector extending the range to $|\eta| < 4.0$.

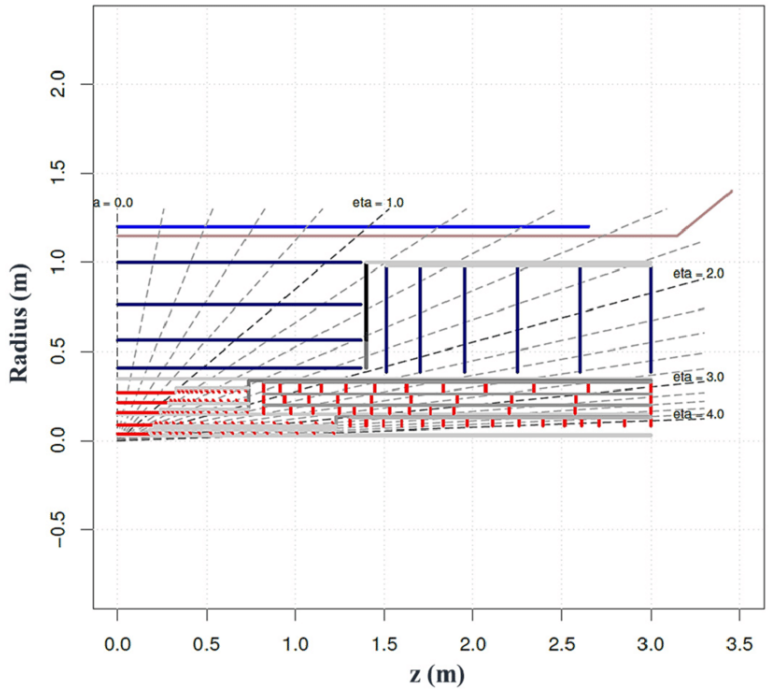


Figure 2.7: **Schematic cross-section of the ATLAS ITk inclined tracker, demonstrating the increased η coverage. The four blue outermost layers compose the strip detector and the five red innermost layers compose the pixel detector. [15]**

2.2.3 Trigger and Data Acquisition (TDAQ)

The collision rate in the ATLAS detector stands at 40 MHz (40 million collisions per second) and the data selected by the trigger must be stored and analysed. The

ATLAS trigger system is responsible for making the decision of which events to keep based on their potential for physics analyses, and may be considered in two parts: the Level 1 (L1) trigger and the High Level Trigger (HLT). In conjunction with the trigger system, the data acquisition system is responsible for channeling the selected data to storage. A schematic for the Run 2 TDAQ system is shown in Figure 2.8. The L1 trigger reduces the event rate from 40MHz to only 100kHz by looking for interesting activity in Regions of Interest (RoIs) in the detector [16]. Using coarse-granularity information from the EM and hadronic calorimeters, the trigger will search for physics objects such as electrons, photons and jets, as well as finding muons in the muon detector. In Run 2, a topological trigger processor (L1Topo) can additionally combine these objects to make topological selections based on, for example, invariant mass requirements or angular separation.

The RoIs identified in the L1 trigger are passed to the HLT for more so-

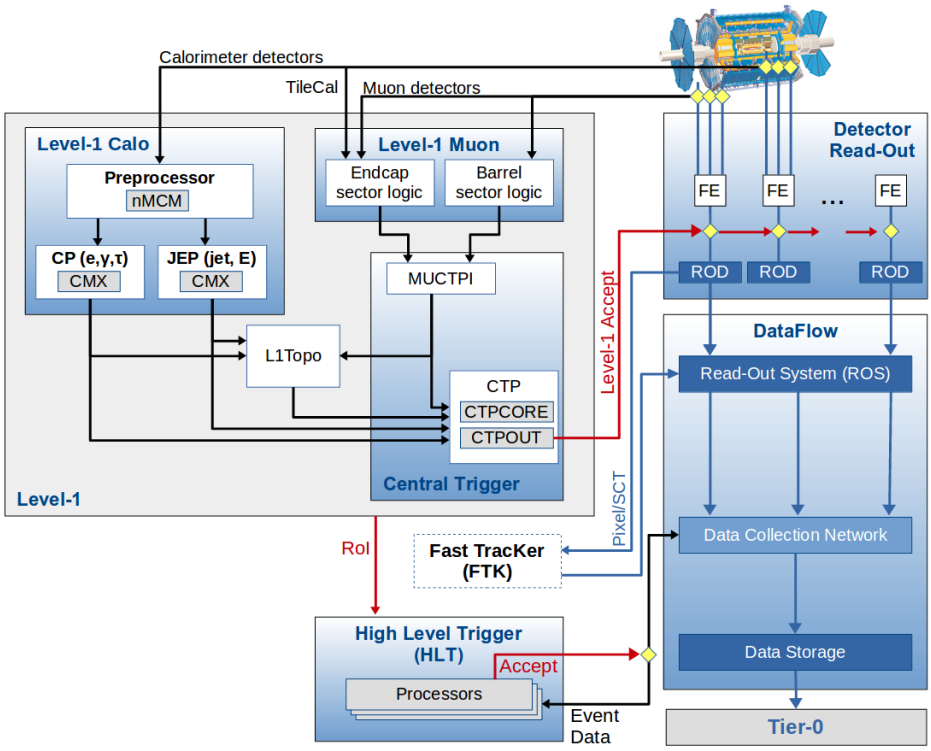


Figure 2.8: **ATLAS Trigger and Data Acquisition (TDAQ) system in Run 2 [16]. Although, note that FtK was not implemented in Run 2.**

phisticated reconstruction using full-granularity information from the rest of the detector. Here, the event rate is reduced to approximately 1kHz with an average processing time of 200ms. Certain tasks, such as the reconstruction of b-jets, would be very CPU-intensive for the HLT. ATLAS pursued a custom-hardware based solution, called the Fast TracKer (FTK), which would provide full-event tracking at 100kHz and would have been replaced by the Hardware Tracking for the Trigger (HTT) for HL-LHC. Both the FTK and HTT rely on the aforementioned pattern matching algorithm, which uses large banks of template patterns to reconstruct tracks. While the ATLAS strategy has since evolved and does not include the FTK and HTT solutions, the algorithmic work developed as part of these projects may still provide the basis for interesting and novel algorithmic solutions, one of which is pursued in this thesis.

2.3 Track Reconstruction in ATLAS

The reconstructed tracks of charged particles are essential for many physics analyses, particularly because of the importance of b-jets. Many high-mass particles, such as the Higgs boson, top quark or predicted BSM particles will decay to b-quarks, so being able to tag b-jets with high efficiency is of great importance for the entire physics programme of ATLAS. The ability to tag jets as b-jets relies heavily on the reconstructed vertices and tracks, so as the luminosity of the LHC increases, it is essential that our ability to quickly reconstruct tracks remains. The general procedure of track reconstruction in the pixel and SCT detectors first involves a clusterisation step which constructs space-points in the detector from energy deposits left by a particle. Then one of several CPU-expensive pattern recognition procedures is implemented. This step generates track candidates by grouping measurements together throughout the inner tracker, and, in addition to finding good track candidates, will produce "fake" track candidates (also sometimes called "ghost" tracks in the literature) not belonging to any real particle. Finally, to gain a more precise estimate of the track parameters and to remove the fake tracks, the quality of the tracks is assessed through a track fitting stage.

2.3.1 Pattern Recognition

The raw measurements within the pixel and SCT sensors are energy deposits above a certain threshold. The process of clusterisation [17] consists of grouping together individual deposits sharing a common edge or corner. From the resulting clusters, three-dimensional measurements of the particle called space-points are made. Once the clusterisation of energy deposits has yielded a set of space points throughout the detector volume, the next stage of pattern recognition is performed. In general, this is equivalent to data association - grouping together hits in different

layers into tracks. This is a non-trivial task, especially when the hit density within the detector is very high, and as such, there are different groups of methods through which this may be done. Broadly speaking, these can be distinguished as *local* and *global* pattern recognition, based on whether the hits are considered sequentially or all at once. The previously described pattern matching algorithm is an example of of global pattern recognition, as all hits are matched to the pattern banks simultaneously.

Local pattern recognition [17] methods are best employed when the detector geometry allows for sufficiently continuous hits such that it is likely that a neighboring hit belongs to the same track. These methods consider tracks as a sequence of hits and, from a particular starting point, will attempt to extrapolate throughout the detector collecting hits belonging to the same track. A method currently used in ATLAS is known as iterative combinatorial track-finding. This relies on firstly generating track seeds - a few selected hits to act as the initial track candidate. A seed will be typically consist of 3 constructed space-points close to the interaction point, in the inner layers of the tracking detector. Then, there must be a way of parameterising the track in order to extrapolate to other, outer layers, to include further hits. This extrapolation is performed from the inside-out, extending from the seed to the outer layers using a Kalman filter [18]. Finally, the track candidates must be assessed by some quality criterion to distinguish real tracks from fake tracks. Track candidates are then passed to the track-fitting stage to obtain more precise estimates of the track parameters.

2.3.2 Track Fitting

Ultimately, the patterns generated through pattern recognition are the artifacts left behind when charged particles traverse the detector volume. In order to be useful for physics analyses, we require reconstruction of the particle kinematics

from these patterns. There are several parameterisations of tracks, but since the shape of the charged particle trajectories in a uniform magnetic field is helical, in general 5 parameters are required to approximate the true trajectory. One such parameterisation is known as perigee:

$$\boldsymbol{p} = (d_0, z_0, \phi_0, cot(\theta), Q/p_T),$$

where the definitions of the parameters are as follows:

- d_0 = transverse impact parameter - distance of closest approach to the beam-line (z-axis).
- z_0 = longitudinal impact parameter - z-coordinate of the point of closest approach to the z-axis (where d_0 is calculated).

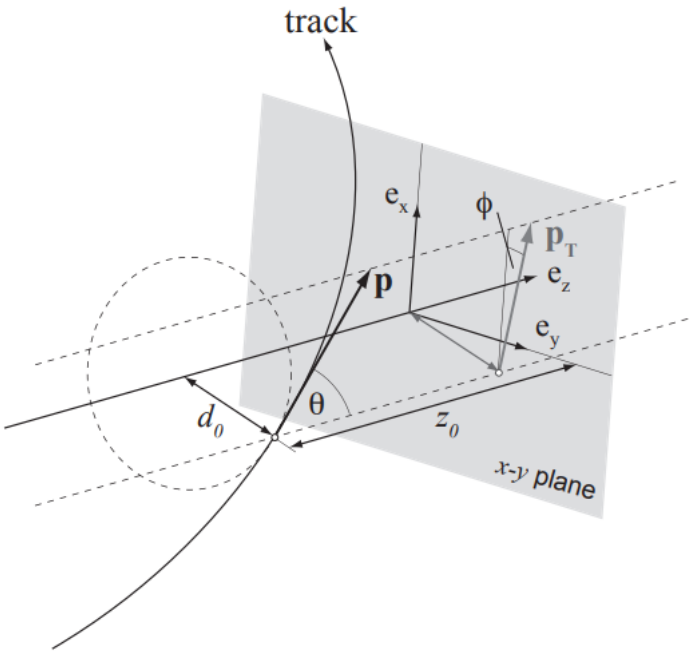


Figure 2.9: **An illustration of the perigee track parameters in ATLAS. The point of closest approach is defined by the signed transverse impact parameter d_0 and the longitudinal impact parameter z_0. The direction of momentum is defined using the global angular coordinates, ϕ and θ. [19]**

- ϕ_0 = azimuthal angle - angle in the $x - y$ plane for the point of closest approach.
- $\cot\theta$ = inverse slope of the track in the (r, z) plane, where θ is the polar angle.
- Q/p_T = inverse transverse momentum multiplied by the charge of the particle.

An illustration of these parameters within the ATLAS coordinate system is shown in Figure 2.9. One method used for fitting these parameters to the observed hit measurements is a global χ^2 fit. For each hit in the track candidate a residual can be calculated, defined as the distance between the track prediction and the centre of the cluster on the plane defined by the sensor surface. Then, the global fit aims to minimise the function:

$$\chi^2 = \sum_{i=1}^{N} \frac{r_i^2}{\sigma_i^2} \tag{2.3}$$

where r_i^2 are the aforementioned residuals for each hit and σ_i^2 are their associated uncertainties. Note that this equation is only correct for 1D measurements.

In the case of the combinatorial Kalman filter [20], the track finding and track fitting stages occur simultaneously. As the seeds are propagated through the detector, the fit predictions can be used to decide which measurements are compatible with the existing track candidate. A number of branches can then be formed with those leading to inactive regions of the detector being dropped, and some threshold on the goodness-of-fit χ^2 limiting the total number of paths.

2.4 Motivation for the HL-LHC

Whatever the long-term objectives of the experiments at the LHC, success lies in our ability to collect more data. With Run 3 having begun at the time of writing and finishing in 2024, operating with an instantaneous luminosity of 2×10^{34} $\text{cm}^{-2}\text{s}^{-1}$, by the end of the run at least $\mathcal{L} = 300$ fb^{-1} of data will have been collected by ATLAS. Since a measurement's statistical uncertainty is proportional to $1/\sqrt{\mathcal{L}}$, a linear rise in the machine's luminosity will bring a relatively stagnant improvement in the precision of measurements. To continue to improve the discovery power, precision and exploration potential, the luminosity will be increased to 5×10^{34} $\text{cm}^{-2}\text{s}^{-1}$ initially for the HL-LHC (2.5 times the Run 3 luminosity), increasing up to 7.5×10^{34} $\text{cm}^{-2}\text{s}^{-1}$ by the mid-2030s. The physics programme offered by the HL-LHC is vast [21] and some of the physics objectives in this programme will be outlined here to motivate the upgrade, and the corresponding challenges for tracking in ATLAS will be addressed.

2.4.1 Physics Motivation

Since the discovery of the Higgs boson at the ATLAS and CMS experiments [22] [23] in 2012, the study of the Higgs sector has greatly expanded to include many precision measurement analyses, new ideas and predictions from theory and searches for rare production and decay processes. One important question to answer is whether the observed Higgs is that predicted from the electroweak symmetry breaking mechanism [24] or if it is, in fact, the first signal in some Beyond Standard Model (BSM) physics. With the accumulated data so far, the identity of the Higgs boson is consistent with SM predictions and all measurements are confined to the couplings of the Higgs to SM particles, which are proportional to the particles' masses, but further precision measurements of these couplings

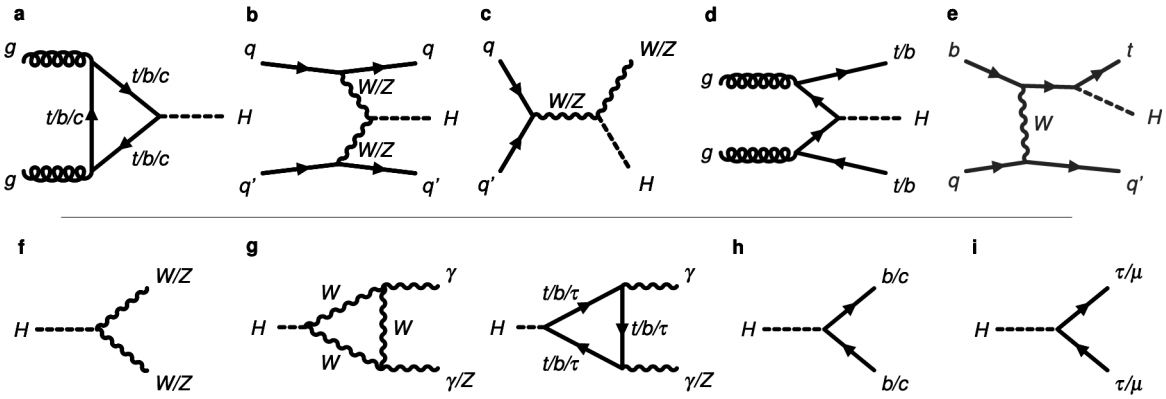


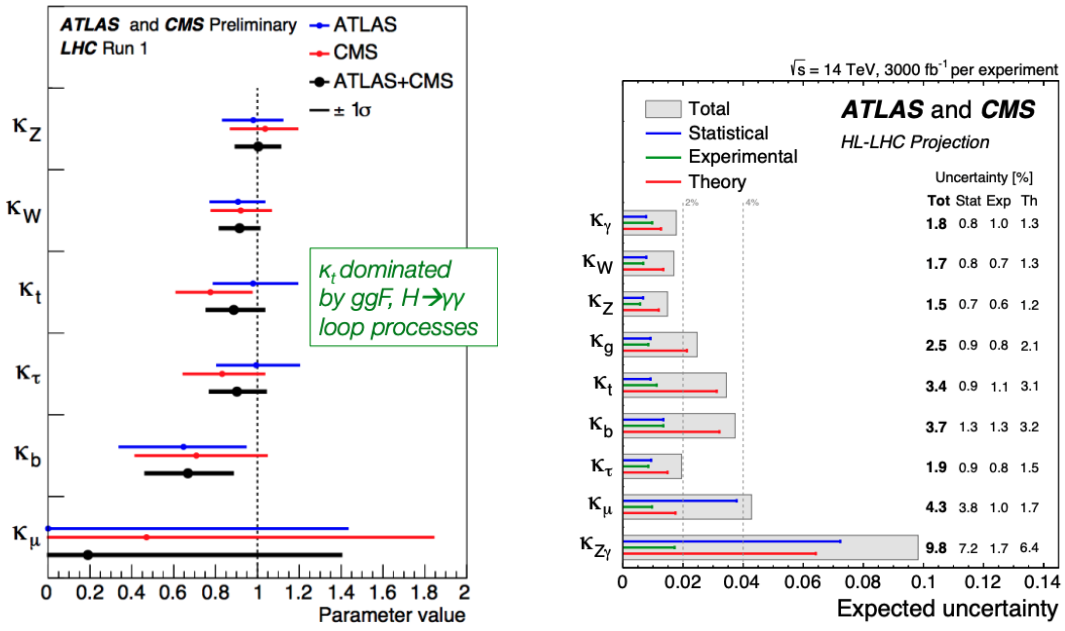
Figure 2.10: **Examples of Feynman diagrams for Higgs boson production and decays. The Higgs boson is produced via gluon–gluon fusion (a), vector-boson fusion (VBF; b), and associated production with vector bosons (c), top- or -quark pairs (d), or a single top quark (e). f–i, The Higgs boson decays into a pair of vector bosons (f), a pair of photons or a boson and a photon (g), a pair of quarks (h), and a pair of charged leptons (i). [25]**

could illuminate any potential discrepancies from prediction. The main production and decay channels for the Higgs boson are shown in Figure 2.10.

The Higgs couplings to the SM particles, including effective couplings to gluons and photons, can be probed in the κ-framework where for a given production or decay mode, we define:

$$\kappa_i^2 = \sigma_i/\sigma_i^{SM} = \Gamma^i/\Gamma_{SM}^i \tag{2.4}$$

where σ_i and Γ^i are the cross-sections and decay widths for the process, respectively. The set of κ_i values then parameterises the potential deviation from SM predictions. For example, κ_g denotes the coupling modifier for the ggH production channel. The κ_i uncertainties in Run 1 and projected uncertainties using combined ATLAS + CMS data in the HL-LHC phase are shown in Figure 2.11. The theory uncertainties are approximated by reducing the present theory uncertainties by a factor of two, to reflect the improved HL-LHC parton distribution functions. With the exception of κ_μ and $\kappa_{Z\gamma}$, which are the rarest decay channels,



(a) **With all data from Run 1 [26]**

(b) **Expected uncertainties for the combination of ATLAS and CMS extrapolations in HL-LHC regime. [27]**

Figure 2.11: **Summary plots showing the total $\pm 1\sigma$ uncertainties on the per-decay-mode branching ratios normalised to the SM predictions.**

the total projected uncertainties from the HL-LHC are dominated by theoretical uncertainty. Additionally, each coupling modifier κ_i is reduced by approximately a factor of 10 when compared with the Run 1 uncertainties.

Another important property of the Higgs boson whose measurement could be a promising window into new physics is its trilinear self-coupling. This comes from the Higgs Lagrangian [28]:

$$\mathcal{L}_h = \frac{1}{2}(\partial^\mu h)^2 - \lambda v^2 h^2 - \lambda v h^3 - \frac{1}{4}\lambda h^4 + \frac{1}{4}\lambda v^4 \tag{2.5}$$

From this, we see the h^2 term that corresponds to the physical Higgs boson, and two higher order terms, describing the trilinear and quartic self-interactions. The Feynman diagrams for these interactions is shown are Figure 2.12. In the SM, with the mass of the Higgs boson measured, the value of λ for both the trilinear and quartic interactions are completely determined,

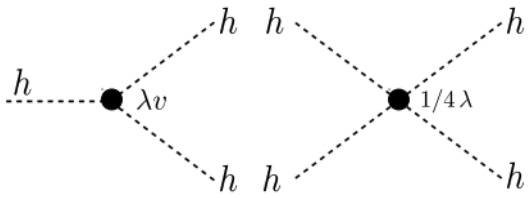


Figure 2.12: **Feynman diagrams for the Higgs boson self-interactions, showing the trilinear (left) and quartic (right) interactions.**

$$\lambda_3^{SM} = \lambda_4^{SM} \approx 0.13 \tag{2.6}$$

Since new physics could modify these values, further constraints on, or precision measurements of, the trilinear coupling provides a test of the SM. A direct approach for measuring this coupling involves a search for Higgs pair production with a range of decay channels, e.g. $HH \longrightarrow bbbb$. However, it is expected that with the combined ATLAS and CMS data, the HL-LHC will only achieve a 4σ sensitivity to the di-Higgs signal. The projected likelihood profile as a function of κ_λ is shown in Figure 2.13. Similarly to Eq. 2.4, κ_λ is defined as the deviation of the trilinear coupling from SM prediction ($\kappa_\lambda = \lambda/\lambda_{SM}$). Depending on the value of κ_λ, and hence whether or not new physics is at play, there may be a difference in the number of total events in the HH signal, and as a result the likelihood profile contains a secondary minimum. This minimum can be excluded at the 99.4% confidence level, and at a 68% confidence level the self-coupling can be constrained to $0.5 < \kappa_\lambda < 1.5$. Furthermore, the HL-LHC will be able to make precise measurements of single Higgs production and decay processes, which can be used to indirectly constrain the trilinear coupling [29], since variations in κ_λ will affect the differential cross sections. In general, precision measurements of the Higgs sector provide indirect probes to just about any extension of the SM. There are many theoretical particles predicted by various BSM scenarios that can also be

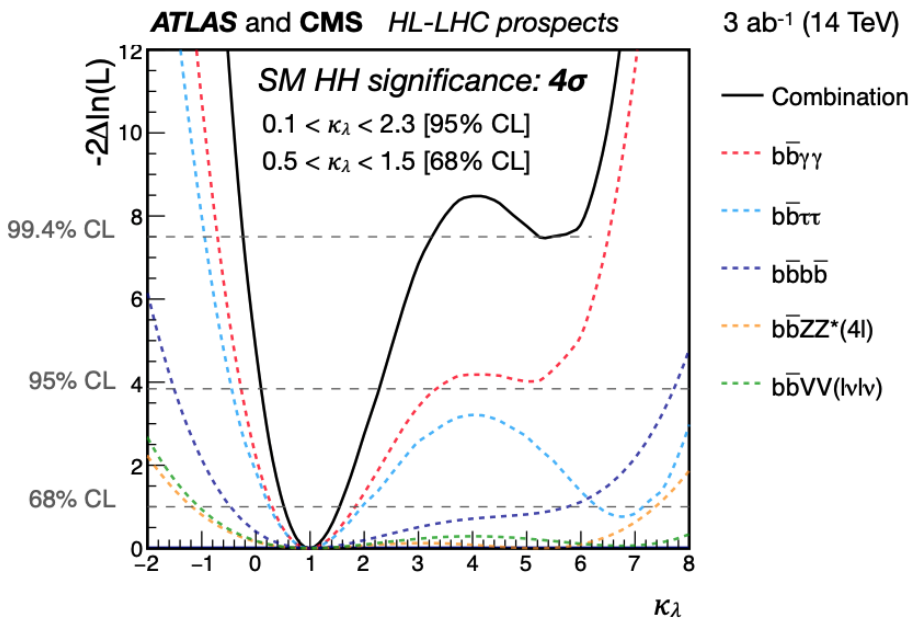


Figure 2.13: **Left: Projected combined HL-LHC sensitivity to trilinear coupling from direct search channels. Right: Sensitivity to BSM Higgs bosons in the $H/A \longrightarrow \tau\tau$ channel. [27]**

searched for in the HL-LHC. One set of such scenarios falls under the title of Supersymmetry (SUSY) [30], which predicts superpartners belonging to every fermion and every boson. The Minimal Supersymmetric Standard Model (MSSM) [31] proposes the minimum number of additional particles, for which mass constraints can be directly made in the HL-LHC. Direct dark matters searches can also be probed at higher mass scales, and new detector upgrades will facilitate searches for long-lived exotic particles. Additionally, BSM physics can be further probed through rare b and c hadron decays that may be measured using the increased integrated luminosity [32].

2.4.2 Necessity for a Tracking Trigger

Regardless of whether SM precision measurements are being made or BSM searches conducted, the identification of charged particle trajectories is key to many different physics analyses at the HL-LHC. The increased level of pile-up poses a chal-

lenge to the trigger system, since many interesting signals require the detection of low-p_T leptons (e.g. leptonic decays from W/Z/H). Hence raising the trigger thresholds would reduce the volume of data at the cost of weakening physics analyses. At present, only information from the muon system and hadronic calorimeters are used in the L1 trigger. One significant improvement that can be made to the tracking system in ATLAS is to include a fast and efficient capacity for track reconstruction within the L1 trigger. Track reconstruction currently occurs in the high level trigger (HLT), but there are many benefits to providing precise measurement of track parameters at an earlier stage. Event rates can be reduced by filtering events based on lepton tracks independent of activity elsewhere in the detector, and associating tracks with calorimeter clusters can improve electron identification. Furthermore, measurements of global quantities, such as the missing transverse momentum, can be improved by using reconstructed tracks to measure primary collision vertices. Completely new analyses may also be conducted with the aid of a tracking trigger that are were not previously possible due to dominant background processes, such as direct searches for long-lived particles [33]. ATLAS has proposed several solutions for tracking for the trigger [34] [35], and in the next section we will outline the plan for the Hardware-based Tracking for the Trigger (HTT), from which the pattern banks used in this work are derived.

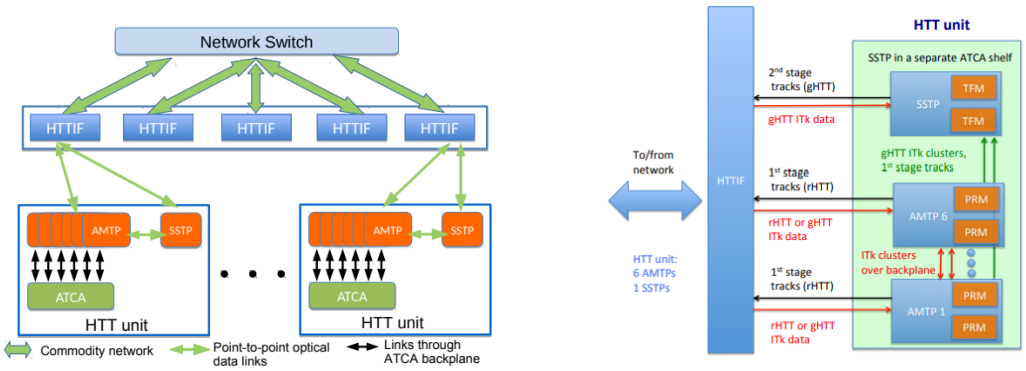
2.5 Hardware-based Tracking for the Trigger (HTT)

2.5.1 Design & Architecture

The design of the HTT is based on that of the Fast TracKer (FTK) [34], a hardware-based tracking upgrade planned for Run 3. The HTT consists of independent tracking units known as HTT units, each containing two types of tracking processors: Associative Memory Tracking Processors (AMTP) and Second-Stage Tracking Processors (SSTP), each of which connect to the Event Filter (EF) processor farm through HTT interface units (HTTIF). The AMTP performs the task of pattern recognition, with each unit holding two Pattern Recognition Mezzanines (PRMs), and the SSTP then provides track-fitting with two Track-Fitting Mezzanines (TFMs). Within each HTT unit, there is a single SSTP and six AMTPs which can perform both regional and global tracking, as per the requests of the EF.

2.5.2 Pattern Banks and Pattern Matching

For the current level of pileup in the ATLAS detector, a local method of pattern recognition known as seeded combinatorial track following meets the data-flow



(a) Overview of the HTT system

(b) Diagram of HTT unit

Figure 2.14: **Schematic of HTT design. [36]**

and reconstruction time requirements, but newer techniques must be sought out as pile-up increases. To outline the pattern matching [17] [36] technique planned to be used by the HTT, a description of patterns and pattern banks is necessary.

First, the clusters formed from individual hits in the strip layers are converted to superstrip measurements, referred to as a SuperStrip Identifier (SSID), where a superstrip is defined as a collection of neighboring strip channels. The SSID is contained within a detector element, and so a hit is defined by its SSID and a detector element ID. For the pixel layer the same is true, with an additional column measurement. We then define a pattern as a set of hits in 8 layers of the tracker, with at least 1 hit fired in the pixel detector. Additionally, each superstrip has a variable width defined by the number of so-called "Don't Care" (DC) bits. The least significant bits in the superstrip number may be ignored in order to combine similar patterns, reduce the number of overall patterns needed and improve the pattern matching efficiency. Hence, a pattern contains 8 detector element IDs, 8 SSIDs and associated DC bits. A pattern bank is a large collection of such patterns. The pattern banks that are used to investigate the simulation of HTT pattern matching are produced by Monte Carlo generated single muons, and separated into 1256 different regions of 0.2×0.2 in $\eta - \phi$ space, each with 2 million patterns (i.e. each pattern bank consists of a collection of 2 million patterns for a single 0.2×0.2 region). The muons are produced with $|z_0| < 150$mm and a flat p_T distribution with $p_T > 1$ GeV up to 400 GeV. Approximately 100 million muons (per 0.2×0.2 region in $\eta - \phi$) are simulated and very high granularity, neighbouring patterns can be grouped together to reduce the total number, such that the pattern matching efficiency is maximised with respect to muons and minimised with respect to pile-up. To effectively model inactive regions in the detector, some patterns will have 'missing hits' known as wildcards in one or two layers. A simple schematic illustrating the composition of a single pattern within these pattern banks is given in Figure 2.15.

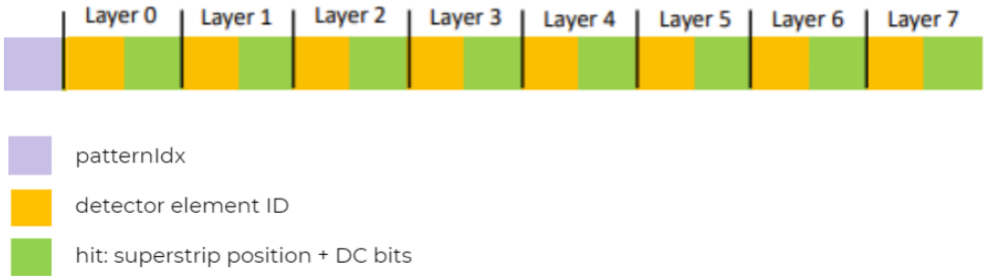


Figure 2.15: **Schematic illustrating the structure of a pattern in the pattern bank, containing a unique pattern index followed by 8 detector element IDs and 8, more granular, SSIDs.**

The implementation of pattern matching used in the HTT is as follows. For each event, the hits are iterated through and any pattern associated with that particular hit (within the allowance of the DC bits) is stored and a counter is incremented for the pattern. If the pattern contains a wildcard in the layer at hand then it is assumed to match. If the counter for any pattern reaches the pattern matching criteria of 7 out of 8 matched layers, the pattern is stored in an output of matched patterns. Hence, the pattern matching efficiency is defined as the percentage of reference muon tracks that are matched to any pattern in the pattern bank. If the efficiency is low, then the pattern bank either contains too few patterns to match all of the muons, or the patterns themselves do not represent the muon tracks precisely.

Chapter 3

Hough Transform Literature Review

In this chapter, the broad history and various applications of the Hough Transform (HT) will be covered. The technique has been employed for decades in the imaging world as a way to detect lines and other shapes in images and has found its place in several high-energy physics experiments. In order to present a clear picture of the methodology used in this work, it is important to understand why this technique has become so useful in this field and exactly how it is implemented. The different applications to tracking in high energy physics produce varying results and we will outline why this is and how this could be improved upon. It is within this context that the more novel aspect of our implementation can be introduced and then explained further in the following chapter.

3.1 History of the HT

3.1.1 Transform Origin

It is quite suitable that the very transform used in this work to develop a novel tracking algorithm was invented in a search for an automated way of detecting the tracks of subatomic particles in bubble chambers [37]. However, the original invention, which is detailed in a US patent [38] by Paul Hough in 1962, only faintly resembles the transform as we use it here. The purpose of the transform is to convert the problem of detecting a line (or a set of colinear points) in an image into the problem of finding intersecting lines in a transform space. The original patent provided no mathematical formulation for the transform or parameterisation of the transform space, but instead contains a graphical description of the point-to-line mapping that the transform essentially performs. This is shown in

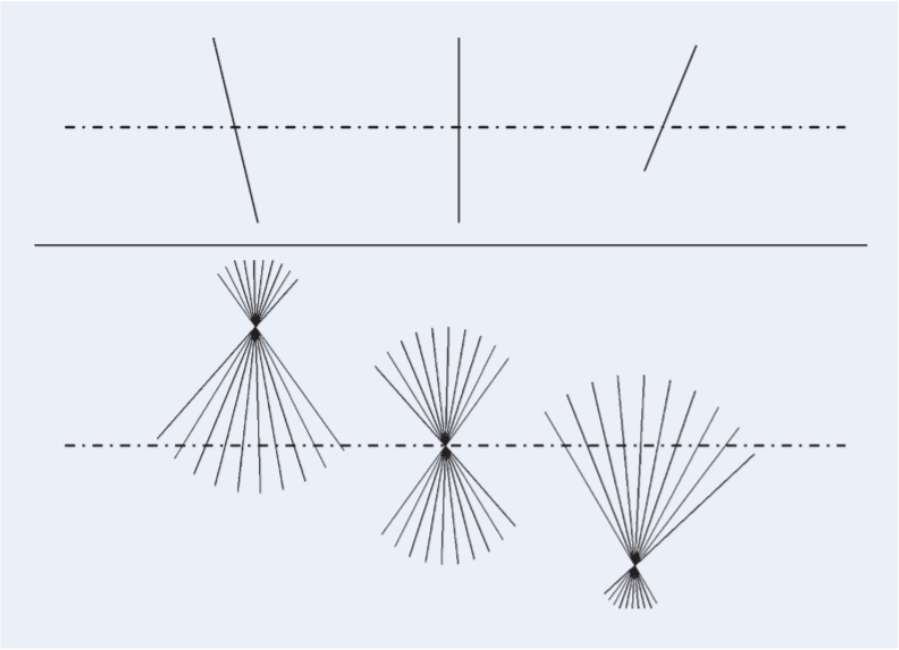


Figure 3.1: **A graphical description of the Hough Transform, as drawn in the original patent. A single point in the upper image space corresponds to a line in the lower image space, such that a colinear set of points maps to a set of lines with an intersection or 'knot'. [38].**

Figure 3.1, where we can see that the colinear points on a given line map to a set of lines that intersect at a single point, referred to by Hough as a knot. This general description is the essence of the Hough transform, which can be thought of as the discrete version of the Radon transform [39], used frequently in biomedical imaging. The obvious benefit of a discrete parameter space in which the transform can be performed is that the execution time is far quicker, hence the more prevalent application in computer vision. The first mathematical formulation of the Hough Transform comes in the book *Picture Processing by Computer* by Azriel Rosenfeld [40] in 1969. Here, the patent is referenced and the transform is defined algebraically by:

$$y = y_i x + x_i \tag{3.1}$$

where x and y are the coordinates of the transform plane and (x_i, y_i) are points in the image plane. This choice of parameterisation retains the property of the transform that if a set of points are colinear, they will intersect at a single point in the transform space. It is also here that, for the first time, the transform space is suggested to be defined as an array of counters, referred to as an accumulator space. In this way, the presence of many superimposed values of 1 result in a high value in the array, allowing the knot or intersection point to be easily identified. In many ways, this implementation of the HT more closely resembles the well known HT used throughout the field today, in that the parameter space is properly defined and is structured as an accumulator, with points in the original image casting "votes" along lines in the parameter space. However, it was in 1973 in a paper by Duda Hart [41], and thanks to an innovation by mathematicians in the field of integral geometry [37], that a more sensible parametric representation was first chosen:

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{3.2}$$

which allows every possible line in the image plane to correspond to a unique point in the transform plane. In other words, there can be a one-to-one point-to-line mapping between image and parameter space, which is not possible when the parameter space is unbounded as with the slope-intercept representation given by Eq. 3.1. However, this problem can also be circumvented using a Cascaded HT [42], which splits the unbounded parameter space into several bounded spaces. The normal representation shown in Eq. 3.2 would go on to be used for decades in textbooks and courses, and comprises the backbone of many different advanced and modified techniques.

3.1.2 Variants and Extensions

Since the formulation of the classical line-detecting HT, different variants have been used to detect circles, ellipses and many more arbitrary shapes. A simple diagram in Figure 3.2 shows the general components of the standard HT. First a transform is performed into the defined parameter space which is constructed as a quantised accumulator, referred to as the accumulator or Hough space. Votes are cast for bins in the array by points in the image, and once all votes are cast some method of peak detection is used to locate instances of a line or object. It may seem natural to extend the transform to more complex shapes if there is an explicit parametric representation available, such as:

$$(x-a)^2 + (y-b)^2 = r^2 \tag{3.3}$$

for a circle, where the parameters (a, b) define the position of the circle and r defines the radius. However, the dimensionality of the parameter space is now

3-D, with a circle in the image space being represented by a cone in the parameter space, an example of which is shown in Figure 3.3a. The higher dimensionality results in significantly higher storage requirements and a far greater space to search for peaks in. As a result, there have been several attempts to avoid a brute force approach when detecting more complex shapes. The conventional approach outlined by Kerbyson [45] is equivalent to applying a convolution of a circle of defined radius R with the image. By removing the variable R, each edge point in the image plane produces a circle in the (a, b) parameter space and a peak will arise at the intersection point of the set of circles. Then for the full calculation, a scan of the possible values of R must be conducted.

While several modifications were made to the classical HT [46] [47] [48], a more general approach was later formulated that can detect any arbitrary shape. Figure

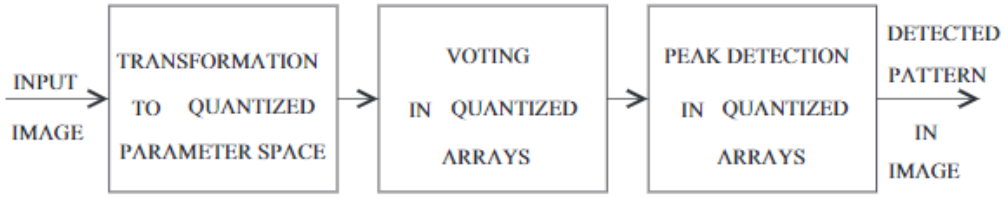
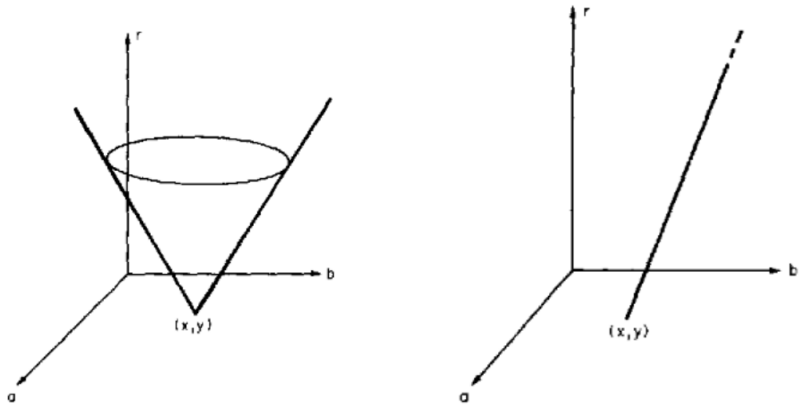


Figure 3.2: **A block diagram of the general Hough Transform process. [43]**



(a) **without directional information**

(b) **with directional information**

Figure 3.3: **The locus of parameters for a single point on a circle. [44]**

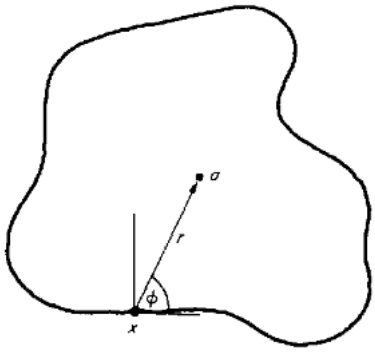


Figure 3.4: **An example of an arbitrary shape to be detected using GHT.**

3.3 is pulled from the work of Ballard [44] which first defines the Generalised Hough Transform (GHT) as a two-stage process involving learning and then recognition. One important improvement shown in 3.3b utilises the gradient direction of the edge point. By leveraging this additional information, the locus of parameters is reduced to a line. The same information is key to detecting arbitrary shapes like the example shown in Figure 3.4. The first learning stage of the GHT consists of choosing a reference point for the shape, $\mathbf{y}$ (essentially a local origin, analogous to the centre of a circle). For each edge point $\mathbf{x}$, the gradient direction $\phi(\mathbf{x})$ is found and the variable radius, $r = \mathbf{y} - \mathbf{x}$ is stored as a function ϕ in what is referred to as an R-table. Once the local gradient properties have been mapped in this table, the next stage is recognition. In the example of circle detection

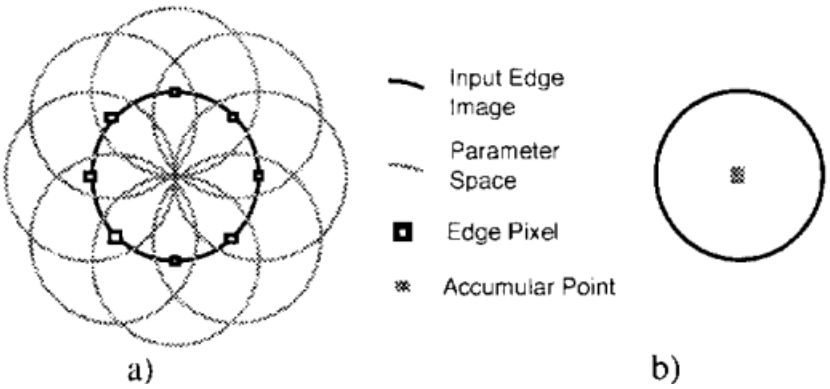


Figure 3.5: **(a) The projected edge points in the Hough space, (b) A single contribution of an edge point to the Hough space. [45]**

in Figure 3.5, for a fixed radius r, the projected circle is drawn in the Hough space for each edge point. Similarly, in the GHT, although the radius is not fixed but specific to each edge point and defined relative to the reference point $\mathbf{y}$, it can be extracted from the R-table depending on the gradient ϕ. Then for each boundary point $\mathbf{x}$, the circle $\mathbf{x}$ + r can be incremented in the Hough space and then, as usual, peaks in this space correspond to instances of the shape. While this is only a brief description of the process, Ballard [44] outlines in detail how this can be used to detect even composite shapes. The extension of this upon the classical HT is summarised in the block diagram shown in Figure 3.6, which shows that the parametric transformation is essentially replaced with the construction of the R-table. As well as advancements of the technique's ability to detect any

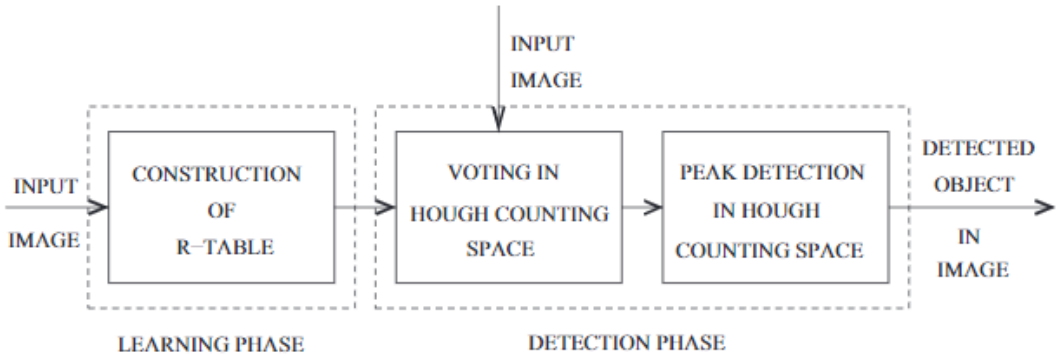


Figure 3.6: **A block diagram of the GHT Hough Transform process. The distinction with respect to the classical HT lies in the learning phase and construction of an R-table, since the explicit transformation is not possible with non-analytic shapes. [43]**

shape, there have been several variants developed to speed up the implementation. Examples of these include the Probabilistic HT [49] and Randomised HT [50], which involve selecting a subset of data points or pixels from the image plane such that the overall reduction in the ability to detect features is negligible. Another probabilistic approach based on the idea that shapes in images will be approximate forms of the exact parametric representation is the Fuzzy HT [51], in which each data point corresponds to a fuzzy region in the Hough space.

3.1.3 Probabilistic Inference with Explaining Away

An important addition to the HT library, for our investigation, is the concept of *explaining away*, a method for reducing the presence of spurious peaks in the Hough image. In order to introduce this technique, it is useful to consider the original image and its pixels as voting elements. Similarly, each point in the Hough space can be thought of as a hypothesis about an object being present in the image at a particular location. The transform then determines, for each pixel, which hypotheses it will vote for in the Hough space. A peak in this space is the result of a large number of votes being cast by different elements in the original image, i.e. the presence of an object or shape. Formulating the transform in this manner provides an appropriate framework to introduce a probabilistic model first introduced by Barinova et al [52]. Here, an example is outlined in which the maximum of the Hough image corresponds to a correctly identified object. The elements that voted for this peak may well cast strong votes for other instances as well. There is, as far as we have outlined so far in this Section, currently no mechanism built into the Hough transform that reduces the impact of these

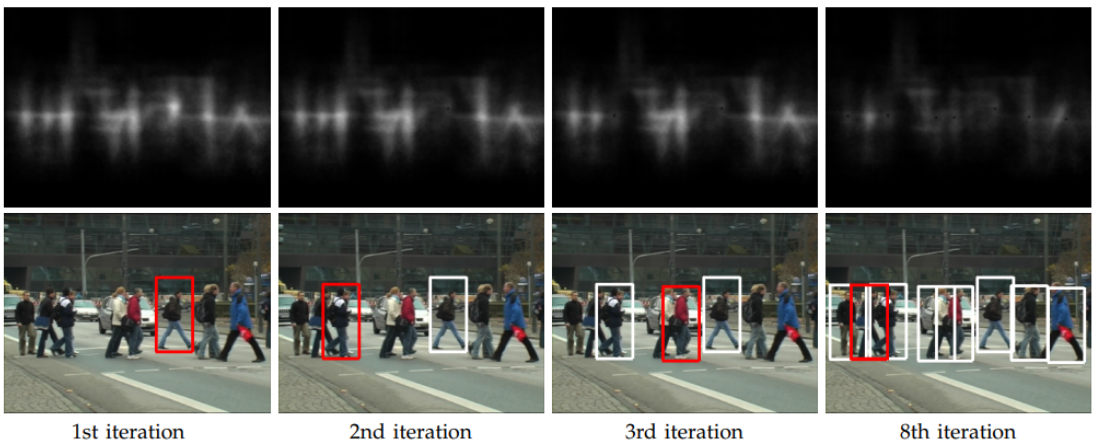


Figure 3.7: **A demonstration of the effect of explaining away on the Hough image and ability to detect multiple instances of an object. As the iterations increase, the Hough image changes and multiple pedestrians are detected despite overlap between instances [52].**

additional votes, despite the fact that the same elements have cast their votes for a more significant peak. This is the principle behind explaining away - to introduce probabilistic interference such that, once an element votes for a local maximum, the weight assigned to its votes for other hypotheses is reduced. By doing so in an iterative way, the presence of spurious peaks can be greatly reduced, and as described in [52], integrating this with the HT allows for efficient detection of multiple object instances. This ability is showcased in a pedestrian-detection task in Figure 3.7.

3.2 HT in High Energy Physics

Given that the original application of the HT in particle physics was to track particles in a bubble chamber, it is not surprising that the technique has been used in the LHC experiments for the same purpose. Both of the general purpose detectors, ATLAS and CMS, have investigated the potential of the transform for charged particle tracking in different ways and these attempts will be discussed briefly here.

3.2.1 Global HT Tracking in ATLAS

As we have described already in this chapter, the HT requires a parametric representation of the object for detection. Additionally, the perigee track parameterisation used in ATLAS was defined in Section 2.3.2. These are summarised here in Table 3.1.

Parameter	Description
d_0	Transverse Impact Parameter
z_0	Longitudinal Impact Parameter
ϕ	Azimuthal Angle at Origin
θ	Polar Angle at Origin
$\mathrm{Q/p}_T$	Charge-Signed Curvature

Table 3.1: **Summary of the perigee track parameters used in ATLAS. [53]**

Since two parameters would be the ideal number for computing a Hough Transform, ϕ and q/p_T are chosen to construct the Hough space. The assumption that $d_0 = 0$ is made, since the outer layers of the ITk are approximately 30cm from the beam line, compared with the negligible $|d_0| < 3$mm that is the case with tracks coming from, for example, b-hadron decays. The transformation from the xy plane to a (ϕ, q/p_T) plane can be derived from Figure 3.8. Here, several track

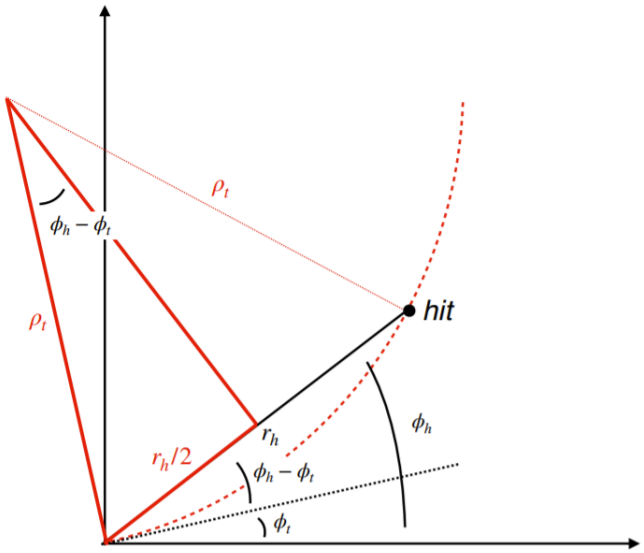


Figure 3.8: **Definition of track variables for the global track parameter HT. ρ_t is the radius of the curvature of the track, ϕ_t and ϕ_h are the azimuthal angles for the track at origin and at the hit, respectively, and r_h is the distance from the origin to the hit. [54]**

variables are illustrated such as the radius of the curvature of the track, ρ_t, the azimuthal angles of the track at both the origin and at the hit, ϕ_t and ϕ_h, and the distance from the origin to the hit, r_h. From the diagram, the following relation can be deduced:

$$\sin(\phi_h - \phi_t) = \frac{r_h}{2\rho_t} \tag{3.4}$$

which, using $\rho_t = p_T/0.3B$ can be rearranged to:

$$\sin(\phi_h - \phi_t) = \frac{0.15 r_h B}{p_T} \tag{3.5}$$

and since $\phi_h - \phi_t$ is assumed to be very small, we can use the linear approximation of $\sin(\phi_h - \phi_t)$ to reach:

$$\phi_t = \phi_h - \frac{0.15 r_h B}{p_T} \tag{3.6}$$

Figure 3.9 shows an illustration of the transformation of several hits using this

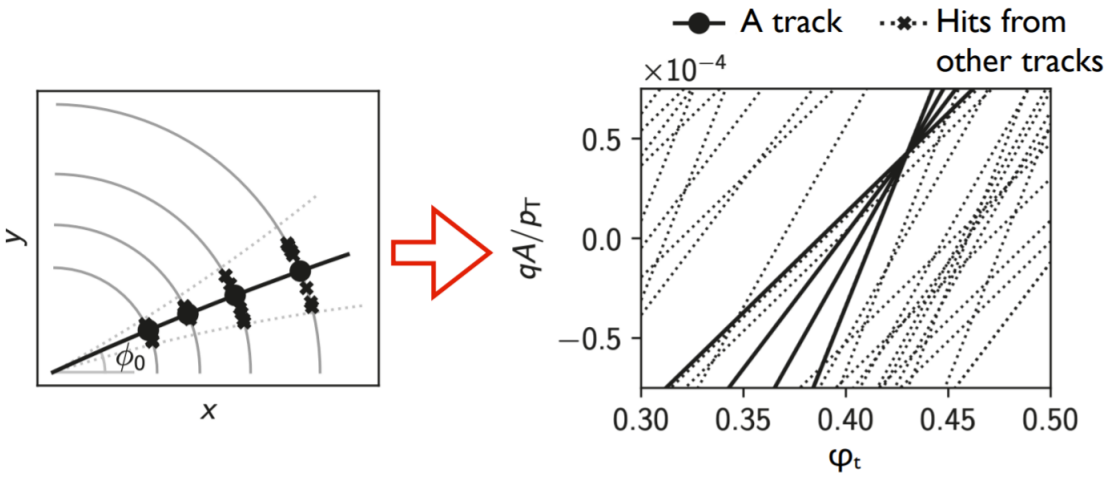


Figure 3.9: **Illustration of a track in detector space and the corresponding transform into the global track parameter Hough space. [54].**

parameterisation. As the image shows, each of the hits lie along a similar direction and the projected track forms an elongated peak. This is expected from Eq. 3.6, which shows that all lines in this space will have a positive gradient and hence the intersections are spread out in one direction. Additionally, Figure 3.10 shows the summation of histograms across 8 layers including pile-up hits. The density of hits in the image is very high and, although the tracks may be identified qualitatively

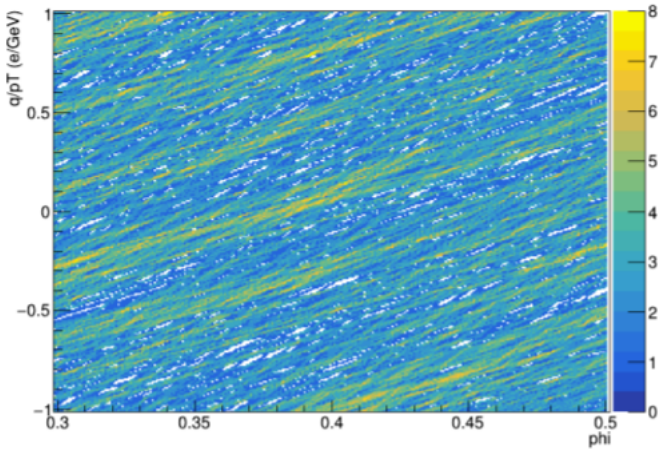
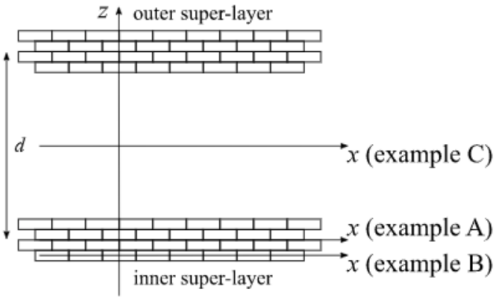


Figure 3.10: **Hough space using the global track parameter representation. The event data contains single simulated muons with pile-up of $<\mu> = 200$ [54].**

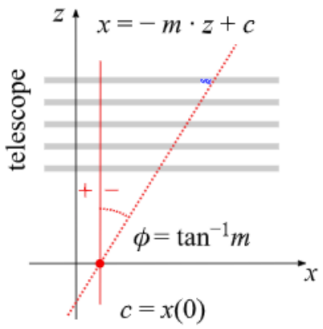
by the diagonal strips, extracting the track parameters with any precision is not easy in this Hough space. Attempts are made to reduce the effect of pile-up using techniques such as hit filtering to reduce the number of hits in the image and "slicing", in which the detector is split into 0.2×0.2 segments in $\eta - \phi$ space. Even so, it is suggested that any Hough space will produce a very large number of combinatorial fake tracks and that further algorithmic refinements would be required to resolve this issue.

3.2.2 CMS Approach

There have also been attempts to employ the Hough Transform for tracking in the CMS experiment. In the work by N. Pozzobon et al. [55], a CMS-like detector layout is used as a case study for implementing a HT-based tracking algorithm focused on straight line parameterisations. The paper outlines this method in detail, but we will only summarise the main points here in order to draw comparisons with the global ATLAS parameterisation and our proposed method. The CMS Drift Tube (DT) detector contains separate tracking stations known as super-



(a) **Layout of the CMS-like DT chamber with three choices for $z = 0$ superimposed to the layout.**



(b) **Representation of straight line parameters with respect to the detector layers.**

Figure 3.11: **Summary of setup for HT track parameterisation in the CMS case study. [55] Angles are measured with respect to the normal direction to telescope layers, and the intercept is measured on a reference plane parallel to the telescope layers.**

layers where measurements of the muon tracks are recorded. A gradient-intercept parameterisation is used to define the tracks as shown in Figure 3.11. The parameters used for the HT are defined in three examples illustrated in Figure 3.11a. There are three different choices for $z = 0$ which defines the location of the intercept - example A with respect to the middle of the inner super-layer, example B at the lowest layer and example C with respect to the middle of the DT chamber. Using these setups, the HT parameters are then defined by the slope:

$$m = \tan(\phi) \tag{3.7}$$

and the intercept:

$$c = x(z = 0) \tag{3.8}$$

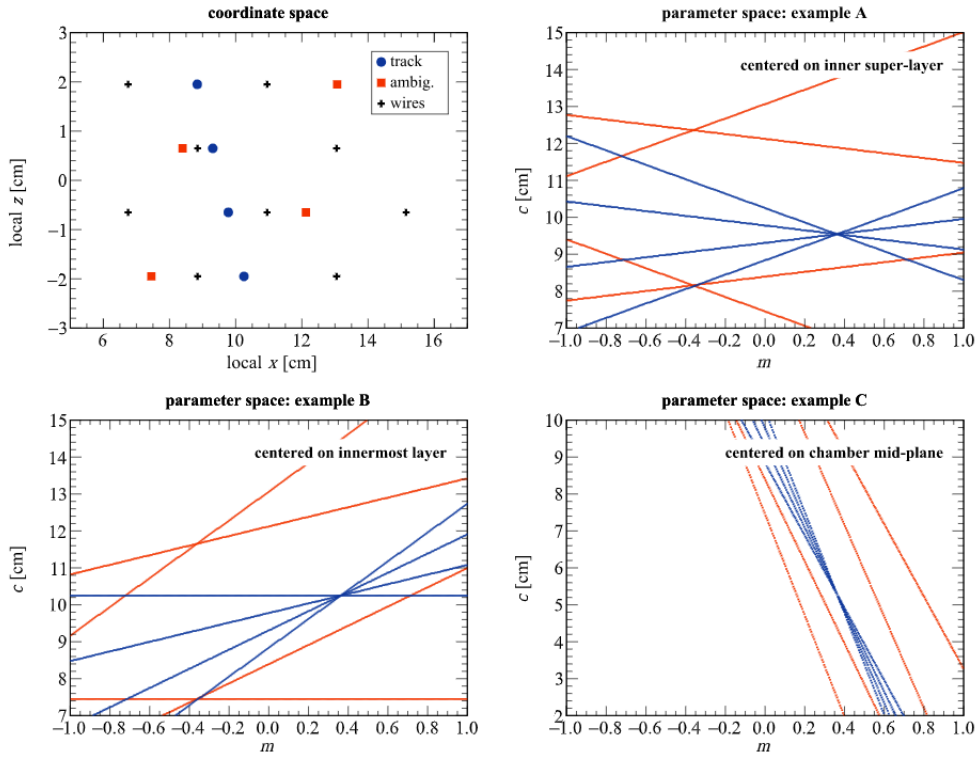


Figure 3.12: **Example of the HT of a straight muon track segment simulated with GEANT4 in the inner super-layer of a CMS DT chamber. The parameter space for each example corresponds to the choice of $z = 0$ as shown in Figure 3.11.**

on a reference plane $x = -mz + c$. Figure 3.12 contains illustrations of several hits transformed into each of the three parameter spaces. The images show that each of the representations produces transformed hits with varying degrees of angular separation. Example C appears to suffer from a similar problem as was outlined in the ATLAS example, in that the hits are almost colinear in the Hough space, producing strip-like peaks. The other two examples, A and B, appear to be more sensible parameterisations as the intersection of the hits can be easily identified. However, since the gradient m is independent of the choice of origin, the different Hough Transforms may be combined to produce improved results.

3.3 HT Tracking using PCA-based Track Parameterisation

The previous attempts to use the HT for tracking have varying degrees of success with respect to the choice of parameterisation. Although we have only qualitatively assessed how suitable the representations are based on the resulting histograms and how well the intersections can be identified, there are clear drawbacks associated with these attempts. Firstly, the parameters are chosen in an ad-hoc way. There is no statistical justification for the choices and the efficacy of the method must be investigated after the fact. In the case of the CMS example, the process of combining different representations was not immediately intuitive and somewhat cumbersome. Additionally, the issue of high image hit density can arise. We saw that in the case of the ATLAS perigee track parameterisation, the Hough images were filled with noise when pile-up was included. We propose an alternative approach that fixes both of these issues.

Firstly, in addition to dividing the detector into 0.2×0.2 regions in η-ϕ space, each region may be further split into a large number of so-called "sectors" defined by a unique set of detector elements. By doing so, the tracking can be performed independently for each sector and the density of hits in the Hough images will be far lower. Secondly, the pattern banks from the HTT can be used as training data for a Principal Component Analysis (PCA)-based pattern data compression. The parametric representation can then be chosen to be the n (ideally two) principal components that explain sufficient variance in the pattern data. In this way, the parametric representation is data driven and images in each layer will be well separated in the Hough space. The resulting images have little noise to handle and relatively circularly symmetric peaks. We can then use probabilistic interference to explain away any spurious peaks and gain competitive track finding efficiency

with fewer track candidates to process. Finally, a track fitting stage is applied to further remove any fake tracks and keep only the highest quality fits. In the next chatper, we will focus on the definition of the compressed pattern space and show its suitability for HT tracking.

Chapter 4

The Compressed Pattern Space

In the last section, it was discussed that we can use PCA to construct a novel parametric representation of tracks in order to perform a Hough Transform. The resulting Hough space is capable of producing well-defined and symmetric peaks that allow for efficient track finding even with high levels of pile-up. Additionally, for any practical implementation of a pattern-based track finding algorithm on massively parallel hardware, such as GPUs, it is crucial to compress the pattern information due to memory size limitations. In this chapter, we describe the lossless dictionary-based compression of Monte-Carlo (MC) generated pattern banks, which organises patterns into so-called *sectors* defined by the unique sets of detector elements containing hits. Next, we use the patterns in each sector to train a PCA model and learn a compressed pattern representation. We will investigate the physical meaning of the resulting PCA components and how they relate to track geometry. Finally, we will report the overall compression factor of the pattern banks and the number of components required to still achieve efficient pattern matching.

4.1 Dictionary-based Compression

For the work in this section, we use ATLAS ITK pattern banks composed of two million patterns from MC generated single muons with $p_T > 4$ GeV (see Figure **??** for distribution). A subset of 8 layers is used to define the pattern bank, and the structure of a single pattern, which was described in 2.5.2, is shown again for convenience in Figure 4.1. We study the compression of pattern banks in 4 different

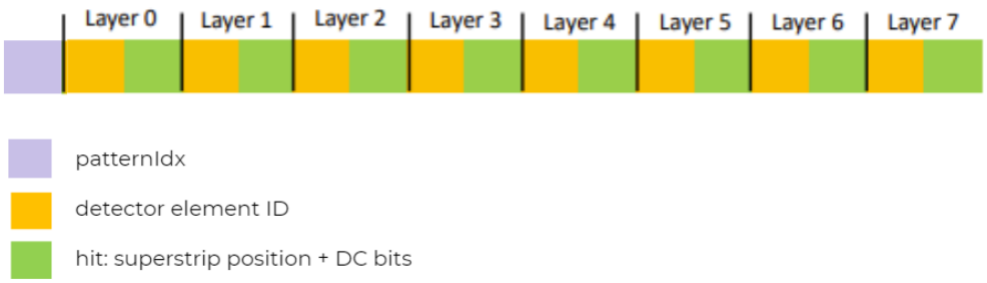


Figure 4.1: **Schematic illustrating the structure of a pattern in the pattern bank, containing a unique pattern index followed by 8 detector element IDs and 8, more granular, SSIDs.**

η regions in the detector, listed in the Table 4.1. These regions cover the central barrel region and out to the end-cap region, and allow us to investigate the effect of the compression algorithms on different regions of interest in the detector. Since the patterns are defined by a coarse set of detector element IDs followed by more granular SSIDs, each of the pattern banks contains some redundancy, as many patterns share the same detector elements and differ only in which superstrips

η **Region**
$0.1 < \eta < 0.3$
$0.7 < \eta < 0.9$
$1.2 < \eta < 1.4$
$2.0 < \eta < 2.2$

Table 4.1: **The different η regions of each pattern bank.**

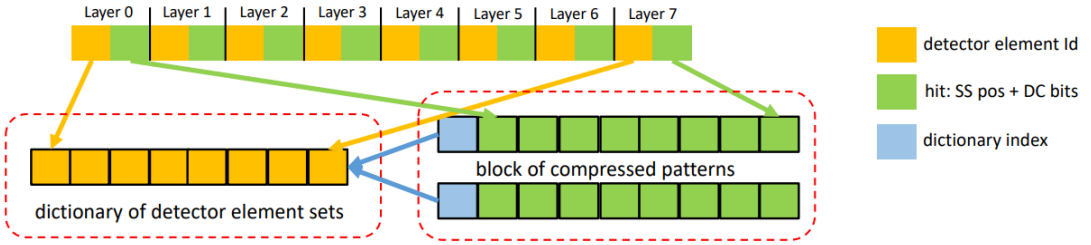


Figure 4.2: **Schematic illustrating the 'factorisation' of a pattern into a dictionary entry defining its sector and a reduced pattern format. Two compressed patterns are shown in the diagram, but in reality there are up to thousands of patterns in each sector.**

fired. If we define a sector as a unique set of 8 detector element IDs, then we refer to the grouping together of patterns into sectors as dictionary-based compression. A visual representation of the procedure is shown in Figure 4.2. Each sector is given a unique ID which is stored along with its detector element IDs in a sector dictionary. The same detector elements can then be dropped in the corresponding patterns within the sector. In doing so, for a sector with N patterns, the set of detector elements only needs to be stored once in the dictionary, as opposed to N times in the pattern bank. The resulting pattern bank, whose detector element IDs have been dropped and replaced with a sector index, will be referred to as a reduced pattern bank (note that the number of patterns is still the same, but the storage requirements have been reduced). With the sectors organised by decreasing pattern population, Figure 4.3 shows the cumulative sum of patterns over all the sectors for the $0.1 < \eta < 0.3$ region. The total number of sectors is around 25000 with approximately half of the sectors containing 90% of all patterns in the bank. With the pattern bank split into a sector dictionary and reduced pattern bank, the lossless compression factor is then defined by:

$$Z = \frac{\text{storage required for full pattern bank}}{\text{storage required for sector dictionary} + \text{reduced patterns}} \tag{4.1}$$

This can be used to assess the level of compression for the pattern banks in each

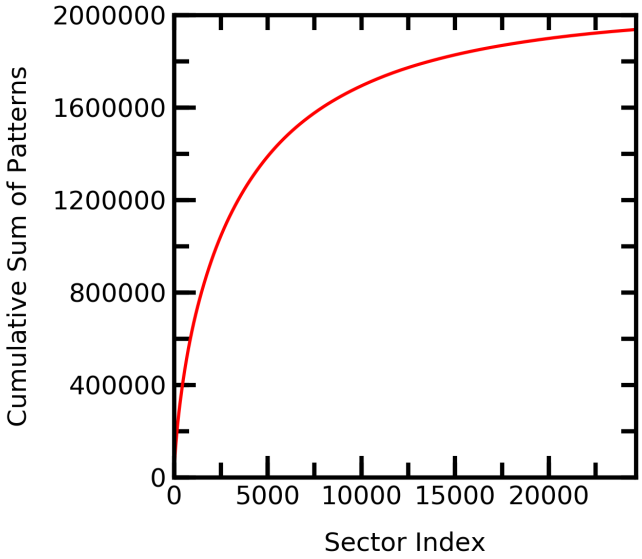


Figure 4.3: **The cumulative sum of patterns with increasing sector index for 2 million patterns in the $0.1 < \eta < 0.3$ region. Sectors are organised in decreasing pattern population.**

of the η regions. Each pattern bank has the same number of patterns (which is determined to be sufficiently high so as to not reduce the pattern matching efficiency), and hence the size of the reduced pattern bank will also be the same across η regions and the only difference in terms of compression will come from the size of the sector dictionary. The compression factor, Z, for each region is given in Table 4.2. The compression factors are very similar as the variable size of the sector dictionary only makes up a small part of the total storage. Overall, the storage required for the pattern banks can be more than halved for each region using this lossless compression. Additionally, by organising the patterns

Eta Region	**Compression Factor (Z)**
$0.1 < \eta < 0.3$	2.089
$0.7 < \eta < 0.9$	2.087
$1.2 < \eta < 1.4$	2.097
$2.0 < \eta < 2.2$	2.143

Table 4.2: **Compression factors for the lossless dictionary-based compression for each eta region.**

into these sectors, we can perform further sector-wise compression using Principal Component Analysis.

4.2 Principal Component Analysis (PCA)

4.2.1 Introduction to PCA

It is common to want to reduce the size of a dataset to allow for easier analysis without removing the statistical potential that can be exploited. A useful tool in statistical analysis that may be used for dimensionality reduction and data compression is Principal Component Analysis (PCA). For a multidimensional dataset, this technique aims to learn along which axes, known as principal components, the data has the greatest variance and perform a change of basis to transform the dataset onto these components. The principal components with the least variance can then be ignored, thereby reducing the dimensionality of the dataset. An example of this on a 2D dataset is shown in Figure 4.4.

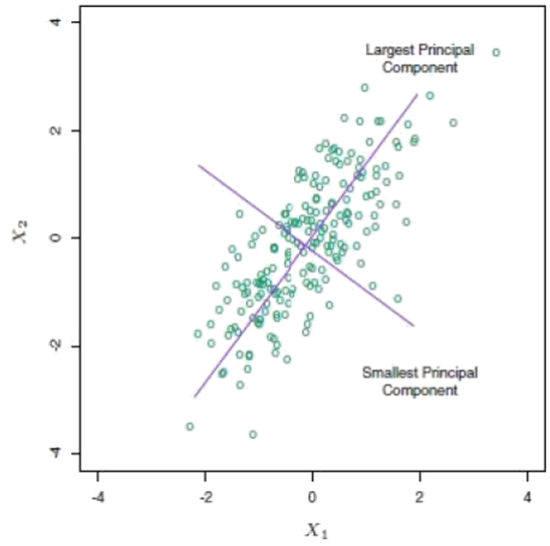


Figure 4.4: **A simple demonstration of the first principal component lying along the axis of greatest variance. [56]**

To describe how PCA is implemented [57], consider m measurements of n independent, inter-correlated variables such that the matrix of measurements can be written as the $n \times m$ matrix:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,m} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & x_{n,m} \end{bmatrix} \tag{4.2}$$

Given that the covariance between two variables is defined as:

$$\mathrm{cov}(x_1, x_2) = \frac{1}{m} \sum_{i=1}^{m} (x_{1,i} - \overline{x_1})(x_{2,i} - \overline{x_2}) \tag{4.3}$$

we can rewrite the matrix $\mathbf{X}$ by subtracting the mean from each measurement and dividing by $\frac{1}{\sqrt{m}}$, such that the $n \times n$ covariance matrix becomes:

$$\mathbf{C} = \mathbf{X}\mathbf{X}^{\mathbf{T}} \tag{4.4}$$

In order to find the principal components, we find the matrix $\mathbf{V}$ which diagonalises the covariance matrix:

$$\mathbf{V}^{\mathbf{T}}\mathbf{C}\mathbf{V} = \mathbf{D}, \tag{4.5}$$

where $\mathbf{D}$ is the diagonal matrix of eigenvalues of $\mathbf{C}$ and $\mathbf{V}$ is the matrix whose columns are the eigenvectors of $\mathbf{C}$. These eigenvalues are sorted in descending order, with the columns of $\mathbf{V}$ being rearranged in a corresponding way. Hence, the first column of $\mathbf{V}$ gives the largest principal component, and the last column give the smallest principal component - the least significant component for describing the data which, along with other components, may be discarded depending on how small the eigenvalues are and how much information one is willing to lose. A measure of this 'information loss' can be gained by the fraction of explained variance of each component, given by the relevant eigenvalue divided by the sum of eigenvalues:

$$f_i = \frac{D_{i,i}}{\sum_{k=1}^{n} D_{k,k}} \tag{4.6}$$

This value will be greatest for the first component and will decrease for each subsequent component. A more useful metric is the cumulative explained variance, which provides an intuitive figure of how much variance is kept once the least significant components are dropped. If, for example, we keep d of the original n components, then the cumulative explained variance is given by:

$$\sum_{i=1}^{d} f_i \tag{4.7}$$

The decision on whether components can be dropped and how many may be dropped can be based on keeping this value as close as possible to one. Once the number of principal components is decided, the dataset is transformed to the basis of new components, reducing the dimensionality of the data.

4.2.2 PCA Compression for Complete Sectors

Until now, it has only been briefly mentioned that many of the patterns in the original pattern banks contain missing hits or wildcards in one or two layers to account for the inactive regions of the detector. Since the detector element ID is not known for these layers, any patterns containing the same remaining detector element IDs and the same wildcard(s) are grouped together into these incomplete sectors. The categorisation of sectors can then be summarised as *complete* (containing no wildcards) or *incomplete* (containing one or two wildcards). The proceeding steps for PCA compression rely on this distinction and to begin with, it is instructive to consider only the complete sectors before moving on to discuss a sector-linking step which accounts for the incomplete sectors.

In order to perform the PCA-based compression on a sector of patterns, the dictionary index and pattern index are removed from the pattern and the DC bits

are removed from each layer. The remaining pattern data is a 9-dimensional vector consisting of two measurements for the pixel layer (row + column), followed by 1 superstrip measurement for the remaining 7 layers. The number of possible values for column and row positions in the pixel layer, as well as superstrip positions in the strip layers, are dependent on the η region due to the larger detector elements in the endcap region, and they are summarised in Table 4.3.

η Region	Col (Pixel)	Row (Pixel)	SS (Strip)
$0.1 < \eta < 0.3$	2	22	32
$0.7 < \eta < 0.9$	2	22	32
$1.2 < \eta < 1.4$	2	22	32
$2.0 < \eta < 2.2$	4	44	152

Table 4.3: **Number of possible values for column and row positions in the pixel layer, and superstrip (SS) positions in the strip layers, at different η regions. The values are used to encode/decode pattern data into the compressed `uint8` format.**

Additionally, the column measurement is a categorical variable and may not be appropriate to include in the PCA. [58]. In this case, however, the proposed solution is to split each sector into the same number of sub-sectors as the number of column values, so that all patterns in these sub-sectors share the same column value and the variable can be dropped. The row measurement in the pixel layer is then essentially treated as a superstrip measurement. The decomposition of the 8-dimensional patterns into n components through PCA then takes the following form:

$$\mathbf{p} = \mathbf{p}_0 + \sum_n \lambda_n \mathbf{e}_n \tag{4.8}$$

where $\mathbf{p}_0$ is the so-called mean pattern, and $\mathbf{e}$ and λ are the eigenpatterns and corresponding eigenvalues. Since the objective is to use the eigenvalues as parameters for a Hough transform and for subsequent imaging techniques, ideally the number of components should not exceed 2. Hence, it is necessary to check what level of

η Region	Expl. Variance (%)
$0.1 < \eta < 0.3$	99.5
$0.7 < \eta < 0.9$	99.5
$1.2 < \eta < 1.4$	99.9
$2.0 < \eta < 2.2$	99.7

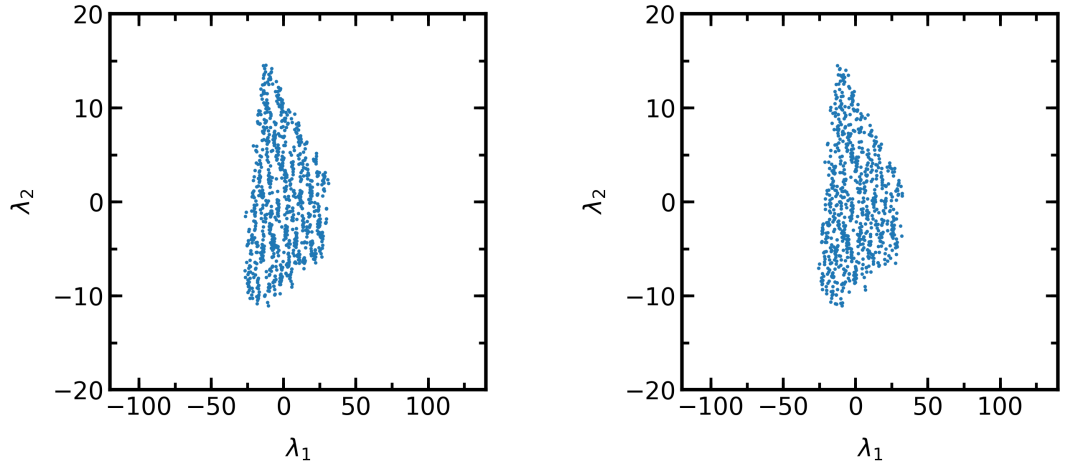
Table 4.4: **The explained variance from 2 PCA components - the figure quoted is the average explained variance across the 15 most populated sectors.**

information is lost by using only 2 components. Table 4.4 contains the explained variance from 2 components for each of the η regions, which shows that there is a negligible loss of variance in the superstrip positions. This result is significant in that it allows us to efficiently represent patterns in 2 dimensions and hence the following parametric representation can be used:

$$\mathbf{p} = \mathbf{p}_0 + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 \tag{4.9}$$

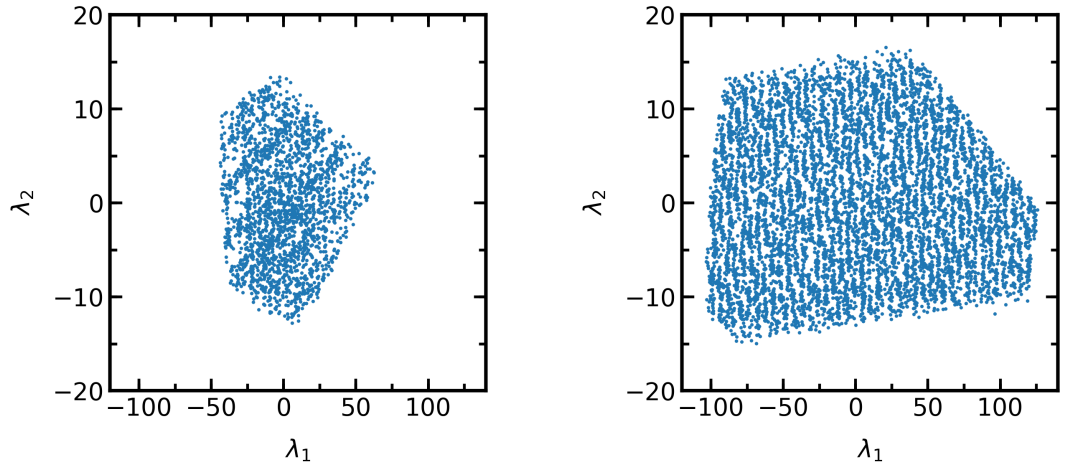
This equation will later serve as the basis for transforming superstrip measurements into a Hough space for tracking, and since the explained variance is very high in all η regions, a transformation to 2 components is unlikely to result in a significant loss of track finding efficiency. It should be noted that the quoted explained variance for 2 components is calculated using only the 15 most populated sectors, and so the approximation for less populated sectors may be less precise. However, we find that for each η region this metric does not fall below 90% for sectors with very few patterns, and since we have already seen that the vast majority of patterns are contained within relatively few sectors, the overall loss of information is still very small. This assumption is later justified by assessing the relative efficiency using the approximated patterns. Additionally, during the HT tracking we set a threshold on the minimum number of patterns in a sector and so the large number of sectors with only a few patterns are omitted. In Figure

4.5, the eigenvalues in the most populated sector in each region are plotted using the two principal components from Eq. 4.9. As the figure shows, the patterns in each sector are bounded by a convex polygon which defines the sector in the new 2-dimensional space. While we are only illustrating the most populated sector, it is clear to see that the patterns in the outer eta regions have a much wider range of values for λ_1, while the range for λ_2 is relatively constant.



(a) $\mathbf{0.1 < \eta < 0.3}$

(b) $\mathbf{0.7 < \eta < 0.9}$



(c) $\mathbf{1.2 < \eta < 1.4}$

(d) $\mathbf{2.0 < \eta < 2.2}$

Figure 4.5: **The distributions of patterns in the most populous sector in each η region are shown. For each, the col = 0 subsector is used.**

4.2.3 Interpretation of Eigenvalues

While Figure 4.5 does provide useful information about where tracks can exist within the image for each sector, it does not uncover the physical meaning of the eigenvalues (λ_1, λ_2). To make this clearer, consider four points at the extreme edges of the polygon in Figure 4.6. By using these (λ_1, λ_2) pairs, we can find the

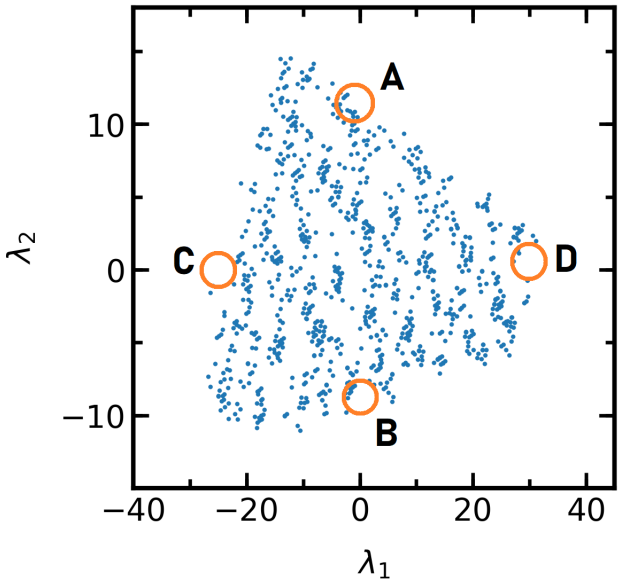
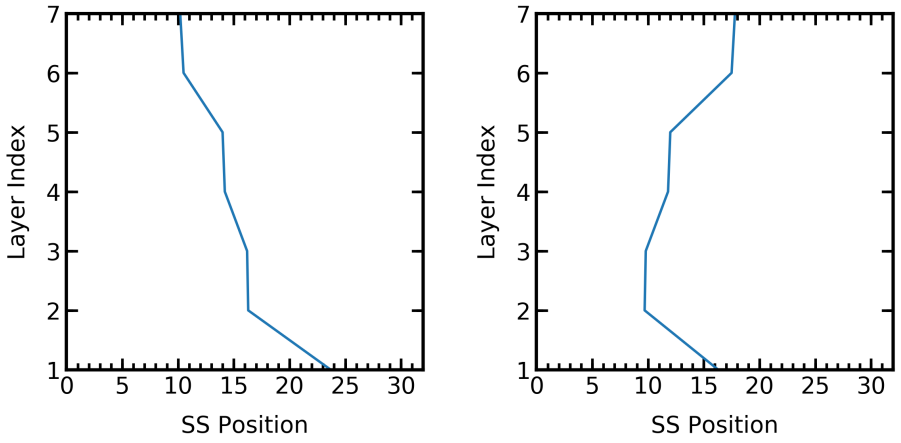


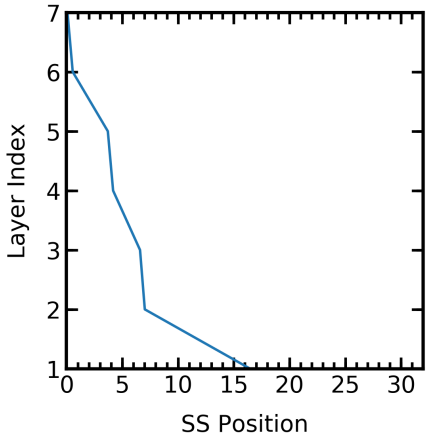
Figure 4.6: **Four extreme points of the polygon along $\lambda_1 = 0$ and $\lambda_2 = 0$ for the most popular sector in the $0.1 < \eta < 0.3$ region, labeled A-D. The polygon is the same as that in Figure 4.5a with the scale adjusted.**

projected superstrip positions across all of the layers using Eq. 4.9. Since each point represents an extreme pattern in both the positive and negative directions for (λ_1, λ_2), the resulting shape and orientation should clarify what the values of these parameters represents physically with respect to the track. These extreme points are approximated by choosing the central value of zero for each component and calculating the superstrip positions for minimum and maximum available values of the other component. Figure 4.7 contains visual representations of the resulting patterns. Note that the range of values corresponds to that listed in Table 4.3 for the $0.1 < \eta < 0.3$ region.

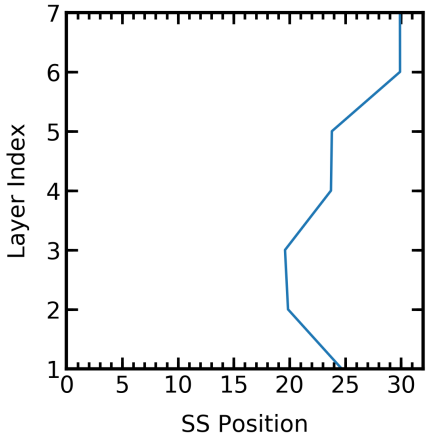


(a) **Point A: $\lambda_1 = 0, \quad \lambda_2 = -15$**

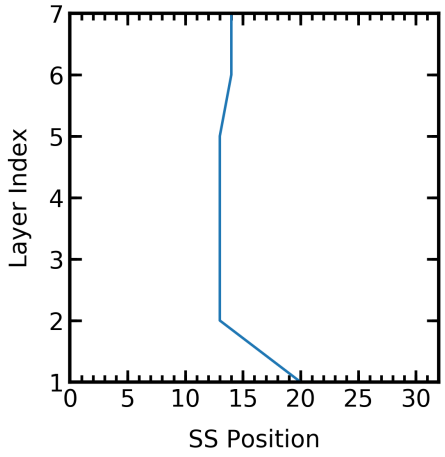
(b) **Point B: $\lambda_1 = 0, \quad \lambda_2 = 15$**



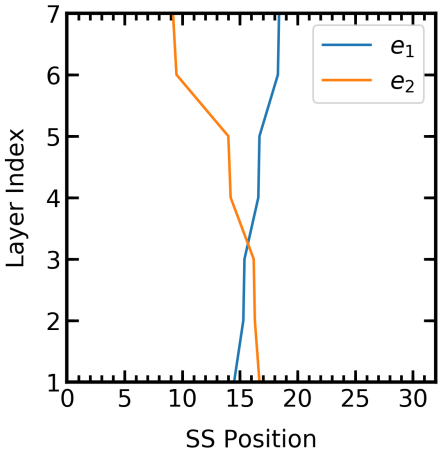
(c) **Point C: $\lambda_1 = -25, \quad \lambda_2 = 0$**



(d) **Point D: $\lambda_1 = 30, \quad \lambda_2 = 0$**



(e) **'Average' pattern: $\lambda_1 = 0, \quad \lambda_2 = 0$**



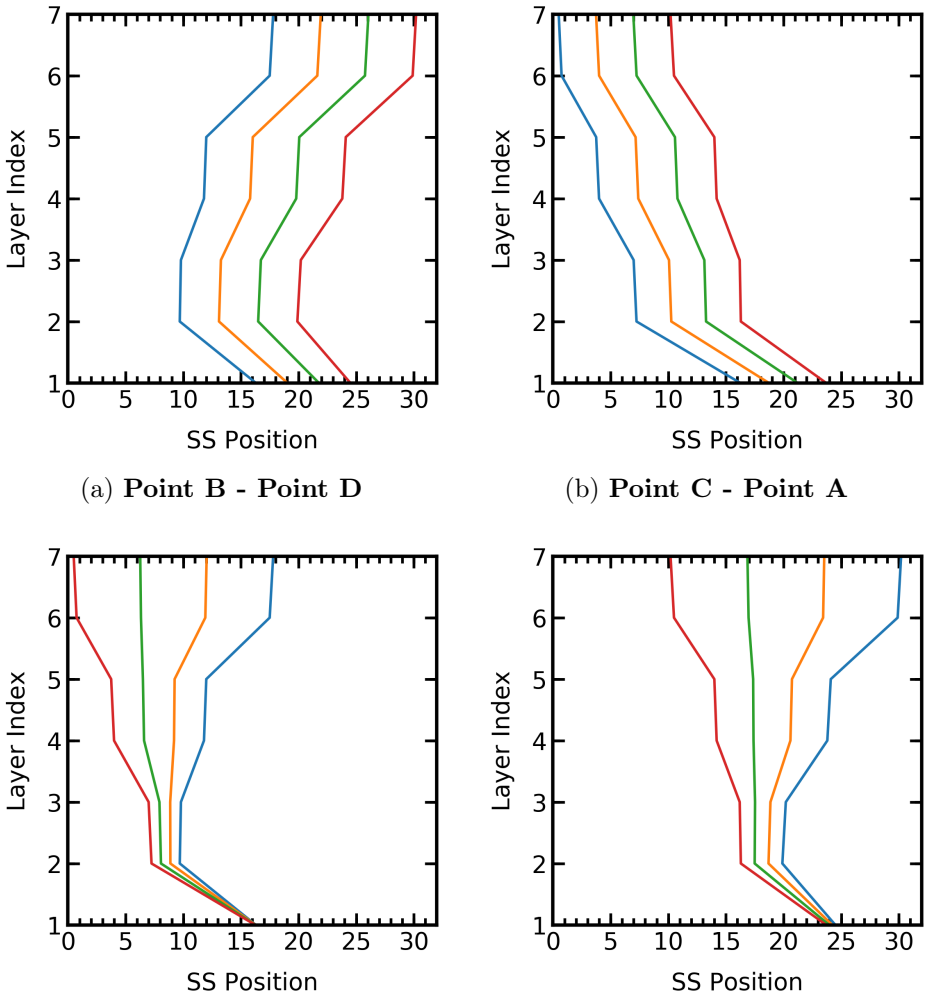
(f) **Eigenpatterns**

Figure 4.7: **The "extreme patterns" obtained by projecting the (λ_1, λ_2) pairs at the extreme edges of the polygon into the pattern space. These correspond to the image in 4.6, for the most popular sector in the $0.1 < \eta < 0.3$ region. Also shown is the 'average' pattern from $(\lambda_1 = 0, \lambda_2 = 0)$ and the eigenpatterns, $\mathbf{e}_1$ and $\mathbf{e}_2$.**

Also included in Figure 4.7 is the central 'average' pattern produced with $(\lambda_1 = 0, \lambda_2 = 0)$ and the shape of the eigenpatterns, $\mathbf{e_1}$ and $\mathbf{e_2}$. Note that the pixel layer has been omitted to provide a better image as the number of superstrip positions is different in this layer.

First, the average pattern is approximately a straight line through the centre with a 'tail' caused by the SS position in Layer 1. This tail is also present in each of the extreme patterns and hence is not a consequence of the values of (λ_1, λ_2). It can be seen that the pattern at point C is almost identical to the pattern at point A and is simply shifted to the right. Equally, the patterns at point B and D are approximately the same shape and differ only from the linear translation. This is shown more clearly in Figure 4.8a and Figure 4.8b, where the patterns from several points along the lines AD and CB have been plotted. As each of the lines is traversed, the pattern shifts to the right of the image. In Figure 4.8c and Figure 4.8d, lines in the opposite direction are traversed (AC and DB). As these images show, the position along these lines determine the direction in which the pattern 'fans' out as it propagates through the detector layers. This is analogous to a shift in the curvature of the pattern, although since the transformation occurs in the superstrip coordinate space defined by the sector, it is not the true ϕ_0 that is defined in the global ATLAS detector coordinate system.

In summary, within each sector the patterns may be described by two components, (λ_1, λ_2). This is possible due to some of the degrees of freedom present in a helical track are constrained due to the sector acceptance. The remaining degrees of freedom represent the linear, parallel translation of the average pattern within the sector and the "rotation" of the average pattern within the sector around a point near the first layer. However, these are not described in a one-to-one mapping by the eigenvalues (λ_1, λ_2). Instead, linear combinations of the eigenvalues, defined by the gradients of the lines traversed in Figure 4.8, parameterise the two physical properties that have been described.



(a) **Point B - Point D** (b) **Point C - Point A**

(c) **Point B - Point C** (d) **Point D - Point A**

Figure 4.8: **The patterns produced along the lines B-D, C-A, B-C, D-A. In each case, the blue pattern is the at the first point and the red at the last.**

4.2.4 Sector Linking & Incomplete Sectors

For the incomplete sectors that contain wildcards, the issue of how to perform PCA with missing data arises. [59] Approximately half of the sectors in the $0.1 < \eta < 0.3$ pattern bank have wildcards in one or two of the layers, accounting for around 36% of the patterns, and so a solution is required to include these sectors. A sector linking stage of the algorithm is devised in order to map a set of incomplete sectors to a corresponding complete sector which shares all the

available detector elements. Figure 4.9 presents a simple diagram of this process. As many as 36 incomplete sectors could, in principle, be linked to a single complete

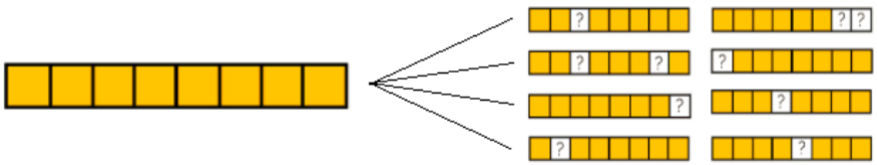


Figure 4.9: **Schematic of the sector linking stage. Each complete sector is linked to a set of incomplete sectors that share all detector element IDs, except for the layers containing wildcards.**

sector (8 single-wildcard sectors + 28 double-wildcard sectors), but in practice the most populated complete sectors will have 8-9 linked sectors. The precision of the PCA will be improved if the number of patterns in the sector is increased, so the patterns from incomplete sectors are concatenated with those from the corresponding complete sector. The layer(s) containing wildcard(s) is dropped, and the PCA compression can be performed on the reduced dimensionality sector, with the wildcards inserted into the mean pattern and eigenpatterns.

Additionally, there are a number of sectors that contain wildcards which can not be linked to any complete sector. These often have relatively low pattern population and as a result the precision of the sector parameters may not be as high as for the rest of the sectors. For these, there are no complete patterns to merge with, and so the wildcard layers are simply dropped and, as before, the PCA is performed on the lower dimensionality sector.

4.3 Compression and Efficiency

In this section we will review the results of the overall compression and relative efficiency in pattern matching of our benchmark pattern banks. These banks have undergone a two-stage compression with both a lossless factorisation method and an approximate PCA transformation. While we have already demonstrated the level of compression for the first stage and also that the explained variance through 2-component PCA compression is very high, we can now investigate the efficiency of the pattern matching algorithm using these approximations.

First, the level of compression of the pattern banks from PCA is not dependent on the value of η and can be calculated since the patterns and sectors have consistent data-types. The superstrip position and DC-bits are encoded in a `uint16` integer, and the detector element IDs are stored in `int32` integers. Since there are 8 layers and an additional unique `int32` pattern ID, each pattern originally required 416 bits to store. With a total of one million patterns in a single bank, which itself is one of 1256 0.2×0.2 regions in η-ϕ space, the total storage required for each is 52MB. After both stages of compression, each pattern requires only the pattern ID, sector index and (λ_1, λ_2) eigenvalues which are stored in `int8` integers. Additionally, there are storage requirements for the sector dictionary and PCA sector parameters. Per sector, the dictionary needs 9 `int32` for the detector element IDs as well as the sector index. To store the sector parameters from the PCA approximation, there are 2 eigenpatterns and a mean pattern. These comprise of 8 `int8` integers and the pattern ID, as well as the column position for the subsector. Combining these together, each sector requires 912 bits per sector. The total number of bits and overall storage needed for the pattern banks is then summarised in Table 4.5. The storage required for each pattern is greatly reduced by over a factor of 5, but additional requirements are picked up due to the sector dictionary and storage of PCA sector parameters. The metrics in this table refer

	Before Compression	After Compression
Bits per Pattern	416	80
Total Pattern Storage (MB)	52	10
Bits per Sector	-	912
Total Sector Storage (MB)	-	1.40
Total Storage (MB)	**52**	**11.40**

Table 4.5: **Summary of storage requirements before and after compression pipeline for $0.1 < \eta < 0.3$ pattern bank. The results are the same for other η regions except for the number of sectors and hence total storage per sector, however this makes up only a small fraction of the total storage.**

to the $0.1 < \eta < 0.3$ bank and the total storage is found using the actual outputs, since it should be mentioned that the sector storage is variable. The number of sectors will vary for each bank and within each sector, it may be the case that there were only patterns with a single column measurement and hence only one subsector of PCA sector parameters is stored. Additionally, for the $2.0 < \eta < 2.2$ region, the number of column positions is 4 and so there may be many more subsectors stored. However, the total storage will not vary considerably given that the bulk of the storage requirements come from the patterns themselves.

Finally, we must assess the pattern matching efficiency using the approximated patterns. To do this, the patterns are transformed into their original pattern space using Eq. 4.8. The resulting restored patterns will not be identical to the original bank due to the loss of information in the PCA transformation. Using the relative efficiency defined by:

$$\epsilon_{rel} = \frac{\text{Efficiency using approximate patterns}}{\text{Efficiency using exact patterns}} \tag{4.10}$$

we obtain the results that are summarised in Table 4.6. As shown, the original pattern matching efficiency and cumulative explained variance for all η regions is very high. We find that for all regions, the relative efficiency is still very high and

η Region	Expl. Variance (%)	ϵ_{exact}	ϵ_{approx}	ϵ_{rel}
$0.1 < \eta < 0.3$	99.5	0.99	0.99	**1.0**
$0.7 < \eta < 0.9$	99.5	0.99	0.98	**0.99**
$1.2 < \eta < 1.4$	99.9	0.99	0.99	**1.0**
$2.0 < \eta < 2.2$	99.7	0.99	0.99	**1.0**

Table 4.6: **Summary of cumulative explained variance and relative pattern matching efficiency for each η region. [60]**

the drop in performance is almost negligible. Since the relative efficiency is 100%, for the following chapters we use the $0.1 < \eta < 0.3$ region to demonstrate how the compressed pattern representation can be used in the HT-based tracking.

The PCA compression results from this section are reported in [60]. Additionally, the plan to use the PCA compressed space for the Hough Transform tracking algorithm is introduced, with some follow-up discussion on its feasibility on GPUs.

Chapter 5

Hough Transform Based Tracking

In this chapter, we will describe the details of a tracking algorithm based on the aforementioned PCA Hough space. First, the transforming of superstrip positions is demonstrated and the initial images in this accumulator space are produced. Through an iterative explaining away procedure, the presence of real and spurious peaks can then be distinguished and the noise in the image removed. We also describe the additional dilation filters used to assist the pipeline's ability to produce sufficiently prominent peaks for real tracks before outlining the different methods of blob detection that can locate these peaks. Finally, we outline a track fitting stage that selects only the highest quality track candidates and removes many instances of fake tracks.

5.1 Transforming Superstrips

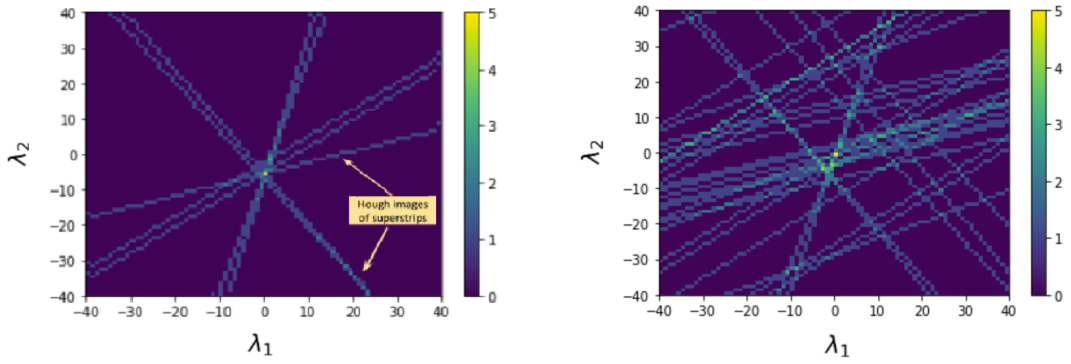
In section 4.2, the compressed PCA space was defined and used to reduce the storage required for the pattern banks. In this way, a 2-dimensional (λ_1, λ_2) space was defined in which patterns, and therefore tracks, can be characterised by a single point. The patterns are decomposed by a linear combination of the mean pattern, $\mathbf{p_0}$, and the two eigenpatterns, $(\mathbf{p_1}, \mathbf{p_2})$,

$$\mathbf{p}(\lambda_1, \lambda_2) = \mathbf{p_0} + \lambda_1 \mathbf{p_1} + \lambda_2 \mathbf{p_2} \tag{5.1}$$

and hence in the PCA space, the pattern is defined by the track parameters (λ_1, λ_2). Given the j^{th} hit in the i^{th} layer, h_{ij}, the PCA model defines a line in this parameter space by,

$$\lambda_2 = -\frac{p_{1i}}{p_{2i}}\lambda_1 + \frac{1}{p_{2i}}(h_{ij} - p_{0i}) \tag{5.2}$$

This situation is then analogous to that of the classical Hough Transform where we have straight lines drawn in the Hough space corresponding to points in the so-called "pattern space". With this general framework, we can perform a sector-



(a) **In an event with only muon hits.** (b) **In an event with pile-up = 200.**

Figure 5.1: **Hough Transform on superstrip hits in the PCA space.**

based Hough Transform in the PCA parameter space to produce images like those in Figure 5.1.

By transforming the superstrip information into the PCA space, the process of pattern matching for track finding is replaced by a peak detection task in Hough images. In principle, the pattern matching criteria of 7/8 hits found (discussed in Section 2.5.2) can be replaced by the detection of a peak in the Hough image with magnitude of at least 7, for example, indicating that at least 7 different hits cast their votes for this peak. The approximate location of the peak provides values for (λ_1, λ_2) which, using the following equation:

$$h_{ij} = p_{0i} + \lambda_1 p_{1i} + \lambda_2 p_{2i} \tag{5.3}$$

can be used to predict the hit position in the i^{th} layer and reconstruct tracks from the image. While the position of such a peak may be intuitively obvious when there are only hits from a single muon, such as in Figure 5.1a, the situation becomes more difficult when the hit density of the event is much higher. Figure 5.1b shows the Hough image for the same muon in an event with $\mu = 200$, where in each particular sector the number of hits is usually 4-5 times greater. The greater population of hits in the detector translate to a dense field of interconnecting lines in the Hough image, for which we must find the most prominent intersections. Furthermore, there needs to be a way of distinguishing true peaks resulting from muon tracks, or, indeed, pile-up tracks, from artefacts caused by the hit density. It is a known drawback of the classical Hough transform that false peaks can arise, in which votes are cast to the same bin from different parts of the image that do not correspond to the same instance of a shape. Additionally, due to the inherently quantised nature of the transform, all of the pixels from a particular shape-instance may not vote for the same exact bin in the Hough space. The following section outlines some solutions to these problems.

5.2 Peak Formation - Explaining Away

In image processing, it is a common problem to recognise certain shapes and objects within the image. Some may be mathematically identified, such as edge detection, by using the Gaussian derivative to find significant local changes in the image's intensity [61]. Another technique used for image patch recognition is known as template matching [62], in which the object being detected is defined as a set of pixels with particular intensities, and the image itself is scanned for local occurrences of such a template. However, there are many problems associated with the metrics that are used to assess the similarity of the image structures to the templates. Pixel-wise comparison does not allow for robust performance when there are occlusions and deformations in the image and using a low global threshold on the similarity can produce a high number of false positives. M.W. Spratling [63] showed that a pixel-level analysis can be used where the templates compete with each other to be matched to the image structure. This concept of competition between matches is precisely what explaining away is based on. Each bin in the accumulator space is competing for the votes cast by the superstrip hits, and through an iterative process the noise in the Hough image can be whittled down so that only the bins that best explain the presence of these hits remain active.

Figure 5.2 shows how a single iteration of how the votes are updated in the case of the Hough tracking algorithm. Starting with an image such as that in 5.1b, the hits belonging to each layer are processed independently. A threshold, t_{line}, is applied to the maximum value of each line, such that lines with insufficiently high values are removed (i.e., all the values in the line set to zero). For the remaining lines, the values are normalised with respect to the maximum value, before a second threshold, t_{pixel}, is applied to the individual pixels along the line, setting any pixel below the threshold to zero. The combined effect of these operations does

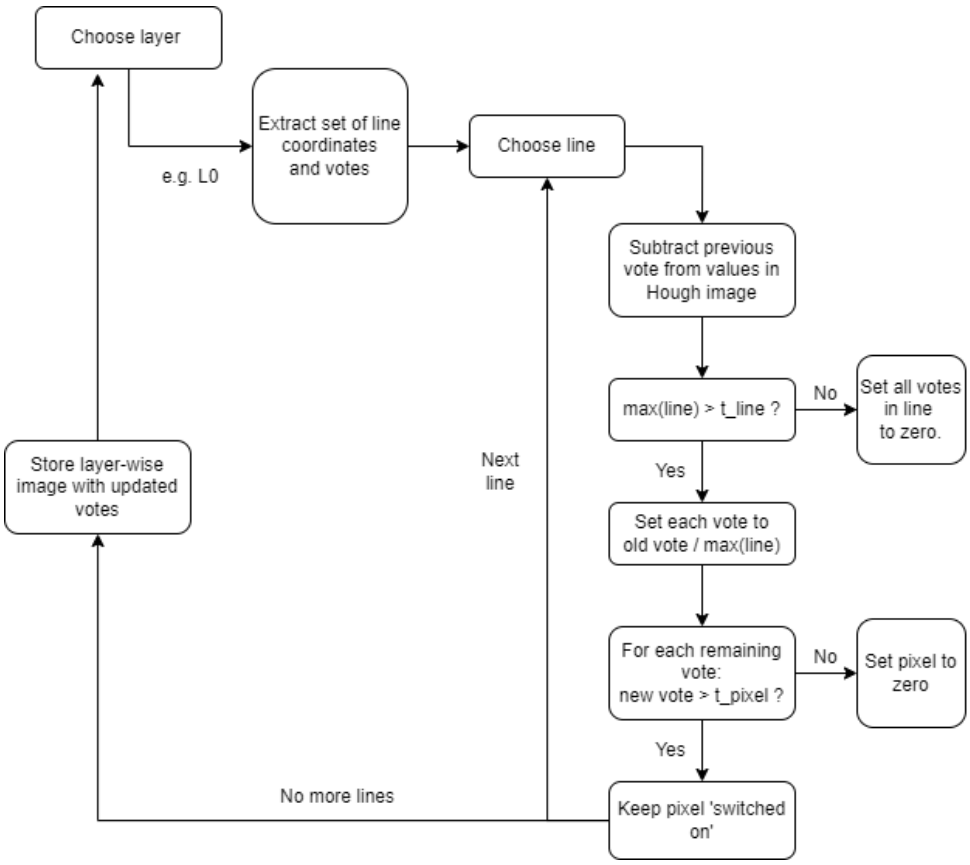
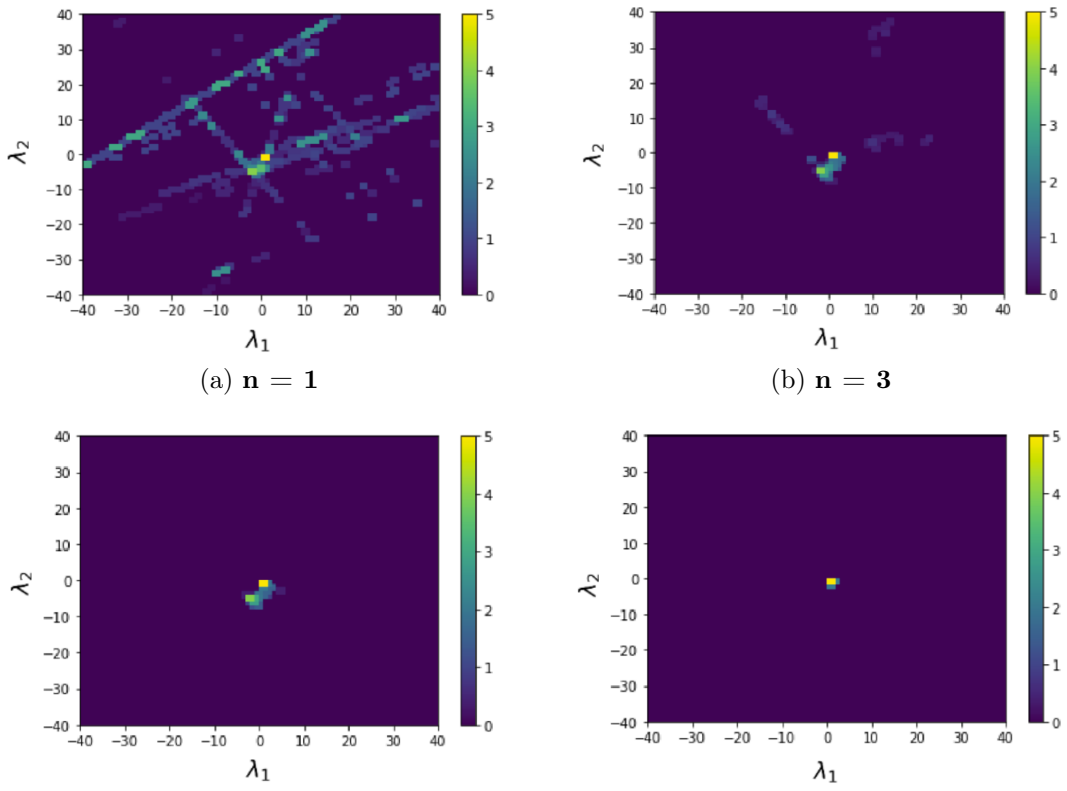


Figure 5.2: **Flow chart illustrating the iterative explaining away procedure. The process is repeated for each layer to build a layer-wise image with updated votes (i.e. number of entries in a bin). The layer-wise images are then summed to produce a new Hough image for the next iteration.** t_{line} **and** t_{pixel} **are hyperparameter thresholds that are used to "switch off" lines or individual pixels that are considered noise.**

two things. The thresholds suppress any part of the image which is insufficiently likely to contain a peak, while the normalisation enhances the regions where a peak is most likely. Note also that due to the normalisation, the votes cast by each hit can be non-integers, but will still be between 0 and 1. The updated votes from each hit are used to produce a new image representing the next iteration of the explaining away. In Figure 5.3, an illustration of the explaining away method on the same pile-up Hough image shown in Figure 5.1 is presented. It can be seen that after $i = 3$ iterations, most of the noise in the image has disappeared and after 7 iterations, the only votes that remain are in agreement on one clearly defined

region. It should be noted that for illustrative purposes some of the thresholds were reduced to slow down the peak condensation in these images and that the convergence can be reached in fewer than 7 iterations. When deciding the number of iterations and values of t_{line} and t_{pixel}, the aim is to set the thresholds sufficiently high so that the explaining away occurs quickly and the number of iterations can be minimised, but not so high as to suppress the formation of real muon peaks. This is achieved qualitatively by checking that, for some test images like those in Figure 5.3, a clear peak is formed that corresponds to a muon track. In practice, a stricter algorithm requirement could be used - for example, requiring a certain number of hits to have been consolidated in a particular region of the image. However, it is sufficient for this work that a fixed number of iterations is used, so long as it is high enough to produce strong peaks.



(a) **n = 1** (b) **n = 3**

(c) **n = 5** (d) **n = 7**

Figure 5.3: **The Hough-transformed superstrips after n iteration of explaining away.**

5.3 Dilation & Residual Line Widths

In image processing, morphological filters are a set of operations that can alter the features of an image based on their shapes. The operations are non-linear and can be used for a variety of purposes to produce an image of the same size with some of the features distorted in some way. In any morphological filtering technique, a sliding window known as the structuring element is applied to the image which defines a neighbourhood around each pixel. Two common morphological operations apply minimum and maximum operations within the structuring element and are known as *erosion* and *dilation*, respectively.

Figure 5.4 presents an example of these operations. While an erosion filter suppresses bright regions in the image, a dilation filter enhances them. In these examples, a circular structuring element (or window) is used to perform the filtering. In the case of the superstrip Hough images, the principle of dilating bright

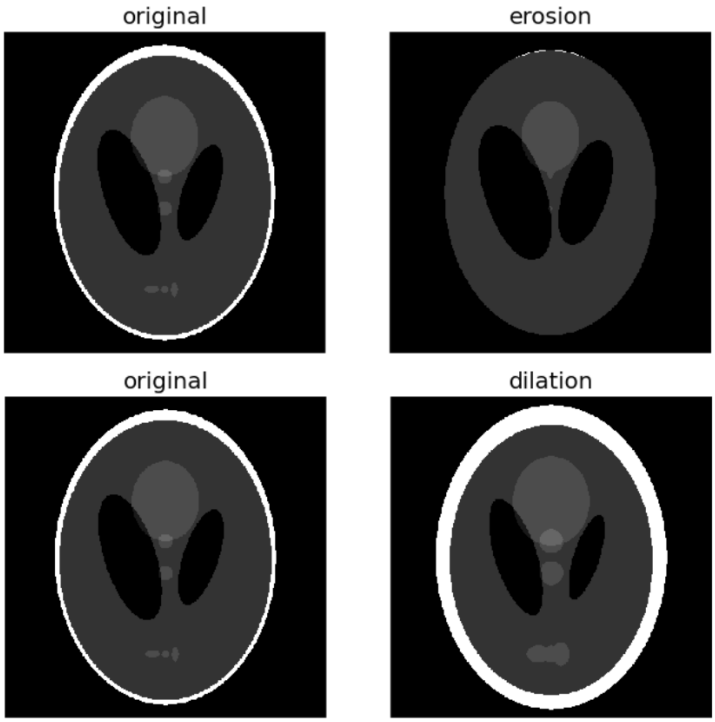


Figure 5.4: **The effects of dilation and erosion filters on an example image. [64]**

spots can be used in a couple of ways. One is the traditional dilation filter that has just been described and it can be applied at the end of each iteration of the explaining away procedure. In order to detect a potential track, a pixel would need to be found containing votes from 7 or 8 superstrips. This is unlikely to happen for many real tracks without some kind of dilation since the superstrip positions, as well as the eigenvectors used in the transform, are approximated through PCA. The result is likely an intersection of a group of superstrips through a broader region of the Hough space, as opposed to a single pixel.

Another method for "dilating" the bright spots in the image is more statistically motivated and based specifically on the PCA approximation. It should first be mentioned that this is not, strictly speaking, a form of dilation in the morphological sense, but the effects on the ability to condense peaks in the Hough space is similar. It was outlined in Section 2.5.2 that the 2-dimensional PCA space used for Hough-tracking is defined on a sector-by-sector basis. There are unique eigenpatterns and mean patterns that define the space and each of these have an associated uncertainty per layer. The uncertainty is modelled by compressing the patterns into the PCA space and then transforming back into the pattern space

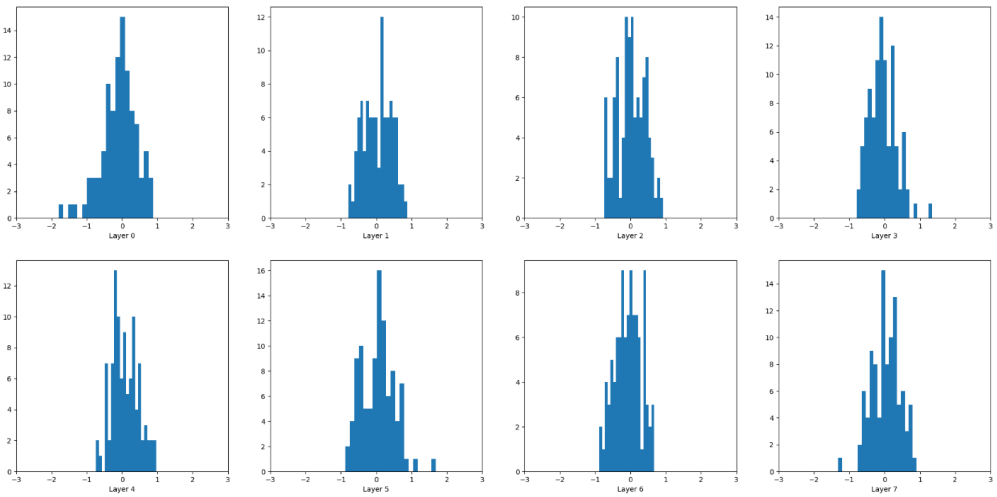


Figure 5.5: **Residual plots illustrating the uncertainty in superstrip measurements due to PCA for each layer in sector.**

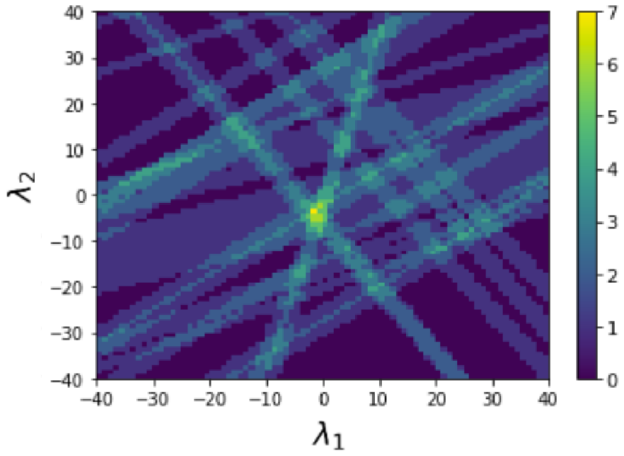
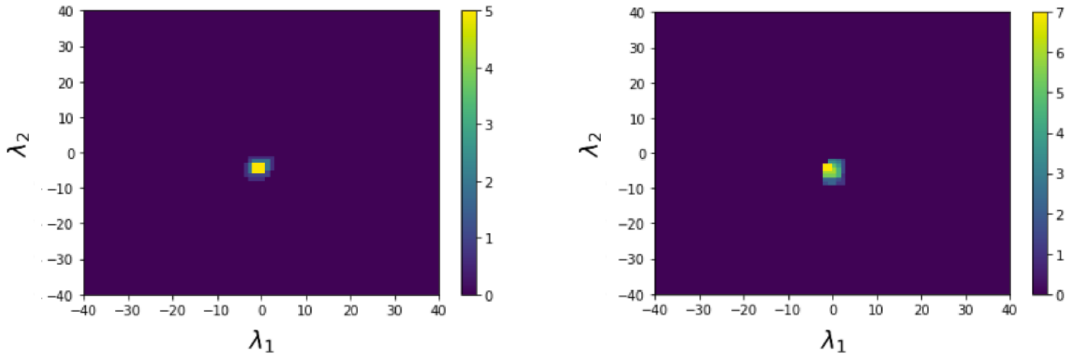


Figure 5.6: **Transformed superstrips with PCA approximation uncertainty factored in.**

and plotting the difference in the superstrip measurements in a series of residual plots. Such plots are shown in Figure 5.5. The widths of each residual plot can be used to determine how many pixels wide the superstrip lines should be drawn. With the uncertainty included, the intersection becomes much clearer to see; an example of this is shown in Figure 5.6. Compared with Figure 5.1b, the lines are far wider and there are many more dense regions in the image. A potential downside of this is that more prominent intersections representing "fake tracks"



(a) **With traditional dilation filter.**

(b) **With extra PCA residual line widths included.**

Figure 5.7: **Single muon peak detected in pile-up event after 5 iterations. The distinction between the two is in the values on the colour-bar - additional votes are captured in the peak in (b) compared with (a).**

may exist and remain in the image after the iterative explaining away. The balance between detecting muons and minimising the detection of fake tracks will be discussed throughout Chapter 6, but as Figure 5.7 shows, in this particular instance only the muon track is found. If such an image were produced with neither the residual line widths nor the traditional dilation included, and assuming the number of iterations and thresholds were kept constant, the result would be an empty image. This is because the initial density of hits is too low in any particular region at the start of the iterations and each line ends up being switched off. The inclusion of dilation to the algorithm sees a clear muon peak emerge, albeit with a maximum value of only 5.

However, as the colour-bar shows in Figure 5.7, once the residual line widths are also accounted for, the peak contains votes from at least 7 layers, making it potentially a viable track candidate. Additionally, no other spurious peaks are detected at the same time. Should there be additional tracks in a given sector, we would need the ability to detect any number of peaks in the image. This is discussed in the following section.

5.4 Blob Detection

As with the process of edge detection, there are many different methods for detecting objects known as "blobs". Generally a blob can be defined as a collection of pixels forming a large object that is distinguishable from the image's background. Several techniques exist for identifying such features in images, some of which are outlined in Figure 5.8, and each have their own benefits and drawbacks making them appropriate for different contexts. In each case, a filter (also referred to as a convolution or kernel) is applied to the image to identify regions of the image which differ from its surroundings. Here, we will describe the three blob detection techniques shown in Figure 5.8, each based on different derivative filters (i.e. filters derived from derivative functions): The Laplacian of Gaussian (LoG), Difference of Gaussian (DoG) and the Determinante of Hessian (DoH).

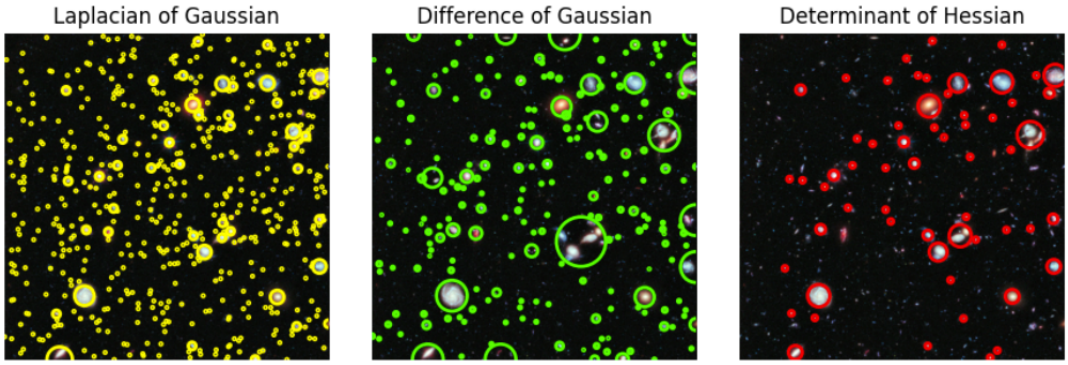


Figure 5.8: **Example illustrations of blob detection performed using Laplacian of Gaussian, Difference of Gaussian and Determinant of Hessian filters [65].**

Lapacian of Gaussian Consider the two-dimensional Gaussian distribution:

$$f(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x^2 + y^2)}{2\sigma^2}\right). \tag{5.4}$$

By applying the Laplacian operator,

$$L(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{5.5}$$

we arrive at the Laplacian of Gaussian filter, which has the form:

$$\mathrm{LoG}(x, y, \sigma) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right] \exp\left(-\frac{(x^2 + y^2)}{2\sigma^2}\right). \tag{5.6}$$

Figure 5.9 shows the shape of this filter. The LoG filter calculates the second derivative so in flat regions of the image (i.e. with constant intensity) the response is zero. However, it is sensitive to regions with changing intensity, such as in the vicinity of a blob. In order to use this filter for blob detection, the relative scale of the blobs is required which is determined by σ, the standard deviation of the Gaussian kernel. The filter is able to detect blobs with radius roughly equal to $\sqrt{2}\sigma$, so in order to detect blobs of arbitrary size a range of values for σ are scanned. One drawback to this method is that detecting larger blobs in the image

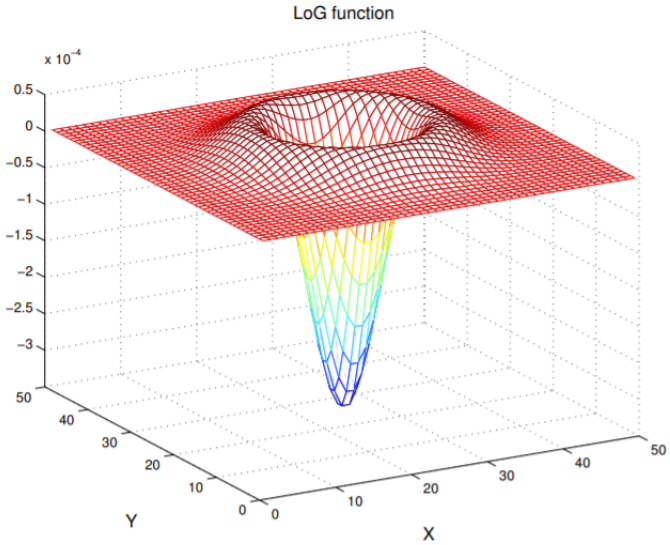


Figure 5.9: **Shape of the Laplacian of Gaussian (LoG) filter. [66]**

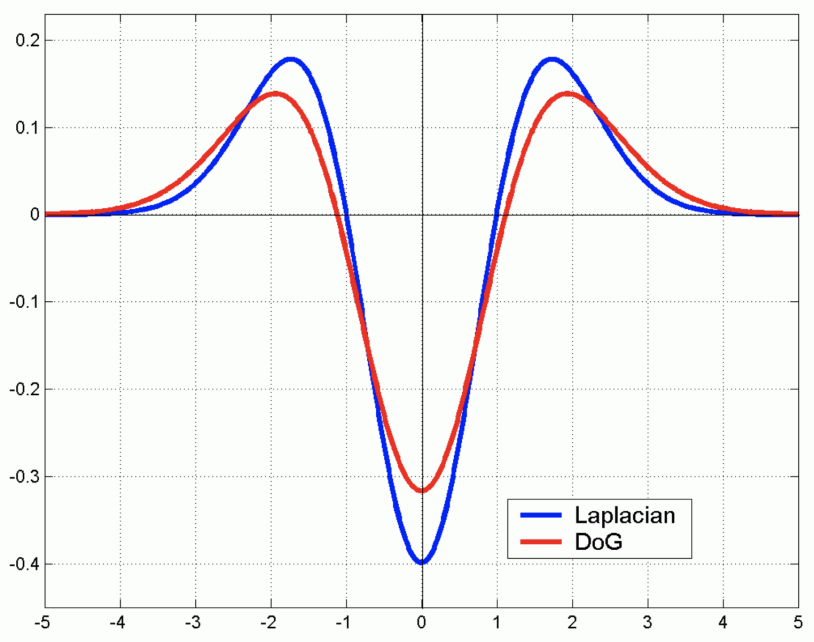


Figure 5.10: **Difference of Gaussian (DoG) v Laplacian of Gaussian (LoG) functions. The LoG function uses standard deviation σ, and the successive Gaussians in the DoG function are 2σ and σ. [67]**

is slower due to the larger Gaussian kernel that is required.

Difference of Gaussian The Difference of Gaussian (DoG) method is an approximation to the LoG method and can be executed much more quickly. This filter involves taking the difference between two Gaussian images with different values of σ. Unlike the LoG filter, which essentially applies the Gaussian filter followed by Laplacian filter, the DoG only applies the Gaussian filter at each value of σ and subtracts the less blurred image. A comparison of the two functions is shown in Figure 5.10, which illustrates that the DoG is an approximation of the LoG filter. As the same Gaussian kernel is used, the method is still time-intensive for detecting larger blobs.

Determinant of Hessian The Hessian matrix is defined as a square matrix of second-order partial derivatives, and for a 2D scalar function is given by:

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

The eigenvectors (e_1, e_2) of this matrix then correspond to the directions of highest and lowest curvature of the function with magnitudes given by the eigenvalues (α_1, α_2), respectively. For a Gaussian function, such as the one shown in Figure 5.11, at the peak of the function both of the eigenvalues would be at a maximum and due to the symmetry of the function, $|\alpha_1| = |\alpha_2|$. Elsewhere in the domain, where the function is at a zero-value, there is no curvature both eigenvalues are zero. Hence, the determinant of the Hessian matrix calculated by:

$$\det(H) = \alpha_1 \cdot \alpha_2 \tag{5.7}$$

can represent the combined level of curvature in both directions for a 2D function (or image). This means that the Determinant of Hessian (DoH) filter can detect blobs of both circular and elliptical shape. As with the other techniques for blob detection, a scale-space representation using σ needs to be built to detect blobs

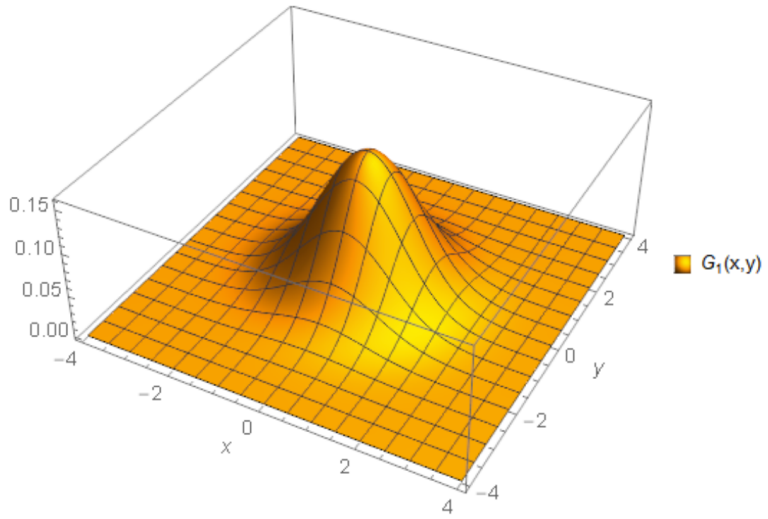


Figure 5.11: **Example of a 2D Gaussian function.**

of different sizes, although this method fails to detect very small blobs effectively. However, unlike the previous methods, the DoH does not take longer for larger blobs since there is no convolution or filter applied to the image based on the value of σ.

Example of blob detection on HT superstrip images Since the DoG filter is quick and simple to apply, Figure 5.12 shows an example of some track candidates in a Hough image and how the DoG blob detection method can be used. Two track candidates are clearly visible in the image and the coordinates of the maximum pixels can be extracted for an estimate of the eigenvalues of the track candidates. For the ease of application and given that the DoG filter is able to easily detect the peaks in these Hough images, it has been chosen as the baseline blob detection method for the studies and results presented in this thesis.

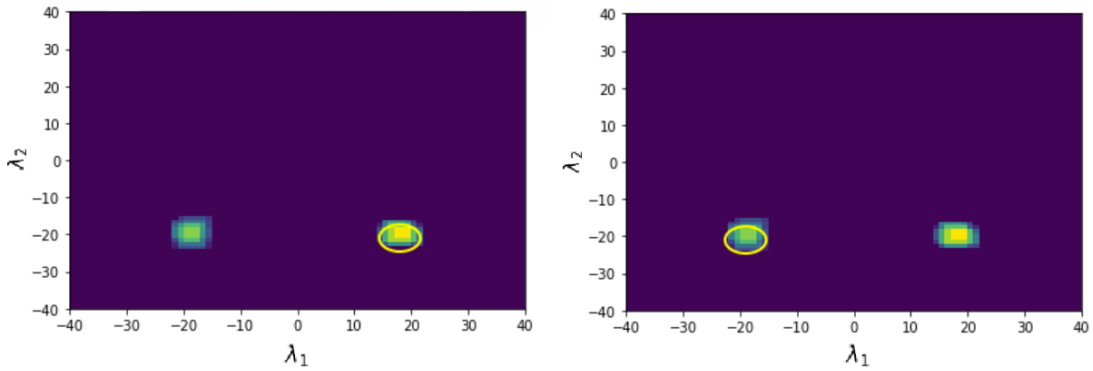


Figure 5.12: **Two track candidates detected using Difference of Gaussian filter.**

5.5 Track Fitting

The final stage of the tracking pipeline takes the initial estimates of the PCA track parameters, (λ_1, λ_2), produced from the peak detection and computes the best possible estimate through some kind of fit. It was outlined in Section 2.3.2 that, in ATLAS track fitting, a global χ^2 fit is often used which is defined by:

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{r_i^2}{\sigma_i} \right)^2 \tag{5.8}$$

where r_i are the residuals in each layer (the difference between the predicted and measured superstrip position) and σ_i are their uncertainties, determined by the width of the residual plots in Figure 5.5. In conventional tracking algorithms, this function is minimised to produce the best possible estimates of the global track parameters defined in Section 2.3.2. In the case of the HT-based tracking, the same function is used to precisely estimate (λ_1, λ_2) and produce a score for the quality of the track. A cut on the value of χ^2 can then be applied to improve the fake track rate for each sector. In comparison with the HTT pattern matching, this additional track fitting step is included within the pattern recognition stage. That is to say, the outputs of the pattern matching algorithm are comparable with those from the HT tracking algorithm after this track fitting stage. In both cases, the resulting tracks are passed onto the global track fitting to obtain the best estimates of the global ATLAS track parameters.

Chapter 6

Implementation and Performance

In this chapter, we will provide the full details of the implementation and the performance evaluation of our HT-based tracking algorithm. The description and preparation of the event data used is covered before detailing the production of the Hough images themselves. The latter will include the line drawing algorithm, handling of hit position uncertainties and the full mathematical formulation for the iterative explaining away. The propagation of the MC truth information through the use of hit-maps is also explained. Next, we discuss the extraction of tracks from the peaks in Hough images cleaned by explaining away, and subsequent track fitting to select the highest quality tracks. Finally, we report the preliminary results of the algorithm performance, optimisation efforts, and comparisons with existing pattern matching frameworks.

6.1 Data Preparation

6.1.1 Event Data

To assess the performance of the HT tracking algorithm, we use a sample of 600 Monte Carlo (MC) simulated events with $<\mu> = 200$. In each event, there is one true muon with $p_T > 4$ GeV for which MC truth information is available. 200 of these events are used to optimise the algorithm's hyperparameters (such as the number of iterations, thresholds for explaining away, and window size for determining peaks) and a further 400 events are used as a test set to validate the results. As has been outlined in Section 5, the Hough space used for tracking corresponds to the compressed 2-dimensional PCA space produced for each sector.

6.1.2 Sector Scanning

The proposed Hough-based tracking is performed on a sector-by-sector basis, due to the properties of the PCA space being sector-specific. Hence, to perform tracking in this space for a given event, we first need to find the sectors that contain a sufficient number of hits to build track candidates from. First, we set a minimum threshold on the number of patterns in the sector used to generate the PCA parameters. Figure 6.1 helps to illustrate why this is necessary, when compared with Figure 5.5. While the residual plots in Figure 5.5 are the result of using over 1500 patterns for the PCA training, those in 6.1 used only 5 patterns. With so few patterns, the precision of the resulting transforms is much lower than for a high-population sector and the uncertainty of the superstrip positions for some of the layers lie outside the range allowed by DC-bits. Hence, we only search for muons in sectors with sufficient pattern population. To ensure good precision of PCA parameters, the threshold on the number of patterns in a sector, p_{min}, is initially set to 30.

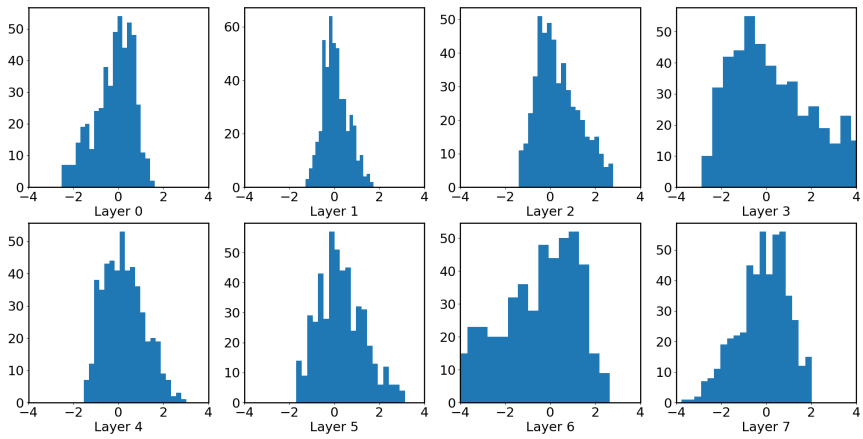


Figure 6.1: **Residual plots illustrating the uncertainty in superstrip measurements due to PCA for each layer within a sector. In this case, only 5 patterns are used to train the PCA model and the residuals are calculated from a further set of 500 patterns in the same sector.**

Next, we record every detector element in which any superstrip has fired and its corresponding layer index. Any sector is then added to a list of "active sectors" if at least 6 detector elements (out of 8) are found in the record of active detector elements. The reason for the 6/8 criteria is the presence of double-wildcard sectors, which only have 6 known detector elements. For each active sector, the proceeding steps are dependent on how many wildcards there are, and whether or not one of them is in the pixel layer. Note that from here on out, the layers will be referred to by their position in the pattern bank, i.e., L0 = pixel layer, L7 = last strip layer.

As explained in 4.2.2, the 2-dimensional measurement in the pixel layer (L0) necessitates that the sectors be split into two subsectors - one for each of the column measurements in the pixel. Due to this, there are three possible outputs when generating Hough images that are determined by the location of the wildcards, as illustrated in Figure 6.2. First, if there is indeed a wildcard in the pixel layer the situation is the simplest. There is only one subsector so the mean pattern and eigenpatterns corresponding to this sector can be selected and used to transform

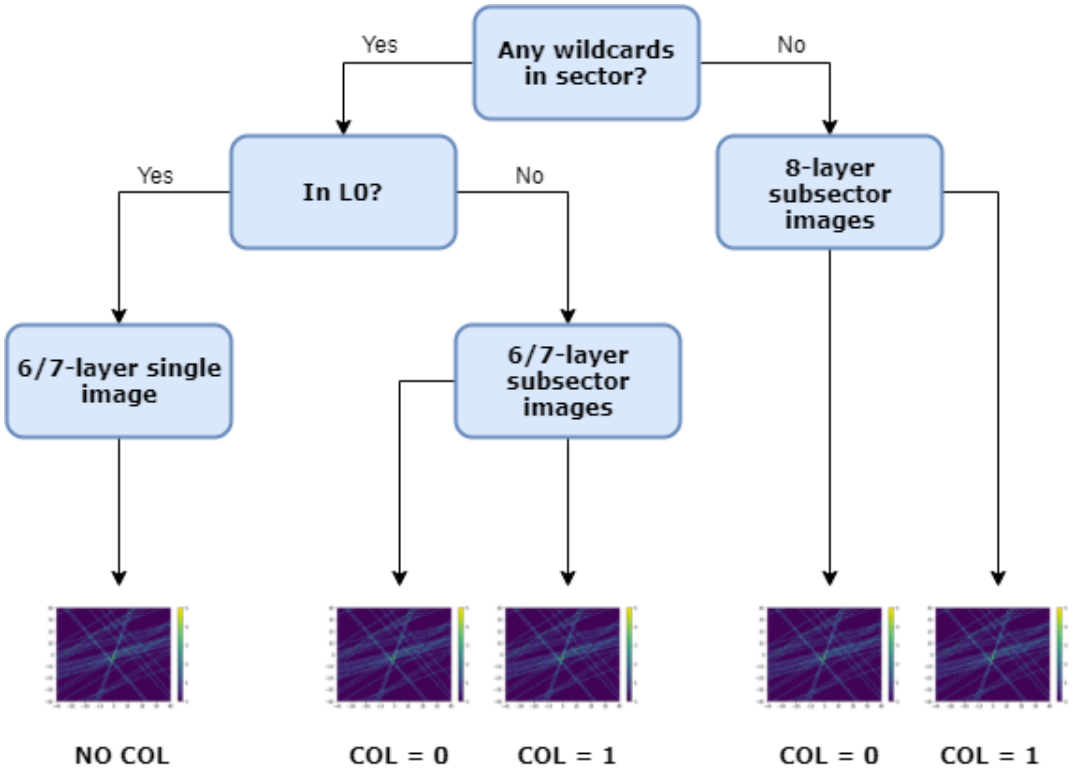


Figure 6.2: **Flowchart demonstrating the algorithmic pathways for handling wildcard sectors. The content of each sub-image is purely illustrative and is meant only to show that a separate Hough image is generated for each scenario, based on the column value (or lack thereof) in the pixel layer.**

all hits present in the sector, resulting in a single Hough image. The number of layers contributing to the image will be $8 - n_{\text{wildcards}}$. If there is no wildcard in L0, then the column measurement is active again and there may be two subsectors to search in. Similarly, if there are wildcards in other layers, then the layer indices must be accounted for to make sure no hits in these layers are projected into the image. Then the subsequent steps are essentially the same whether there are wildcards or not. Hits from all layers beyond L0 are present in both subsector images, but when considering the pixel layer hits only those with the correct column measurement are projected into the corresponding subsector image. The output for each pathway ensures that the correct (and most precise) PCA parameters are used to transform each superstrip and to ensure only the correct hits are

present in each image. To summarise, for each active sector with hits in at least 6 different layers, we generate one or two images, depending on whether or not there is a wildcard in the pixel layer.

6.2 Hough Imaging

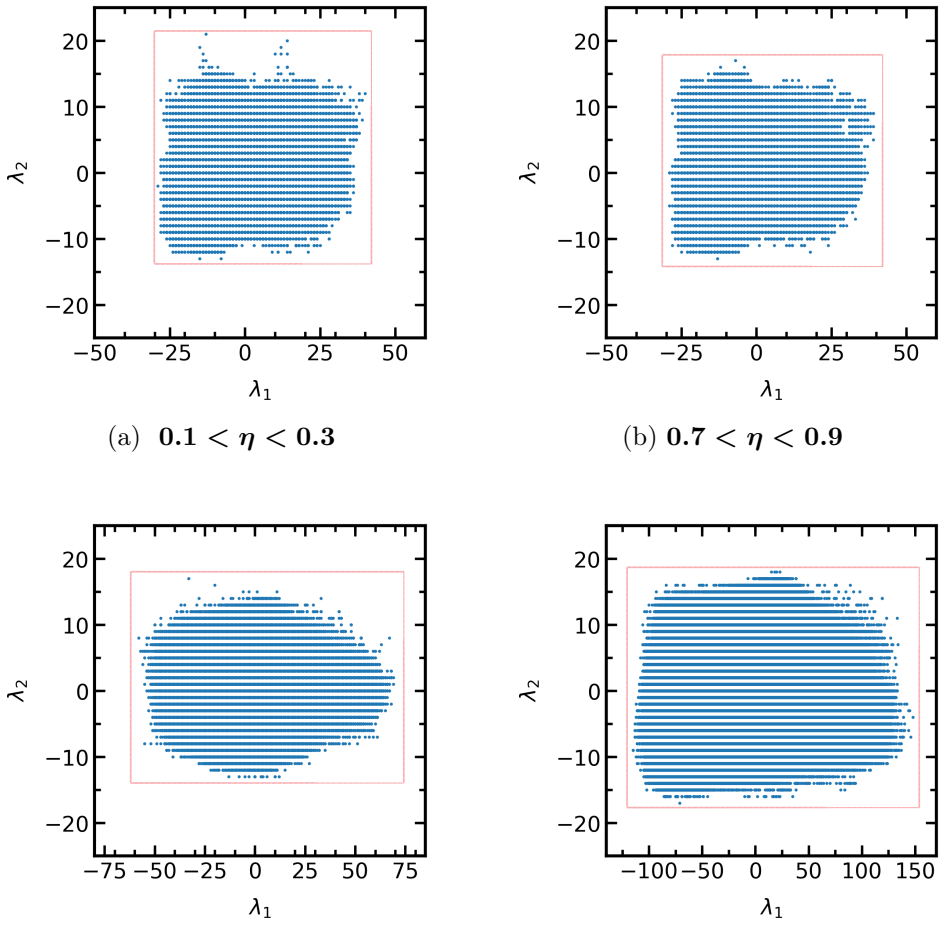
This section will cover the part of the pipeline responsible for creating and handling the Hough images. The details of the initial transformations, error propagation, iterative image processing and propagation of truth information through the algorithm will be covered.

6.2.1 Bresenham Line Drawing

As outlined in the previous section, the mean pattern and eigenpatterns are selected based on the details of the sector's wildcards and can be used to perform the Hough Transform on the superstrip positions, providing us with a pair of eigenvalues (λ_1, λ_2). Since the Hough space is discrete (with integer spacing, for ease of implementation), the Bresenham line algorithm [68] (or some modification thereof) is required to determine exactly which pixels a given line will pass through. This can be achieved with ease using the Python Imaging Library (PIL) [69] ImDraw package. The line parameters are known from the Hough transform calculation, from which the coordinates at the edges of the image can be calculated and used to draw the lines. The boundaries of the image are chosen using the typical distributions of patterns in the η-ϕ region.

	$\boldsymbol{\lambda_1}$ **range**	$\boldsymbol{\lambda_2}$ **range**	**No. bins**
$\mathbf{0.1 < \eta < 0.3}$	$[-29, 40]$	$[-13, 21]$	2346
$\mathbf{0.7 < \eta < 0.9}$	$[-29, 39]$	$[-13, 17]$	2040
$\mathbf{1.2 < \eta < 1.4}$	$[-58, 69]$	$[-13, 17]$	3810
$\mathbf{2.0 < \eta < 2.2}$	$[-115, 148]$	$[-17, 18]$	9205

Table 6.1: **Range of values for λ_1 and λ_2 and total number of bins required in images for each η region, in order to cover all possible patterns from the pattern bank. Note that integer spacing between bins was chosen for simplicity, but the number of bins could be reduced by increasing bin size.**



(a) $\mathbf{0.1 < \eta < 0.3}$ (b) $\mathbf{0.7 < \eta < 0.9}$

(c) $\mathbf{1.2 < \eta < 1.4}$ (d) $\mathbf{2.0 < \eta < 2.2}$

Figure 6.3: **The same distributions of patterns in each η region as shown in Figure 4.5. In each case the scale has been adjusted to illustrate the requirements for the image boundaries in each region. The red box surrounding each distribution shows the minimum image boundaries needed to detect all available patterns.**

Figure 6.3 presents the distributions for each $\eta - \phi$ region (also shown in Figure 4.5) with the scales adjusted to easily see the extremal values of (λ_1, λ_2). From this, we see that for high-η regions the image boundaries must allow for the larger range in λ_1. The range of values and total number of bins required to perform the Hough tracking for each η region are summarised in Table 6.1 As shown, for the $0.1 < \eta < 0.3$ region, a minimum image size of 69x34 (determined by the range of values for (λ_1, λ_2)) would be required so as to be able to detect all muons contained within the sector from the pattern bank. This corresponds to

a minimum of 2346 bins in each image. While this gives an indication of the computational complexity of the task, the focus of this work is not to optimise the execution time of the algorithm, but to demonstrate its performance in track finding efficiency and fake track rate. Hence, for the purpose of clear illustration we will use 80x80 images for the remainder of this work. The Hough images are constructed layer-by-layer, as shown in Figure 6.4, which presents an illustrative example of partial images. Building the image in this way prevents any overlap of hits within the same layer, which are not uncommon especially if the precision of the PCA parameters is quite low and/or the hit occupancy is high - this is because the lines are drawn with a predetermined width motivated by the PCA residual distributions, which will be discussed in 6.2.2.

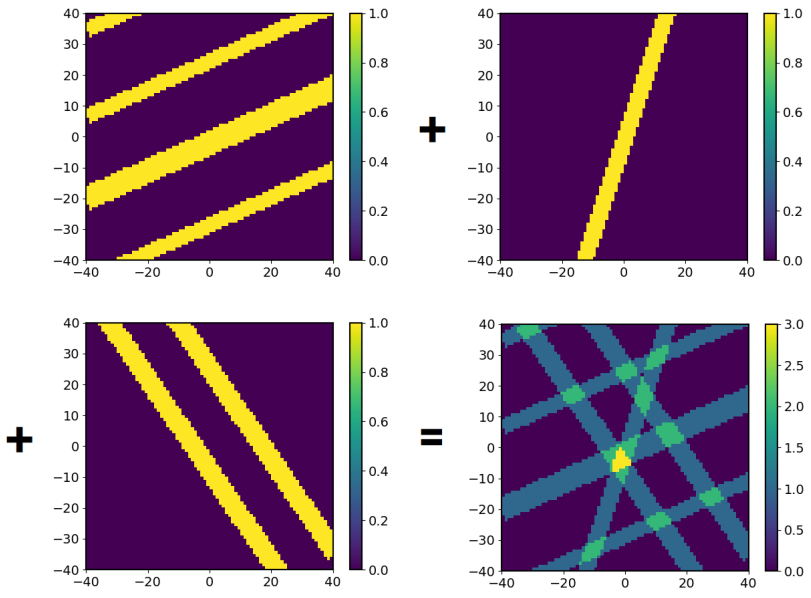


Figure 6.4: **An illustrative example of line drawing using PIL and addition of layer-wise arrays. The three component images are comprised of hits from the first, fifth and sixth strip layers (L1, L5 and L6), respectively.**

6.2.2 Residual Error Propagation

It has already been mentioned that the compression of patterns to a 2-dimensional space comes with some uncertainty that needs to be correctly modelled in the Hough-based tracking. Accounting for this uncertainty is both necessary from a statistical point of view and quite beneficial in terms of our ability to consolidate hits in a small region of the image. The first step is to assess what the uncertainty in superstrip position is before looking at how this maps to the Hough space. For each sector in the dictionary described in Section 4.1, all the patterns belonging to the sector are compressed into the 2-dimensional PCA space and then the inverse transformation is applied, in order to derive the estimated superstrip positions at each layer, as shown in Eq. 5.3, shown again here for convenience.

$$h_{ij} = p_{0i} + \lambda_1 p_{1i} + \lambda_2 p_{2i}$$

The differences between the estimated superstrip positions and original positions for each pattern are organised by layer index and the resulting histograms are

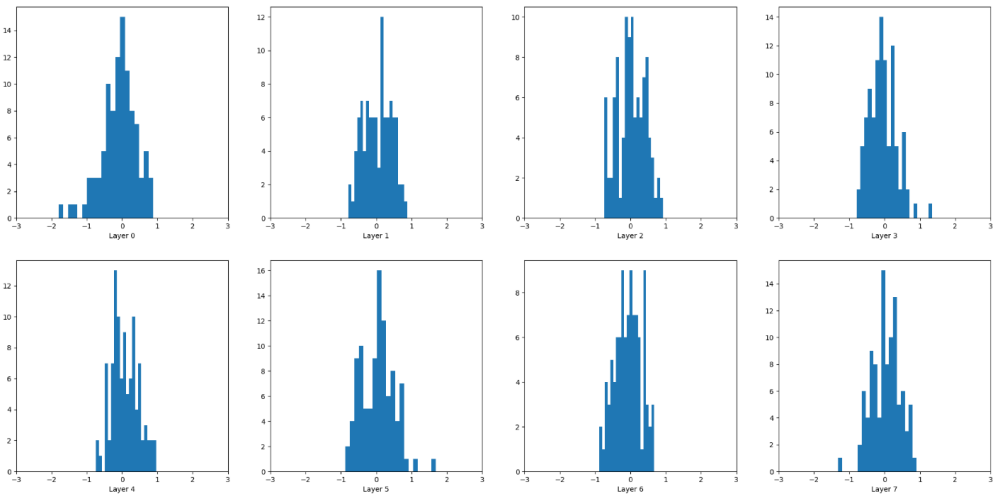
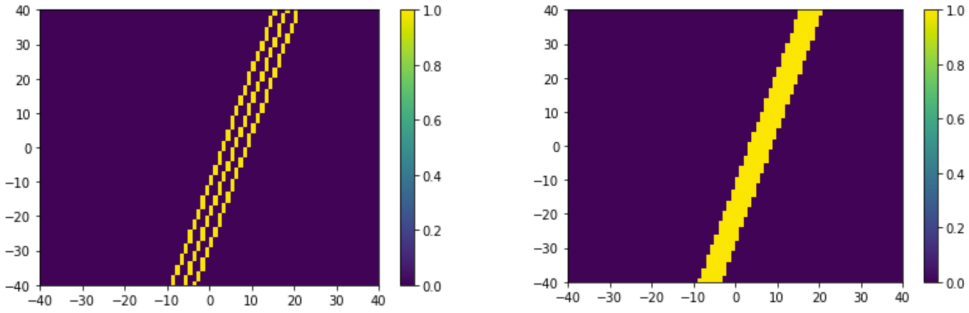


Figure 6.5: **Residual plots illustrating the uncertainty in superstrip measurements due to PCA for each layer in the most populous sector. The uncertainty in each layer is contained within the range allowed by the DC-bits.**



(a) **Hough superstrips for positions 14, 15 and 16.**

(b) **Hough superstrip for position 15 drawn with width = 5.**

Figure 6.6: **Comparison of three adjacent superstrips vs one central superstrip drawn with additional width. All superstrips belong to L4.**

shown in Figure 6.5. In each layer, we have an approximate Gaussian distribution of residuals with very few estimated hits that are more than ± 1 from the original value. The shape of these distributions is fairly consistent across sectors and has little dependence on the number of patterns in the sector. Since the Hough accumulator space is discrete, the width of the superstrip lines and uncertainty in position must be an integer. The distributions also have sharp boundaries, so to calculate the layer-wise uncertainty we use:

$$w_r = \text{ceil}(r_{max} - r_{min}) \tag{6.1}$$

where r_{max} and r_{min} are the maximum and minimum values in the residual distribution, respectively, and w_r is the residual width used to determine the Hough line width. The mapping of this uncertainty to the Hough space is then illus-

	L0	**L1**	**L2**	**L3**	**L4**	**L5**	**L6**	**L7**
Residual Width ($r_{max} - r_{min}$)	3	2	2	2	2	2	2	2
Line Width ($2w_r + 1$)	7	5	5	5	5	5	5	5

Table 6.2: **Maximum width of residual plots for each layer and the corresponding line width used in the filling of Hough histograms.**

trated in Figure 6.6. By transforming three adjacent superstrips in a particular layer we can see the distance between them in the image. From Figure 6.6a it is clear that the superstrips are not completely adjacent in this space and that there are "empty lines" between them. Figure 6.6b then shows that, for 3 adjacent superstrips, in order to obtain the same coverage in the Hough space with a single, central superstrip we require a width of 5. To account for the empty space between superstrips, then, in general, the width of the line must be $2w_r + 1$. The residual widths and resulting line widths for this particular sector are summarised in Table 6.2. The widths for each layer are stored for every sector and then used with the PCA parameters to generate each of the initial Hough images.

Max-Pooling Filter While the broadening of lines to account for the PCA uncertainty acts as a form of dilation, in order to ensure that each hit from a given track passes through the same pixel, it may be necessary to apply an additional dilation filter to the image (see Section 5.3). The type of dilation filter used here is a max-pooling filter, where a sliding window passes through the image and sets all pixels in the window to the maximum value within the window. This filter is applied at each iteration of the explaining away.

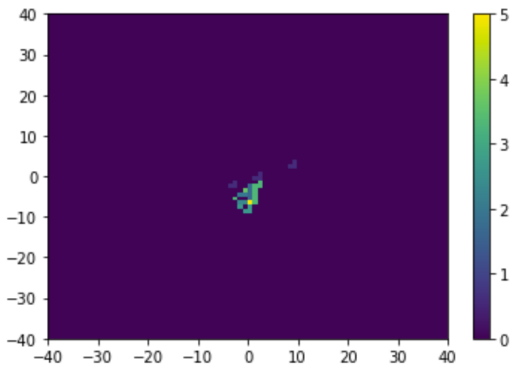
6.2.3 Hit-Map Pooling

Until this point in the description of the Hough-based tracking, we have only considered the images themselves and the absolute number of votes present in the accumulator space. From a final Hough image it is possible to see well-defined peaks and attribute a certain number of votes to them. However, it has not yet been discussed how the hit information is propagated through the algorithm such that we can extract the superstrip positions, detector elements and MC truth information from the peaks. This is important to be able to do for two reasons. Since a successful match can involve only 6 or 7 hits, the same combination of hits

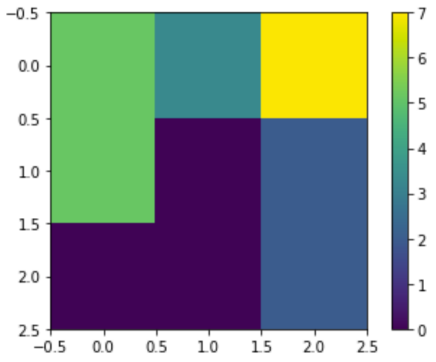
can be found in multiple sectors. By extracting defined hit-sets from the peaks we can avoid double counting the same combination of hits that may appear in different sectors, and we can also use the truth information to determine exactly how many muon hits are present in any peak. This can then be used to determine whether tracks are fake and assess the efficiency of the tracking algorithm.

To include this functionality, we define a *hit map* which relates the relevant hit information (layer index, superstrip position, MC truth) to the coordinates of the bins in which it has voted. In other words, the hit map summarises the hit-to-bin association. When the initial Hough image is produced, the hit map is filled out quite simply according to the initial bins that correspond to each hit. In parallel with the explaining away and max-pooling filter, the hit map is updated at each iteration in the following way:

- The difference between the max-pooled image and the original image is taken which then consists of all the bins whose values changed during the filtering. This step essentially prevents the need to scan through the whole image



(a) **The difference between the Hough image before and after applying max-pooling. The resulting image shows which pixels were affected by the pooling and gives the values they were changed to.**



(b) **An example of a 3x3 window surrounding one of the altered pixels in (a). This window is taken from the original image (before pooling) to demonstrate which neighbouring pixel had the maximum value.**

Figure 6.7: **Illustrative examples of the steps in expanding hit-maps according to the max-pooling filter applied to the Hough image.**

with the next steps and targets only the necessary bins. An example image is shown in Figure 6.7(a).

- Since the updated bins have been isolated, any hit that contains these in the hit map will need to be updated as well. For each bin, the surrounding 3x3 window is checked in the *original* image. Again, an example is shown in Figure 6.7(b). The maximum pixel(s) in this sub-image shares its votes with the central pixel during the max-pooling.

- Using the coordinates of this maximum pixel, the hit map is updated by first finding any hit entry in the map that contains the coordinates. To any hit containing the coordinates, the coordinates of the central pixel are added since the hit has essentially extended its reach to this pixel. Once this is done for every altered pixel, the hit-to-bin association has been updated according to the "sharing" of values between pixels.

The output of this part of the algorithm consists of a series of fully-processed Hough images containing blobs for each sector with associated hit maps to extract the hit information. The next section will focus on forming track candidates and obtaining more precise estimates of the track parameters, (λ_1, λ_2), with track fitting.

6.3 Track Reconstruction

6.3.1 Hit-to-Peak Association

The association of hits to each peak in the Hough image is central to building track candidates and assessing the efficiency of the Hough-based tracking. First, the peaks must be identified using the blob detection method outlined in Section 5.4. Any number of blobs may be detected in a single image assuming they are within the range allowed by the value of σ, the standard deviation of the Gaussian kernel. Once the blobs are detected, we extract the central window of pixels (the size of which is determined by a hyperparameter referred to as the *peak window*) and compare these with the corresponding hit map for the image. If any hit in the hit map contains any of the window pixels that are being searched for, the hit and its associated layer index are added to a track candidate dictionary. This dictionary records, for each layer, all the hits present in the peak. Once all the hits that were initially projected into the image have been checked, the track candidate dictionary may contain several possible branches depending on how many hits there are in each layer. For any layer with more than one hit, the values of (λ_1, λ_2) are used to estimate the superstrip position and select one of the branches. The PCA compression equation from Eq. 4.8 is used to calculate estimates for the superstrip position in each layer, and whichever measured superstrip is closest to the estimate is chosen. Finally, a hit-count threshold is applied to the number of layers that contain hits. This threshold is initially set to 8, ensuring that the track candidate must have hits in all 8 layers. However, the threshold may be changed depending on whether the relevant sector contains any wildcards. If so, the threshold is set to $8 - n_{wildcards}$ to allow for track candidates with fewer hits.

6.3.2 Track Fitting

The final stage in assessing whether or not a track is declared to have been found is performing a fit for the track candidate and setting a threshold on the sum of the χ^2 values across layers. In this way, only high quality track candidates are selected and low quality tracks (ideally, only fake tracks) are omitted. To perform the fitting, we once again use the Eq. 4.9:

$$\mathbf{p}(\lambda_1, \lambda_2) = \mathbf{p_0} + \lambda_1 \mathbf{p_1} + \lambda_2 \mathbf{p_2}.$$

The mean pattern and eigenpatterns for the sector as well as the measured superstrip positions in the track candidate are used to update the estimates of (λ_1, λ_2). This is achieved using the curve fit function in the scipy Python package [70]. With the more precise eigenvalues, the superstrip positions are recalculated and used to calculate the χ^2 value for each layer:

$$\chi_i^2 = \left(\frac{h_i - h_{meas}}{\sigma_i} \right)^2 \tag{6.2}$$

where σ_i is the uncertainty in the superstrip position in the $i\text{-}th$ layer. Finally, the total normalised χ^2 for the track candidate is obtained using:

$$\chi^2 = \sum_i \frac{\chi_i^2}{n_{dof}} \tag{6.3}$$

where n_{dof} is the number of degrees of freedom in the candidate. This is dependent on the number of wildcards in the sector, or simply the number of hits in the track and given by $n_{hits} - 2$. A cut is then applied to the normalised χ^2 values for each track candidate to distinguish true muon tracks from fake tracks. Finally, the pipeline concludes with a duplicate-removal step, which avoids counting the same set of hits in multiple tracks. The remaining tracks of sufficiently high quality are sorted by χ^2 value and, starting from the highest quality track, any subsequent

track sharing four or more hits with a preceding track is considered a clone and removed.

6.4 Results

6.4.1 Performance Evaluation

In this section, the performance of the tracking algorithm will be evaluated and the results discussed. To do so, we first outline how the efficiency is defined and lay out the default settings used. Several regimes will be used to extract the results and an optimisation of the hyperparameters will also be carried out. The final settings will be then tested on an independent set of events.

In order to define the efficiency, we use 200 Monte Carlo muon reference tracks from the $0.1 < \eta < 0.3$ region with $p_T > 2$ GeV. The tracks are generated with a pile-up of $< \mu > = 200$. As has been outlined in Section 4.2, the Hough space used for tracking corresponds to the compressed 2-dimensional PCA space produced for each sector. As was explained in 6.1.2, any sector that contained fewer than 30 patterns is omitted from the analysis on the basis that the precision of the PCA parameters is likely to be poor. The length of a track candidate is required to be 8 (requiring hits in all 8 layers) unless the sector contains wildcards, in which case the minimum length is given by $8 - n_{wc}$. The HTT TDR [36] defines a successful match as any sufficiently high-quality track candidate containing at least 80% MC truth muon hits. Rounding to the nearest integer, Table 6.3 then summarises how many muon hits are required in our implementation depending on the number of layers containing hits in the track candidate.

Equally, to avoid defining tracks that are known to consist of a mix of muon

	6-layers	**7-layers**	**8-layers**
No. muons required	5	6	6

Table 6.3: **The number of true muon hits required for track candidates with hits in n-layers in order to be considered a successful match.**

and pile-up hits, a fake track is defined using the same criteria. For a given event, a successful match is declared if any track candidate passes these hit-count thresholds, the global χ^2 threshold, and contains the sufficient number of muon hits according to the truth information. The number of additional "fake" tracks for the event is also recorded. The efficiency can then be calculated using:

$$\text{Efficiency} = \frac{\text{Number of events with at least one successful muon track}}{\text{Total number of events}} \tag{6.4}$$

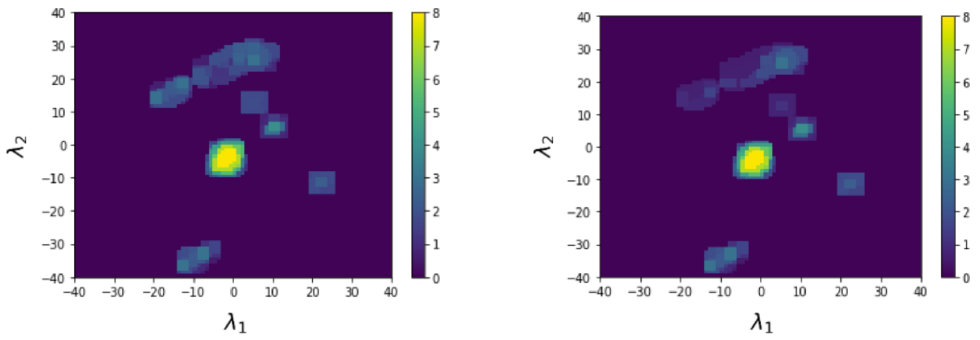
The uncertainty in the efficiency calculated from Bayesian statistics to be:

$$\text{SE}_{\text{eff}} = \sqrt{\frac{(k+1)(k+2)}{(n+2)(n+3)} - \frac{(k+1)^2}{(n+2)^2}} \tag{6.5}$$

where k is the number of succesful events (i.e. where the muon was detected) and n is the total number of events. Similarly, the average number of fake tracks can be is calculated using the mean with the uncertainty modelled by:

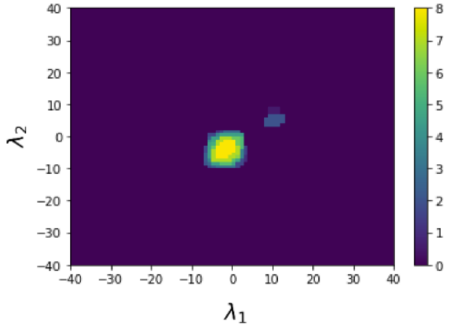
$$\text{SE}_{\text{ft}} = \frac{\sigma}{\sqrt{n}} \tag{6.6}$$

where σ is the standard deviation for the set of fake track values. In order to choose the default settings for the various hyperparameters, before investigating more optimal settings, we first ensure that the algorithm converges in a qualitative sense and that there are visible peaks in the images. Since the function of t_{line} and t_{pixel}, defined in 5.2, is essentially to reduce the number of iterations required, we want to find the highest possible values for these that allow for fast convergence without sacrificing the quality of the image peaks. With a fixed number of iterations ($i = 5$) and $t_{pixel} = 0.5$, some possible values of t_{line} are tested. As Figure 6.8 shows, greater values of t_{line} allow regions of density in the image to be explained away more quickly. However, above $t_{line} = 5$ the same image becomes empty

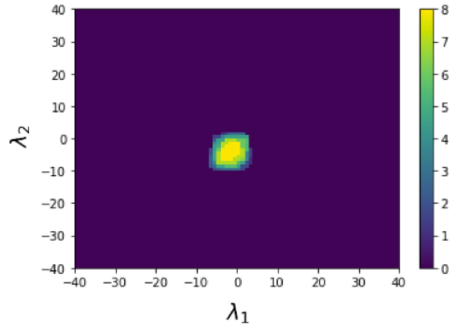


(a) $t_{line} = 2$

(b) $t_{ine} = 3$



(c) $t_{line} = 4$



(d) $t_{line} = 5$

Figure 6.8: **The Hough-transformed superstrips after 5 iterations of explaining away.**

because all lines end up being switched off. To ensure images with fewer than 8 muon hits don't vanish, an intermediate value such as $t_{line} = 4$ is a sensible choice as it is unlikely to switch all lines off and still shows fast convergence. A similar scan of possible t_{pixel} values produces similar images, but the time taken for the calculation is longer when t_{line} is low. This is because this threshold is not responsible for removing any hits from the image but only individual pixels from lines.

Additional variable parameters include the size of the max-pooling filter as well as the peak window. This window (e.g. 3x3) is used once the peaks have been detected and the central pixel identified. This pixel and its neighbours, decided by the size of the window, are then indexed in the hit-map to find which hits contribute to the peak. A greater window allows for more potential hits to be included in the track finding. Finally, there is the threshold for the minimum

Hyperparameter	Default Setting
No. iterations	5
t_{line}	4
t_{pixel}	0.5
Nearest neighbours for max-pool filter	1
Peak Window	3x3
Minimum pattern threshold (p_{min})	30

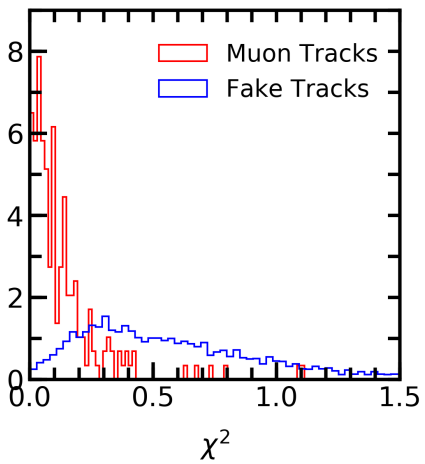
Table 6.4: **Summary of the hyperparameters and thresholds used in the imaging pipeline and their default settings.**

number of patterns in a sector, set to $p_{min} = 30$. The default settings used are shown in the Table 6.4. Using this setup and the criteria for track candidates and successful tracks, we aim to detect the 200 muon reference tracks and investigate any failures, before testing the improved settings on a further 400 reference tracks. First, the distribution of χ^2 values for muon track and fake track candidates is shown in Figure 6.9a. For each event, only the lowest-χ^2 muon track candidate is kept, since only one track is required for a successful match. Removing other low quality clones results in a clearer separation between the track types. As expected, muon tracks mostly occupy a low-χ^2 region and fake tracks have a much wider range of χ^2 values. Several thresholds could be implemented to obtain various efficiencies and average fake track rates. This trade-off can be seen in Figure 6.9b, which shows that the maximum available efficiency is $96.5 \pm 1.4\%$ for which an average of 16.1 ± 1.0 fake tracks are detected per event. It should be restated that, although we refer to these as fakes, some of the tracks, especially those with low-χ^2, may well be real pile-up tracks. With these settings, the tracking algorithm fails to produce any track candidates for 7 out of the 200 muons in the initial sample. As the distribution above shows, these would not be detected by any higher χ^2 threshold as the lowest quality muon can be seen at $\chi^2 = 1.1$. The next section will focus on optimising the hyperparameters of the tracking algorithm to

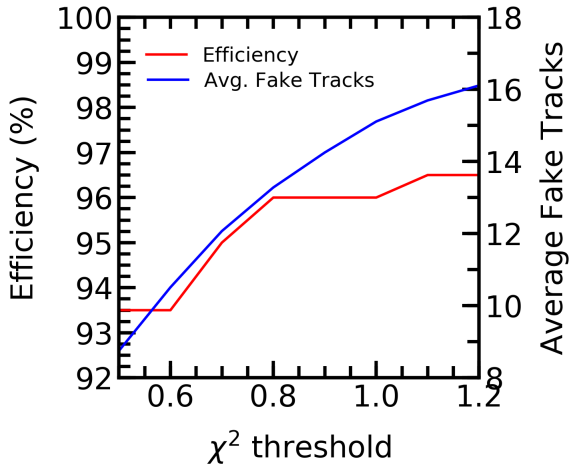
obtain a higher efficiency.

6.4.2 Analysis and Optimisation

There are several variable parameters to consider when trying to boost the efficiency, and so a general scan of each would be time-intensive and would lack any insight into the improvements that could be made. A more productive approach would be a targeted investigation to find the point in the pipeline where each muon is lost, and to make the necessary adjustments. It is helpful to work backwards and begin by checking whether any track candidates were made in the true muon sector, that is, the sector whose detector elements are shared exactly with the muon hits. If the muon is missing one or two hits because it passed through an inactive part of the detector, then the true sector will have wildcards in those layers. The purpose of this is to see whether an image with any good quality peaks were generated in this sector, in which case the issue is in the peak-to-hit



(a) **Distribution of χ^2 values for fake track and muon track candidates. For the muons, only the highest quality track candidate per event is kept.**



(b) **Trade-off between muon efficiency and average fake track detection for a range of χ^2 thresholds.**

Figure 6.9: **Preliminary results for Hough-based tracking using the hyperparameter settings outlined in Table 6.4.**

association. There may then be track candidates formed from the peak, but with an insufficient number of muon hits. If there are no track candidates in the sector, then we can check whether any image exists at all.

By looking at the image before the track finding process begins, we can gain insight as to whether the explaining away worked and whether or not any substantial peaks were formed. An example of such a case would be if a peak with maximum magnitude of 6 were present in a complete sector that should contain 8 muon hits. The length of the track candidates extracted from an image like this may not meet the minimum requirement and the muon would be lost here. This could be the case if the precision of the PCA parameters for the sector were low or if the presence of other fake tracks lowered the probability of votes being correctly cast for the muon peak. Finally, if no image exists in the sector at all then this must be an issue with the sector itself, namely, that there were likely no eigenpatterns to extract and use to perform the Hough transform. This would be because either the sector did not exist in the training data for the PCA space or because the number of patterns in the sector was too low (i.e. below the threshold p_{min}).

With this method of analysis, three reasons for efficiency loss were identified in the initial 200 muon sample:

- In one event, a track candidate is found in the true muon sector that contained only 5 MC truth muon hits. In total, the event contains 7 MC truth muon hits, and the sector has a single wildcard, so the 80% muon hit requirement is not met. In this case, it is likely that some combination of pile-up hits and muon hits yielded a lower χ^2 than muon-only hits in the track fitting stage.

- In five other events, it was found that the number of patterns in the true muon sector was less than the threshold of 30 patterns required in a sector

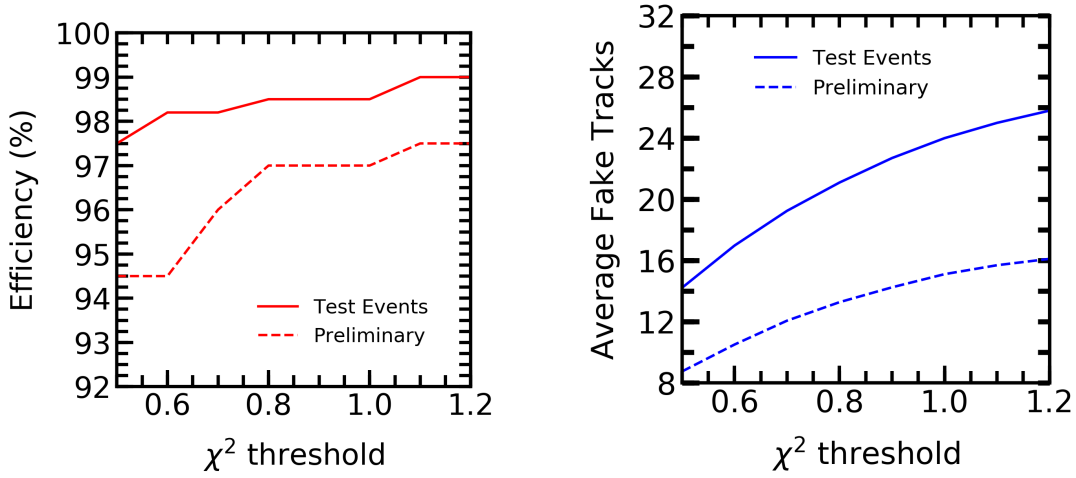
in order to perform the PCA, hence there were no images produced.

- In one event, the true muon sector was contained in a sector that did not exist in the sector dictionary (i.e., there were

Based on these findings, two changes were made: the minimum patterns threshold was lowered to $p_{min} = 15$ and the peak window extended to 9x9. The first change is motivated by the sectors in which no image was produced, and including them in the calculation should yield better results. For the second change, a 9x9 window around the central pixel in each peak is used to extract hit information, instead of a 3x3 window, to try and prevent track candidates being formed with one or two missing muon hits. The muon images are likely to intersect in a broad region and if the max-pooling did not manage to consolidate the hits previously, then casting a wider net to extract the hits from the peak may help. For the other hyperparameters, there is no reason to assume the default choice was not already sensible. Finally, any event for which the true muon sector is not present in the sector dictionary can be removed from the efficiency calculation, since this is caused by insufficient training data and we are interested in the efficiency with respect to muons that are detectable. In summary, the updated hyperparameters are shown in the Table 6.5, with the bold text indicating a change. The optimised

Hyperparameter	**Default Setting**
No. iterations	5
t_{line}	4
t_{pixel}	0.5
Nearest neighbours for max-pool filter	1
Peak Window	**9x9**
Minimum pattern threshold (p_{min})	**15**

Table 6.5: **Summary of the optimised hyperparameters and thresholds - the updated parameters are in bold font.**



(a) **Efficiency as a function of the χ^2 threshold.**

(b) **Average number of fake tracks per event as a function of the χ^2 threshold.**

Figure 6.10: **Summary of performance for Hough tracking algorithm. The preliminary sample consists of 200 muon reference tracks. These are used to perform some tuning of hyperparameters before testing on the test events sample, which consists of 400 different muon reference tracks.**

efficiency and rate of fake track detection are displayed in Figure 6.10.

As the figure shows, there is an increase in efficiency of 1.5% - 3% across the threshold values, at the expense of an increase of 6 - 10 fake tracks per event, depending on the value of χ^2. At the maximal available efficiency, this corresponds to 4 out of the 200 event test sample in which the muon was not detected. In each of these cases, it is the presence of pile-up hits that either disrupt the formation of clear muon peaks or are selected in favour of muon hits in the track fitting stage. (Note that this is not including any events for which the true muon sector does not exist in the sector dictionary). In summary, a maximal efficiency of $99 \pm 0.6\%$ is possible with an average fake track rate per event of 25.8 ± 0.7. Any remaining loss of efficiency is caused by either lowering the χ^2 threshold to reduce the fake track rate, or due an insufficient training set (pattern bank) used to construct the sector dictionary and PCA space. A brute force approach to improve the performance of the tracking algorithm would simply be to generate larger Monte

Carlo pattern banks and widen the coverage of detector space by the training patterns. Other forms of data augmentation on the existing pattern bank could be applied to avoid computationally expensive pattern bank generation, such as exploiting the rotational symmetry of the detector. The detector is symmetric with respect to the azimuthal angle, ϕ, so for any sector containing too few or no patterns at all, the set of detector elements that correspond to the same η and opposite ϕ should contain the same distribution of patterns. Hence, if a sector is empty and no Hough images are produced, the 'opposite' sector could be selected and the PCA parameters drawn from there.

6.4.3 Comparison of Performance

While the application of the Hough transform itself to charged particle tracking is not novel, our choice of parameterisation through the PCA training is. We have already demonstrated in Section 3.3 that the data-driven approach of finding the principal components that can define a pattern produces well separated Hough images in each layer. It was qualitatively shown that this parameterisation is well suited to the HT, but we must now compare the results with those from the existing pattern matching framework. Table **??** contains the results from the ATLAS-TDR-029 for the pattern matching performance of the Associative Memory (AM) step. Since we have only been investigating the HT tracking for

η range	muon eff.	mean matches pile-up	99% interval matches in pile-up	Superstrip width			DC bits
				pixel	barrel	end-cap	
$0.1 < \eta < 0.3$	99.1%	31	151	33/402	40		21111122
$0.7 < \eta < 0.9$	99.2%	21	93	33/402	40		21111122
$1.2 < \eta < 1.4$	98.8%	42	159	33/402	40	20	21111122
$2.0 < \eta < 2.2$	98.7%	10	56	16/200		10	21111122

Table 6.6: **Pattern-matching performance for the AM step simulated on minimum bias $< \mu > = 200$ pile-up events ($\eta \times \phi = 0.2 \times 0.2$ regions, $p_T > 4$ GeV) from the ATLAS-TDR-029 [36].**

	Muon Eff.	Avg. tracks per event
Pattern Matching	99.1%	31
HT Tracking	99.0 ±0.6%	25.8 ±0.7

Table 6.7: **Comparison of muon efficiency and average fake tracks per event for pattern matching vs HT-tracking algorithm.**

the $0.1 < \eta < 0.3$ region, only the results for this region can be used for comparison. We are interested in comparing the muon reconstruction efficiency as well as the "mean matches pile-up", which is equivalent to the average number of fake tracks in Figure 6.10 and describes the mean number of tracks per event matched to pile-up. The efficiency in this region is 99.1% with a mean number of matches of 31 tracks per event. Table 6.7 contains these results in comparison with the analogous HT results. With our tracking algorithm, there are at least 4.5 fewer tracks per event and the pattern matching efficiency is within the error range of the HT efficiency. While there is a track fitting stage in the HT algorithm, this is not the same as the track fit that these tracks are then passed into after the pattern matching. Hence, the lower track rate from the HT results in significantly fewer track fits being computed at the next stage. An even greater drop in average tracks per event could be obtained if efficiency were somewhat sacrificed or if the additional events whose true muon sector could not be looked up in the sector dictionary could be included.

Chapter 7

Summary and Outlook

The aim of this work was to develop a novel tracking methodology for multi-element silicon detectors using the popular computer vision technique, the Hough Transform (HT). While there have been many attempts to use the HT in high energy physics experiments for this purpose before, we present a more statistically motivated method that circumvents many of the previous drawbacks. There are two significant distinctions that we present; the first is to divide the detector volume into detector element groups called sectors, derived from most the most probable trajectories of Monte Carlo simulated muon tracks. The second is to use these large pattern banks of muon tracks to train a Principal Component Analysis (PCA) model and extract a compressed, 2-dimensional representation of tracks in a sector. Through both the lossless sector dictionary-based compression and the PCA compression, we can reduce the memory requirements for any pattern-based track finding algorithm by approximately a factor of 5, sacrificing a negligible loss in track finding efficiency. In our case, the compressed representation is then used as a statistically motivated parameterisation for the application of the Hough Transform. By applying the transform at the sector-level, the occupancy of the resulting images, even with high levels of pile-up, is low enough to efficiently isolate single muon tracks. Furthermore, using PCA to parameterise the tracks (as

opposed to some ad-hoc parameterisation) leads to very little correlation between the track parameters and produces well-defined, circularly symmetric peaks. This leads to advantages and simplifications in the subsequent steps of the pattern recognition, such as the simpler application of blob detection and hit-to-peak association.

We find that, compared with current pattern matching methods for tracking, our HT-based tracking algorithm achieves comparable muon detection efficiency and requires, on average, fewer track candidates to process in the subsequent track fitting stage. The small, observed loss in efficiency beyond this is the result of insufficient training data for the PCA. To improve the training of the PCA parameters, it is required that the pattern banks include rarer corner cases of tracks. This could be achieved by using larger MC generated pattern banks or by exploiting the symmetry of the detector to provide PCA parameters to sectors which had little to no available training data in the current study. Additionally, while this project does demonstrate the potential of the techniques employed, further work would need to be carried out to reduce the computational complexity of the algorithm and optimise execution time for any specific hardware, such as GPUs.

As levels of pile-up increase in ATLAS over the coming years, it is essential to maintain efficient tracking capabilities and reduce the reconstruction walltime per event. The findings from this work present a promising direction that can be taken in ATLAS to achieve this goal. Outside the field of high energy physics, the novel application of PCA compression for Hough Transform parameterisation could be leveraged for many object detection tasks for which sufficient training data is available.

List of Figures

List of Tables

Bibliography

[1] *LEP design report.* CERN, Geneva, 1984. Copies shelved as reports in LEP, PS and SPS libraries.

[2] Esma Mobs. The CERN accelerator complex - August 2018. Complexe des accélérateurs du CERN - Août 2018. Aug 2018. General Photo.

[3] Framework TDR for the LHCb Upgrade: Technical Design Report. Technical report, Apr 2012.

[4] K. Aamodt et al. The ALICE experiment at the CERN LHC. *JINST*, 3:S08002, 2008.

[5] *Technical proposal.* LHC technical proposal. CERN, Geneva, 1994. Cover title : CMS, the Compact Muon Solenoid : technical proposal.

[6] Lyndon Evans and Philip Bryant. LHC machine. *Journal of Instrumentation*, 3(08):S08001–S08001, aug 2008.

[7] CERN. Longer term lhc schedule. http://lhc-commissioning.web.cern.ch/schedule/LHC-long-term.htm.

[8] Technical Design Report for the ATLAS Inner Tracker Pixel Detector. Technical report, CERN, Geneva, Sep 2017.

[9] Joao Pequenao. Computer generated image of the whole ATLAS detector. Mar 2008.

[10] *ATLAS detector and physics performance: Technical Design Report, 1.* Technical design report. ATLAS. CERN, Geneva, 1999.

[11] G. Aad et al. ATLAS pixel detector electronics and sensors. *JINST*, 3:P07007, 2008.

[12] Alessandro La Rosa. The atlas insertable b-layer: from construction to operation, 2016.

[13] A. Polini et al. Design of the ATLAS IBL Readout System. Technical report, CERN, Geneva, Oct 2011.

[14] Technical Design Report for the ATLAS Inner Tracker Strip Detector. Technical report, CERN, Geneva, Apr 2017.

[15] D M S Sultan. *Development of Small-Pitch, Thin 3D Sensors for Pixel Detector Upgrades at HL-LHC.* PhD thesis, 06 2017.

[16] Aranzazu Ruiz-Martinez and ATLAS Collaboration. The Run-2 ATLAS Trigger System. Technical report, CERN, Geneva, Feb 2016.

[17] R Mankel. Pattern recognition and event reconstruction in particle physics experiments. *Reports on Progress in Physics*, 67(4):553–622, Mar 2004.

[18] Pramod R. Gunjal, Bhagyashri R. Gunjal, Haribhau A. Shinde, Swapnil M. Vanam, and Sachin S. Aher. Moving object tracking using kalman filter. In *2018 International Conference On Advances in Communication and Computing Technology (ICACCT)*, pages 544–547, 2018.

[19] Andreas Salzburger. *Track Simulation and Reconstruction in the ATLAS experiment.* PhD thesis, Innsbruck U., 2008.

[20] Sebastian Fleischmann. *Track reconstruction in the ATLAS experiment: The deterministic annealing filter.* PhD thesis, Wuppertal U., 2006.

[21] ATLAS and CMS Collaborations. Report on the physics at the hl-lhc and perspectives for the he-lhc, 2019.

[22] Georges Aad et al. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys. Lett. B*, 716:1–29, 2012.

[23] Serguei Chatrchyan et al. Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC. *Phys. Lett. B*, 716:30–61, 2012.

[24] A. Pich. Electroweak symmetry breaking and the higgs boson. *Acta Physica Polonica B*, 47(1):151, 2016.

[25] A detailed map of higgs boson interactions by the ATLAS experiment ten years after the discovery. *Nature*, jul 2022.

[26] Wouter Verkerke. Constraints on Higgs boson couplings from a combination of ATLAS and CMS measurements. Constraints on Higgs boson couplings from a combination of ATLAS and CMS measurements. Sep 2015.

[27] M. Cepeda et al. Report from Working Group 2: Higgs Physics at the HL-LHC and HE-LHC. *CERN Yellow Rep. Monogr.*, 7:221–584, 2019.

[28] V. A. Bednyakov, N. D. Giokaris, and A. V. Bednyakov. On the higgs mass generation mechanism in the standard model. *Physics of Particles and Nuclei*, 39(1):13–36, jan 2008.

[29] G. Degrassi, M. Fedele, and P.P. Giardino. Constraints on the trilinear higgs self coupling from precision observables. *Journal of High Energy Physics*, 2017(4), apr 2017.

[30] Adel Bilal. Introduction to supersymmetry, 2001.

[31] CSABA CSÁ KI. THE MINIMAL SUPERSYMMETRIC STANDARD MODEL. *Modern Physics Letters A*, 11(08):599–613, mar 1996.

[32] Xabier Cid Vidal et al. Report from Working Group 3: Beyond the Standard Model physics at the HL-LHC and HE-LHC. *CERN Yellow Rep. Monogr.*, 7:585–865, 2019.

[33] Anders Ryd and Louise Skinnari. Tracking triggers for the HL-LHC. *Annual Review of Nuclear and Particle Science*, 70(1):171–195, oct 2020.

[34] M Shochet, L Tompkins, V Cavaliere, P Giannetti, A Annovi, and G Volpi. Fast TracKer (FTK) Technical Design Report. Technical report, Jun 2013. ATLAS Fast Tracker Technical Design Report.

[35] Johanna Gramling. Hardware-based tracking at trigger level for atlas: The fast tracker (ftk) project.

[36] Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System. Technical report, CERN, Geneva, Sep 2017.

[37] Peter E. Hart. How the hough transform was invented [dsp history]. *IEEE Signal Processing Magazine*, 26(6):18–22, 2009.

[38] P .V .C. Hough. Method and means for recognizing complex patterns,".

[39] Riccardo Aramini, Fabrice Delbary, Mauro C. Beltrametti, Michele Piana, and Anna Maria Massone. The radon transform and the hough transform: a unifying perspective, 2016.

[40] Azriel Rosenfeld. Picture processing by computer. *ACM Comput. Surv.*, 1(3):335–336, September 1969.

[41] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[42] T. Tuytelaars, L. Van Gool, M. Proesmans, and T. Moons. The cascaded hough transform as an aid in aerial image interpretation. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 67–72, 1998.

[43] Priyanka Mukhopadhyay and Bidyut B. Chaudhuri. A survey of hough transform. *Pattern Recognition*, 48(3):993–1010, 2015.

[44] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[45] D.J. Kerbyson and T.J. Atherton. Circle detection using hough transform filters. In *Fifth International Conference on Image Processing and its Applications, 1995.*, pages 370–374, 1995.

[46] E.R. Davies. A modified hough scheme for general circle location. *Pattern Recognition Letters*, 7(1):37–43, 1988.

[47] Pär Kierkegaard. A method for detection of circular arcs based on the hough transform. *Mach. Vision Appl.*, 5(4):249–263, September 1992.

[48] HK Yuen, J Princen, J Illingworth, and J Kittler. Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, 8(1):71–77, 1990.

[49] N. Kiryati, Y. Eldar, and A.M. Bruckstein. A probabilistic hough transform. *Pattern Recognition*, 24(4):303–316, 1991.

[50] Lei Xu, Erkki Oja, and Pekka Kultanen. A new curve detection method: Randomized hough transform (rht). *Pattern Recognition Letters*, 11(5):331–338, 1990.

[51] Joon H. Han, LászlóT. Kóczy, and Timothy Poston. Fuzzy hough transform. *Pattern Recognition Letters*, 15(7):649–658, 1994.

[52] Olga Barinova, Victor Lempitsky, and Pushmeet Kholi. On detection of multiple object instances using hough transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1773–1784, 2012.

[53] Introduction to atlas. "https://wikihost.uib.no/ift/images/a/ab/ATLASperformance.pdf".

[54] Hough concept and ideas. "https://indico.cern.ch/event/986990/contributions/4195284/attachments/2175787/3674003/HoughIntro.pdf".

[55] Nicola Pozzobon, Fabio Montecassiano, and Pierluigi Zotto. A novel approach to hough transform for implementation in fast triggers. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 834:81–97, 2016.

[56] TIBCO Data Science Team Studio User's Guide. Principal component analysis | data modeling and model validation. "https://docs.tibco.com/pub/sfire-dsc/6.5.0/doc/html/TIB_sfire-dsc_user-guide/GUID-BF34695C-47FD-44CA-9002-8A3FA69FB4E0.html".

[57] Jonathon Shlens. A tutorial on principal component analysis, 2014.

[58] Hirotaka Niitsuma and Takashi Okada. Covariance and pca for categorical variables, 2007.

[59] Abel Folch-Fortuny, Francisco Arteaga, and Alberto Ferrer. Pca model building with missing data: New proposals and a comparative study. *Chemometrics and Intelligent Laboratory Systems*, 146, 05 2015.

[60] John Baines, Richard Brenner, Viviana Cavaliere, Alessandro Cerri, Markus Elsing, Dmitry Emeliyanov, Igor Gavrilenko, Fabian Klimpel, Ulf Fredrik Mikael Martensson, Jiri Masik, Marek Palka, Charlie Pitman Donaldson, Frank Winklmeier, and Benjamin Wynne. Report of the ATLAS Alternative Event Filter Tracking Working Group. Technical report, CERN, Geneva, Oct 2019.

[61] M. Sharifi, M. Fathy, and M.T. Mahmoudi. A classified and comparative study of edge detection algorithms. In *Proceedings. International Conference on Information Technology: Coding and Computing*, pages 117–120, 2002.

[62] Liyong Ma, Yude Sun, Naizhang Feng, and Zheng Liu. Image fast template matching algorithm based on projection and sequential similarity detecting. In *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 957–960, 2009.

[63] M.W. Spratling. A neural implementation of the hough transform and the advantages of explaining away. *Image and Vision Computing*, 52:15–24, 2016.

[64] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. https://scikit-image.org/docs/dev/auto_examples/applications/plot_morphology.html, 2014.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn:

Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[66] Lucas Assirati, Núbia Rosa, L. Berton, Alneu Lopes, and Odemir Bruno. Performing edge detection by difference of gaussians using q-gaussian kernels. *Journal of Physics Conference Series*, 490, 11 2013.

[67] Stanford Silvio Savarese. Lecture 10, detectors and descriptors, p15. https://pdfslide.net/documents/lecture-10-detectors-and-descriptors-silvio-savarese-silvio-savarese-lecture.html.

[68] Colin Flanagan. The bresenham line-drawing algorithm. https://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html.

[69] P Umesh. Image processing in python. *CSI Communications*, 23, 2012.

[70] SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.