LUND UNIVERSITY

Department of Physics

Division of Particle and Nuclear Physics

Pattern-Based Electron Counting Algorithm for LDMX

Lucia Kvarnström Meissner

Supervisor: Dr. Lene Kristian Bryngemark

Co-Supervisor: Dr. Ruth Pöttgen

15 HP Bachelor of Science Thesis

Project Duration: 2 months

Examined June 2026

Abstract

The Trigger Scintillator (TS) of the Light Dark Matter eXperiment (LDMX) is responsible for counting the electrons in the incoming beam in order to accurately determine the trigger threshold used in identifying missing momentum events. Electrons are counted by reconstructing their approximately horizontal tracks through the TS. The software algorithm operates on a maximum vertical tolerance value, making it both computationally expensive and inflexible to a misalignment in the TS geometry.

This thesis proposes an alternate counting algorithm using a data-driven lookup table (LUT) containing real electron patterns derived from simulation or data. This allows the tracking definition to naturally adapt to the TS geometry presented in the data. To write the LUT, electrons' characteristic propagation patterns are first examined using truth-level information and then compared to unfiltered simulation data. It is shown that real tracks can be isolated from random combinatorics patterns by applying a minimum pattern frequency threshold to simulated data. At an optimal threshold of 0.0008, the data-driven LUT is able to achieve a maximum tracking efficiency ϵ_{Trk}=99.37% for a fake track rate $r_{fake} = 0.001668\%$, compared to the tolerance definition algorithm's $\epsilon_{Trk} = 98.00\%$ and $r_{fake} = 0.00073\%$. This study demonstrates the feasibility of a data-driven tracking method, with potential for further improvement through refinement of the frequency-threshold definition.

Acknowledgements

Thank you firstly to Lene Kristian Bryngemark, not only for your help and guidance, but also for the opportunity to pursue this project in the first place. I would also like to thank Ruth Pöttgen, who first introduced me to LDMX and experimental particle physics as a whole, as well as everyone from the student workshops that took the time to listen to all my questions. I appreciate all of the answers. I also extend special thanks to Hans Alin for all his extra help and patience—and last but never least, to my friend Wilde Hedin for her company.

Abbreviations

ADC: **A**nalog-to-**D**igital **C**onverter

CMB: **C**osmic **M**icrowave **B**ackground

DM: **D**ark **M**atter

ECal: **E**lectromagnetic **Cal**orimeter

GEANT4: **GE**ometry **AN**d **T**racking 4

HCal: **H**adronic **Cal**orimeter

LDMX: **L**ight **D**ark **M**atter e**X**periment

`ldmx-sw`: **L**ight **D**ark **M**atter e**X**periment **S**oft**w**are

LUT: **L**ook**u**p **T**able

MeV,GeV,TeV: **M**ega-, **G**iga-, **T**era- **e**lectron**V**olt

PE: **P**hoto**e**lectron

PVT: **P**oly**v**inyl**t**oulene

SiPM: **Si**licon **P**hoto**m**ultiplier

SM: **S**tandard **M**odel

SRT: **S**ilicon **R**ecoil **T**racker

STT: **S**ilicon **T**agging **T**racker

TDC: **T**ime-to-**D**igital **C**onverter

TS: **T**rigger **S**cintillator

WIMP: **W**eakly **I**nteracting **M**assive **P**article

Contents

1 Introduction

Dark matter (DM) lies at the intersection of various physics subfields, first inferred from cosmological observations as early as 1930[1], then solidified as a valid theory in the late 1970s[2]. DM is one of many phenomena which has disrupted the once seemingly comprehensive Standard Model (SM), so the experimental attention it receives today comes as no surprise. Some approaches seek to detect it indirectly via the missing momentum signature, a large loss of energy in SM particles to indicate production of DM. One such effort is the Light Dark Matter eXperiment (LDMX), designed to probe the MeV-GeV mass range of so-called *thermal relic DM* models with an unparalleled increase in sensitivity.

LDMX will fire an 8 GeV electron beam toward a sheet of tungsten to catalyze production of dark particles—largely through dark bremsstrahlung[3]. The scarcity of this process requires detector components with precise reconstruction capability, including a three-layer Trigger Scintillator (TS) to count the number of beam electrons with high fidelity. Knowing the number of beam electrons is vital in determining the energy lost in DM production. A beam electron passes through the TS's three modules prior to collision, producing signals in each module from which its track can be reconstructed[4].

LDMX's software framework contains an existing approach to track reconstruction. A tolerance-based tracking algorithm defines a configurable maximum vertical deviation from a perfectly horizontal trajectory that an electron may propagate between the first and last modules. Relying on any form of geometric constraints, however, makes the algorithm vulnerable to a misalignment in the detector geometry, which is likely inevitable in realistic experimental conditions. To combat this, one could instead implement a data-driven approach, allowing the algorithm to naturally adapt to any misalignment shown in the data.

This study realizes this method. Physically valid electron propagation patterns are derived from simulation and written to a lookup table (LUT), which creates tracks by matching possible signal combinations from the modules to pre-validated track patterns. Additionally, LUT-based tracking is already employed in a separate firmware tracking method, demonstrating its ability to function within the strict timing constraints of LDMX. First, truth-level electron patterns, as shown in simulation, are observed to gain an understanding of the electrons' true behaviors. A method is then derived for isolating these valid patterns from unfiltered simulation data, by applying a pattern frequency threshold to suppress any false, low-frequency combinations. The LUT is constructed to contain only the most frequently appearing patterns, and its tracking performance is then evaluated as compared to the existing distance-based tolerance tracker. This method aims to achieve an efficiency to match that of the existing algorithm while also striving to remain robust in future misalignment studies.

2 Background

2.1 Dark Matter

The idea of dark matter was first suggested by cosmology in the twentieth century, when the observation of motion in galaxies and galaxy clusters implied that there must exist unobservable, extra mass in the universe[1]. Astrophysicists observed seemingly unexplainable galactic-scale effects, such as gravitational lensing—light from faraway galaxies seems to bend as it travels around other objects on its path to observers on Earth. The gravity of ordinary matter is not enormous enough to explain the magnitude of this effect, nor is it able to explain the rotational velocity profiles of spiral galaxies. Even the outskirts of our own Milky Way move at speeds higher than what its visible mass seems to suggest[5]. Additional obscurity emerges from the Cosmic Microwave Background (CMB), whose perturbations would require the existence of matter that does not interact with photons[6]. With mysteries emerging from every corner of astrophysics, only the explanation of some extra, invisible mass seems to answer all questions. In short, dark matter acts as the invisible skeleton of the universe. Albeit hidden, its existence can be inferred from the structure of visible matter around it.

While many of the details surrounding dark matter are unknown, the theory is largely accepted. Alternate models are able to explain some phenomena, but fail at others. The theory of modified Newtonian dynamics, for example, tweaks Newton's law of gravity in order to explain galactic rotational curves, but falls short with no explanation for CMB fluctuations[7]. Despite the overwhelming favor for dark matter, its constituents remain unknown. Its gravitational interaction with ordinary matter has been well established, but some theories also predict a weak coupling to SM particles on a microscopic level[2].

Experimental attempts at detection of DM particles therefore predominantly favor *thermal relic* dark matter, a theory which postulates that dark matter originated together with visible matter in the thermal equilibrium of the early universe, implying a small non-gravitational interaction between them. This assumes that dark and visible particles were annihilating and decaying into one another at equal rates shortly after the Big Bang. The universe then proceeded to cool and expand enormously. The abundance of dark matter particles subsequently decreased with time and, at one point, eventually became fixed—a mechanism named the *freeze out*[2].

So, there exists a fixed amount of thermal relic dark matter in the universe today, and it has a theoretical production mechanism via SM particles. These dark particles are required to lie in the MeV-TeV mass range[2]. The most famous candidate in the upper half of this region (GeV-TeV) is named the WIMP (Weakly Interacting Massive Particle)[8]. Accelerator-based experiments are ideal for studying the possible production mechanisms of WIMPs, yet no experiment has concretely detected any thus far. The lack of dark signals in the upper edge of this region therefore incentivizes searches in the lower

half[9]. One such search is the Light Dark Matter eXperiment (LDMX).

2.2 Introduction to LDMX

The Light Dark Matter eXperiment (LDMX) is a proposed experiment to search for dark matter in the sub-GeV range. The experiment directs an 8 GeV electron beam onto a thin tungsten target with the primary objective of producing dark particles via dark bremsstrahlung. Dark bremsstrahlung, as illustrated in Fig. 1, refers to the scattering of a high-energy electron off an atomic nucleus, producing DM through a hypothetical *dark photon* mediator[8]. LDMX aims to identify and observe events where post-collision electrons are scattered from the target with missing momentum—the dark matter signature. A dark matter signal is characterized by a significant loss of energy. The dark particles go unnoticed by the LDMX detectors, but they are the carriers of this seemingly missing energy[3].

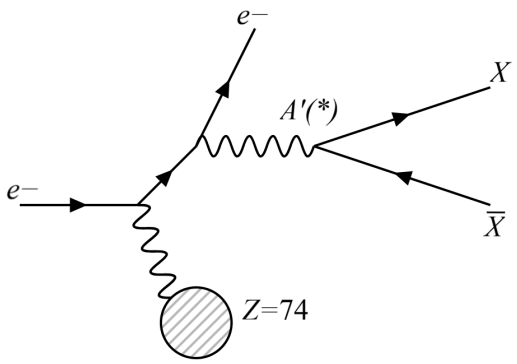


Figure 1: An electron e^- undergoes dark bremsstrahlung as predicted to occur in LDMX, colliding with a tungsten nucleus $Z = 74$ to produce dark sector particles $X, \overline{X}$ via a dark mediator $A'(^*)$.

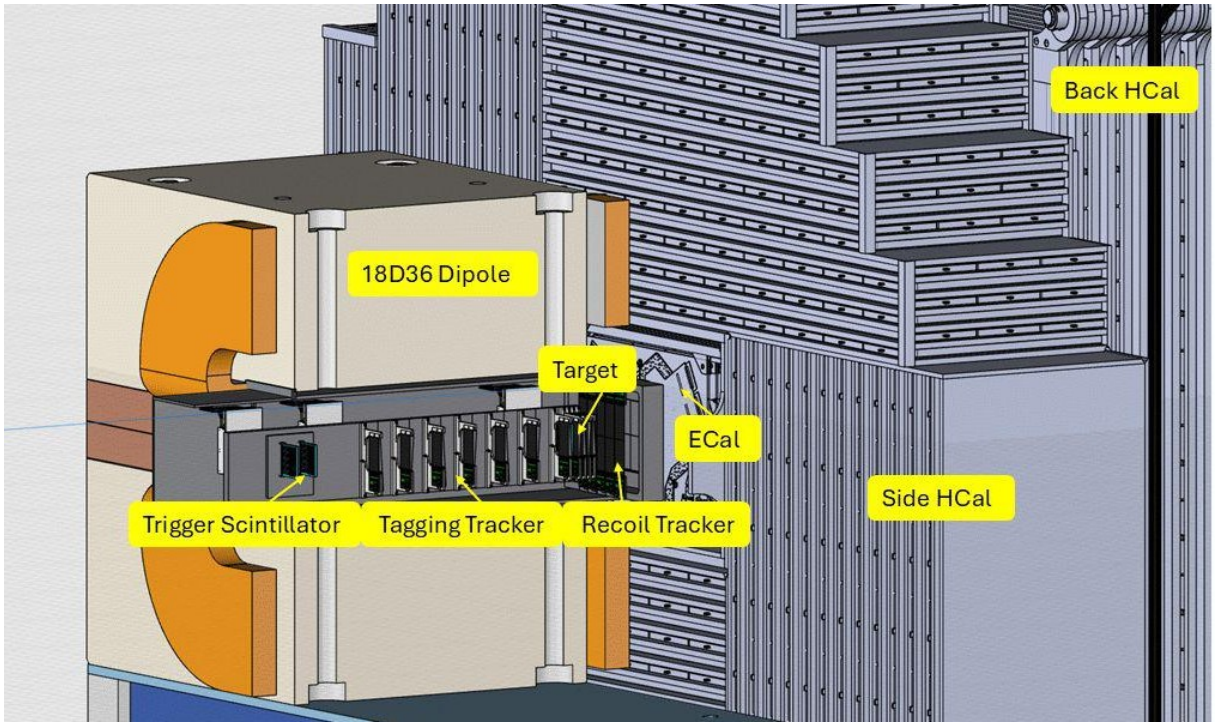


Figure 2: Setup of detector components in cutaway view, including the TS, STT, Dipole Magnet, Target, SRT, ECal, and HCal, side and back. Image: [3].

The design of the detector subsystems (see Fig. 2) is optimized for accurate measurement of electron kinematics before and after collision, since analyzing their energy and momenta is crucial to identifying dark matter signals. Prior to collision, the beam (in bunches of 37.1 MHz) passes through a Silicon Tagging Tracker (STT) and a Trigger Scintillator (TS) system, both contained within a dipole magnet placed before the target. The magnetic field causes charged particles to curve while passing through, from which their momenta can be measured. Directly behind the target sits a Silicon Recoil Tracker (SRT). The beam may be contaminated with off-energy electrons or non-electron particles. In order to isolate on-energy electrons, each beam particle's momentum must be precisely measured. The STT performs this measurement for the beam pre-collision, and the SRT does the same for the post-collision beam[3]. The TS assists in *triggering*—a filtration mechanism that determines which candidate events are recorded for later analysis—by providing a count of the incoming beam electrons. Only ~0.01% of the experiment's total events are selected for data storage[10]. In this case, the trigger is a minimum energy threshold applied to target anomalously low-energy DM signals. Setting this threshold requires knowing the number of beam electrons, thus the TS's primary objective is electron counting[3].

Behind the SRT lies an Electromagnetic Calorimeter (ECal) surrounded by a Hadronic Calorimeter (HCal) to measure the energies of post-collision particles. Scattered electrons and photons produce electromagnetic showers of energy, detected in the ECal. Though LDMX's primary objective is to observe dark bremsstrahlung, other background processes can introduce neutral hadrons and muons to the detector environment. While both calorimeters measure the energies of incoming particles, the primary objective of the HCal is to identify and subsequently eliminate these background events[3][4].

2.3 Trigger Scintillator

A DM signal is indicated by large amounts of missing energy in the ECal, and as such, the ECal requires a low-energy minimum trigger threshold. This trigger is set with the help of the TS, which must accurately determine the number of electrons n in each incoming beam pulse in order to do so. Functionality requirements define a TS maximum overcounting rate of 1 kHz. The relation between n and the missing energy E_{miss} is as follows:

$$E_{miss} = nE_{beam} - E_{ECal},$$

where E_{beam} is the energy of a single beam electron and E_{ECal} is the total energy detected in the ECal[3]. Events with large energy deposits in the ECal may still indicate DM production, depending on the number of electrons in the event. It is large energy loss in any one electron which is of interest, and the trigger is therefore different depending on the number of beam electrons within a given event[10].

The TS comprises three modules, all connected to a support plate which also houses the STT, located between modules 2 and 3. The first two modules are closely positioned, while the third lies further downstream just in front of the target. The construction of the TS—not including the STT and target—is shown in Figure 3. Each module is built from two staggered columns of 24 PVT scintillator bars, for a total of 48 bars per module. The columns are offset such that a beam electron cannot pass through the TS without being detected at every module. A thin layer of reflective film surrounds each bar to prevent photon leakage between the transparent scintillators. Located behind each module, on the opposite side of the support plate to minimize interaction with beam electrons, sits a Silicon Photomultiplier (SiPM) board. Light pipes connect each bar to the boards.

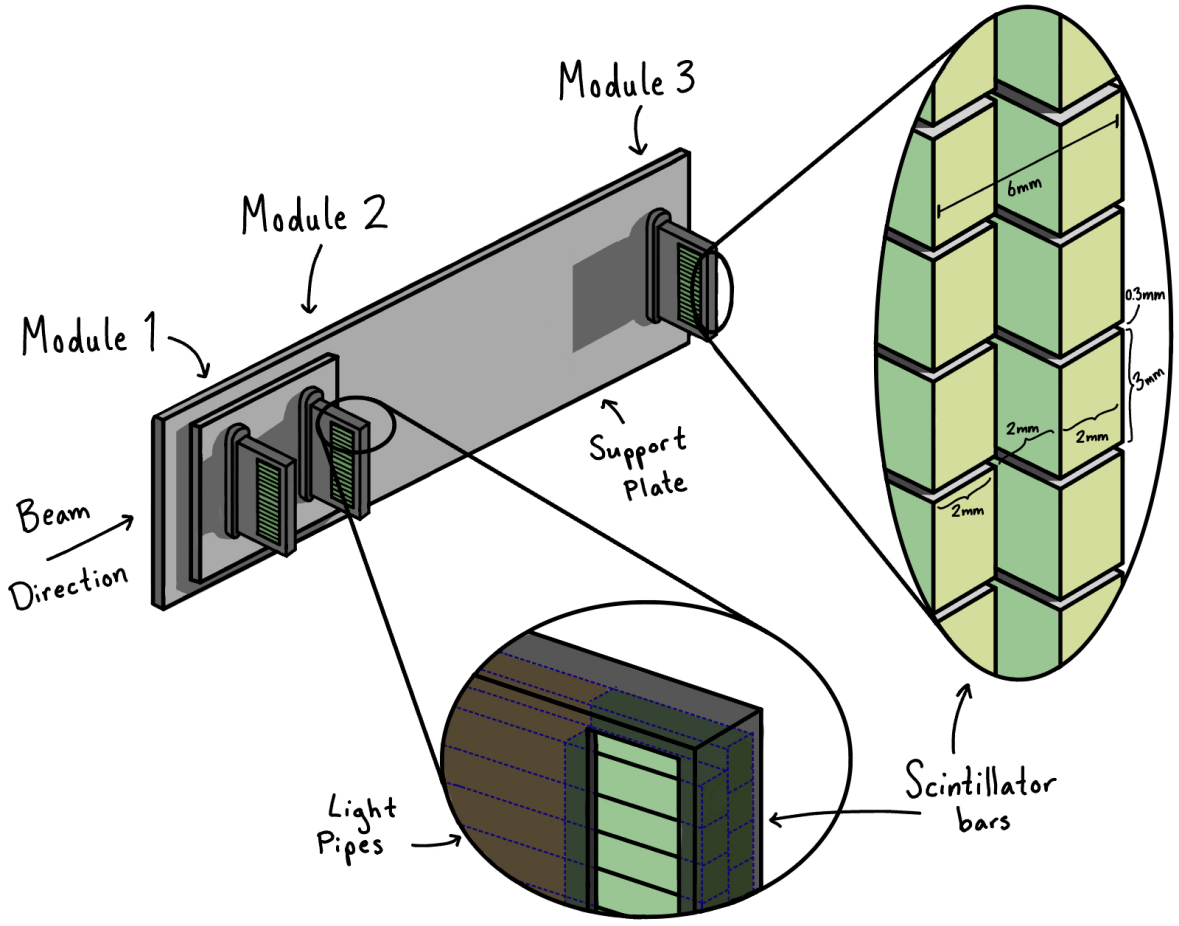


Figure 3: TS setup indicating three modules, scintillator bars, light pipes, support plate. SiPMs are hidden on the other side of the plate. Readout electronics not included.

An electron passing through a bar results in a number of scintillated photons, referred to as a *hit*. The SiPMs give each hit an associated photoelectron (PE) count, which acts as a quantization of the hit's energy deposition. Frontend electronics sit coupled to the SiPMs on the other side of the support plate, responsible for converting the SiPM output to ADC (Analog-to-Digital Converter) and TDC (Time-to-Digital Converter) values, representing the PE count and the timing of each hit, respectively[3].

3 Method: Software and Analysis

As of spring 2026, LDMX awaits formal approval. Analysis carried out during the course of this project is instead performed using simulation and detection software, and all code developed for this project can be found on GitHub[11]. A *data-driven* LUT will be derived from real results once LDMX begins data collection, but for now this concept is validated in simulation only. All LDMX software is contained in the repository `ldmx-sw`. `ldmx-sw` models the TS as three *pads* with 48 *channels* each, not including backend electronics. The exact geometry of the TS can be customized within the framework[12].

3.1 `ldmx-sw`

`ldmx-sw` extends the capabilities of GEANT4, a simulation toolkit developed for particle physics use. GEANT4 models the passage of particles through matter with Monte Carlo methods[13] and returns a list of `SimHits` describing each point during the simulation. Each of these hits stores associated values of position, energy deposition, mother particle type, and timestamp[3]. The output list is produced as a ROOT-file. The ROOT framework, also established for use in particle physics, handles data storage in the form of trees, branches, and leaves[14]. Each tree contains branches, and each branch contains leaves. The tree structure emulates data folders, where the lowest-order leaves store the actual data. The data stored in a ROOT-file depends on the details of the simulation. For example, a ROOT-file with a tree of simulation events may be produced, which may contain a branch titled "TriggerPad1SimHits"—a folder with information pertaining to hits in the first trigger pad—with individual leaves containing energy deposition, step times, particle IDs, etc. Each branch is also attributed a *size* in regards to the number of items created *per event*—in this case, `SimHits` per event.

The LDMX repository is publicly available on GitHub and consists mainly of C++ code. Somewhat similar to ROOT's tree imitation, GitHub enables `ldmx-sw` to contain parallel *branches*—versions of the same software to test-implement different changes without risking bugs in the main *trunk*. During event processing, one can customize a number of input parameters to configure a simulation or event step to suit a specific process. Examples of such variables include bar lengths, minimum PE counts for hit making, and simulation run numbers which ensure randomness when generating multiple event samples. `ldmx-sw` assigns a default value to each parameter as well, which is assumed when none other is specified[12]. The following descriptions of *digi* making, clustering, and tracking assume the default parameters[1] set by each algorithm unless otherwise specified.

[1] from `ldmx-sw` v4.6.1.

3.1.1 Digitization

Bridging the gap between simulated and experimental data is done by *digitizing* simulated events, which allows simulated and real detector data to be processed in the same way. `ldmx-sw`'s TS digitization process converts the Geant4 `SimHits` to *digis* (digital signals to mimic electronics output). *Digi* production utilizes two `SimHit` variables to do this: deposited energy and time. The energy deposited in each bar is converted to a pulse with an amplitude (proportional to the PE count, which each digi is also assigned), and the time is translated to the peaking time of the pulse. These are then translated to ADC and TDC values, meant to mimic the output of the readout electronics. To model the detector response as realistically as possible, simulated electronics noise is also included in the digi translation[3].

3.1.2 Clustering Algorithm

Due to the staggering of the scintillators, single electrons often leave hits in multiple bars—one in the left column and one in the right. For this reason, hits in adjacent bars are grouped into *clusters*, where each cluster is the result of one individual electron. Included in `ldmx-sw` is an algorithm for cluster formation (see Fig. 4), which defaults to two neighboring bars per cluster, but this definition may be configured up to three.

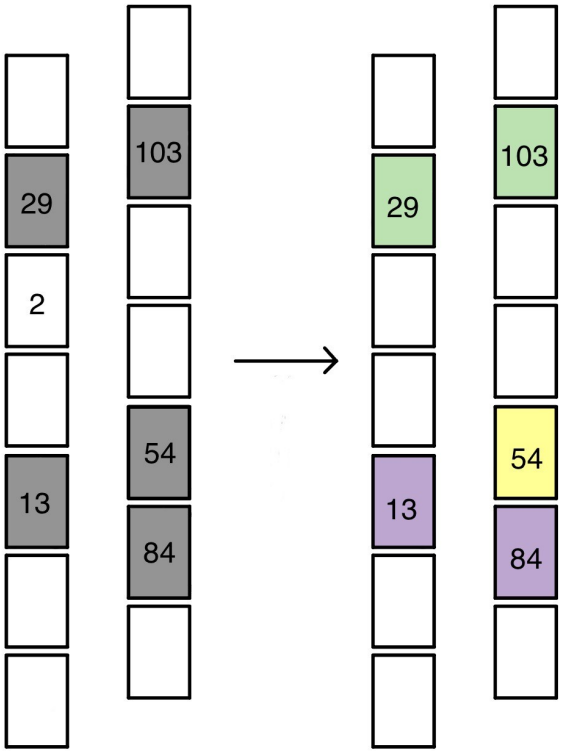


Figure 4: Seeds and hits above clustering threshold (left) to clusters (right). PE counts shown for each registered digi.

Each bar is assigned a channel ID of 0-47, starting with bar 0 at the bottom of the pad and continuing upwards until bar 47 at the top. The algorithm begins at channel 0 and searches for a seed—a PE count greater than a configurable minimum threshold (called the seed threshold) of default 30 PE. Upon detecting a seed, the algorithm steps both backwards and forwards to neighboring channels to check for hits with PE counts above a so-called (also configurable but default 0) clustering threshold. If found, they are stored together with the seed as one cluster.

The clustering algorithm, as well as all other event processors in `ldmx-sw`, must find

ways to combat naturally occurring errors. Electrons, for example, do not move through the TS perfectly. They are able to scatter backwards off the target and hit the TS pads after collision, or send stray photons to other pads. The clustering algorithm receives digi collections as input, and sets timing constraints on the digis accepted in each pad to filter out events that originate from such errors. Additional issues may arise from the electronics noise included in digis, which typically lies at levels of 1-2 PEs. One can configure the clustering threshold to filter such noise out[15].

All clusters formed from one pad are saved to a ROOT branch ("TriggerPad1Clusters"), with leaves pertaining to total PE counts, seed channels, and centroid values (among others). The centroid value c refers to the channel that the cluster is centered at.

$$c = \frac{\sum_i (\mathrm{BarID}_i \cdot \mathrm{PE}_i)}{\sum_i \mathrm{PE}_i}$$

It is calculated by the formula above as the photoelectron-weighted average of all channels included in the cluster[16]. This method assigns greater weight to those channels with higher PE counts, which centers the centroid value at the location with the "strongest" hit.

3.2 Tracking in `ldmx-sw`

In the context of the trigger scintillator, *tracking* refers to reconstructing the path of an electron from its signature in each TS pad, and the number of tracks corresponds to the number of electrons. `ldmx-sw` contains two track-forming methods[17]. Though the input of both existing algorithms differ, the overall premise consists of determining if a track candidate—a triplet of clusters from each pad—is consistent with the algorithm's track definition. When comparing track candidates, the algorithms both utilize *residual values*, which act as a measurement of how well the clusters in each candidate horizontally align. The residual r,

$$r = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(c_i - \bar{c})^2},$$

is defined as the summed squared differences of all centroid positions c_i to the (average) track centroid position $\bar{c}$. Here, $n = 3$ since there are three TS pads. For a perfectly horizontal track where all three pads return the same centroid values, the residual is zero. The residual value then increases alongside the vertical propagation between pads. The candidate with the lowest residual is then selected and stored as a track.

Both algorithms rely on the assumption that electrons will leave approximately horizontal tracks (while allowing for some limited vertical deviation), an assumption that makes them reliant on the geometry of the TS. The first of these algorithms is an implementation in firmware, and the other is a software tolerance-based approach.

3.2.1 Firmware-Based Tracking

Though the firmware tracking method is not a subject of study within this project, its implementation supports the LUT-aspect of the new algorithm. Like the software counting method, the firmware approach employs a track definition reliant on a fixed maximum propagation, only instead with a lookup table (LUT) for track pattern matching. Its performance has been thoroughly validated in 2025 slice tests[4], and as such provides a practical foundation for exploring another LUT-based approach with a new track candidate definition. No direct performance comparison is made here to the firmware tracker. Instead, it is included in this project as a conceptual reference for LUT construction.

Firmware refers to designated software-style instructions for specific pieces of hardware[18]. In the TS, tracking must be done in the real-time of the experiment, e.g. it must be *fast*. One way it meets this requirement is by storing data in the form of 12-bit integer types rather than 32-bit floats as typically done in software, which compresses them for compatibility with hardware processing. Since positions are only stored as integers, cluster centroids are saved only at exact channel IDs and exactly between channels, at whole and half- integer values. To account for the half-integers, each position is simply multiplied by two, resulting in centroid positions at all integers between 0 and 96[19].

More importantly than integer types though, is the use of a LUT when building track candidates from cluster centroids, to keep with the low runtime requirement of firmware. A LUT is a data structure that contains predefined values, eliminating the need for repeated calculations by instead linking input to output directly—similar to an answer sheet. Using a LUT essentially trades time for space. The runtime is significantly reduced, but memory is required to store this long list of predetermined values[20]. In the case of tracking, the LUT encodes permitted track candidates for each track seed c_1 (clusters in the first pad). The permitted trajectories stem from the geometric logic that an electron should only be able to propagate a maximum of ± 1 centroid unit between pads 1 and 2, and similarly so between pads 2 and 3[15]. By this definition, a centroid seed c_1 in pad 1 has three possible locations[2] in pad 2: $c_1 - 0.5$, c_1, and $c_1 + 0.5$, illustrated in Fig. 5. Each pad 2 position c_2 then similarly has three pad 3 positions c_3. This results in nine acceptable track trajectories, stored in a LUT for each c_1 input. The permitted tracks in Fig. 5 are then translated into *track patterns* as demonstrated in Fig. 6. The plot to the right in Fig. 6 displays these patterns.

$$\Delta_{12} = c_2 - c_1 \quad \text{and} \quad \Delta_{23} = c_3 - c_2.$$

Each pattern is characterized by Δ_{12}, the vertical propagation between pads 1 and 2, and Δ_{23} between 2 and 3.

[2]The 0.5 values here refer to the *physical* bar positions. In firmware, 0.5 is doubled and represented as 1.

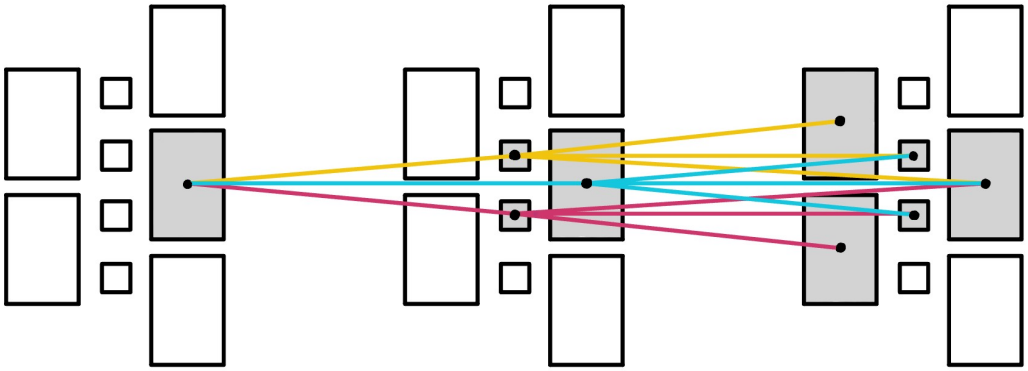


Figure 5: Track patterns permitted by the firmware tracking algorithm. The colors correspond to those in Fig. 6. The smaller squares in the middle of each pad represent centroid positions rather than physical TS components.

Tracks: (c_1, c_2, c_3)	$(\Delta_{12}, \Delta_{23})$
$(c_1,\ c_1+0.5,\ c_1+1)$	$(0.5,\ 0.5)$
$(c_1,\ c_1+0.5,\ c_1+0.5)$	$(0.5,\ 0)$
$(c_1,\ c_1+0.5,\ c_1)$	$(0.5,\ -0.5)$
$(c_1,\ c_1,\ c_1+0.5)$	$(0,\ 0.5)$
$(c_1,\ c_1,\ c_1)$	$(0,\ 0)$
$(c_1,\ c_1,\ c_1-0.5)$	$(0,\ -0.5)$
$(c_1,\ c_1-0.5,\ c_1)$	$(-0.5,\ 0.5)$
$(c_1,\ c_1-0.5,\ c_1-0.5)$	$(-0.5,\ 0)$
$(c_1,\ c_1-0.5,\ c_1-1)$	$(-0.5,\ -0.5)$

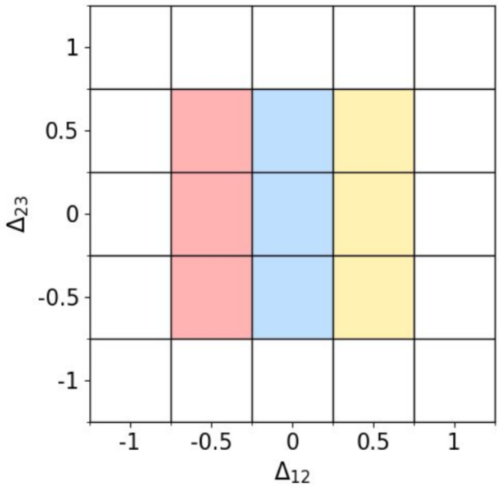


Figure 6: Left: translation of geometric tracks (c_1, c_2, c_3) from Fig. 5 to $(\Delta_{12}, \Delta_{23})$ patterns. Right: $(\Delta_{12}, \Delta_{23})$ patterns represented in heatmap-style plot. Colors correspond to those tracks shown in Fig. 5.

When referencing the LUT, the algorithm compares track candidates by residual values to select the candidate with the lowest residual as the track. After all tracks are made, duplicates are removed. By the algorithm's definition, tracks cannot share seed clusters, so this step applies primarily to clusters in the second and third pads. If two or more tracks share clusters, only the track with the lowest residual is kept.

3.2.2 Tolerance-Based Software Tracking

`ldmx-sw`'s software-based algorithm hinges on a configurable maximum deviation parameter labeled **maximum delta** Δ_{max}. It defines a maximum displacement of clusters in pads 2 and 3, c_2 and c_3, relative to a track seed c_1 in pad 1. The default Δ_{max}, assumed unless otherwise specified, is 0.75. In this algorithm, clusters are considered part of the

same track if their centroids lie within the Δ_{max} tolerance of the seed. While the firmware logic defines each step between pads relative to the previous one, the software algorithm sets no explicit constraint on the relationship between clusters in pads 2 and 3. Rather, both are independently required to lie in a specified region relative to the seed. Though this definition formally allows tracks that, for example, first propagate upwards a certain distance between the first two pads and then downwards between the second two, such counterintuitive behavior is assumed to never occur. Therefore no constraint between pads 2 and 3 is deemed necessary. Fig. 7 illustrates the permitted track patterns by the tolerance method, compared to those in the firmware method (see Fig. 6). Though Δ_{max} assumes 0.75 by default, displaying (Δ_{12}, Δ_{23}) patterns in terms of integer and half-integer values, as seen in Figs. 6 and 7, makes this not visualizable. Instead, Fig. 7 displays patterns permitted by $\Delta_{max} = 0.5$ and $\Delta_{max} = 1.0$.

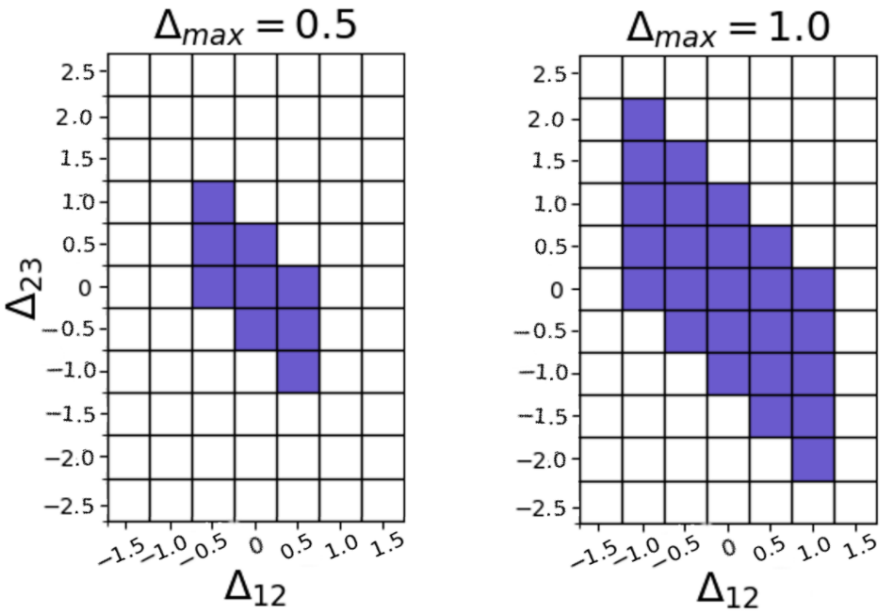


Figure 7: Track patterns permitted by tolerance algorithm, based on Δ_{12} and Δ_{23}, for different Δ_{max} values of 0.5 and 1.0.

The input collections in this method are the cluster collections from all three pads, which the algorithm iterates over to build candidate tracks. The tolerance condition is based on the cluster centroid positions, represented by floating-point values with finer spatial resolution than in firmware. Because this method iterates over all cluster collections to build all possible tracks, it is somewhat computationally demanding. Similarly to the firmware tracker though, only the track candidate with the lowest residual is actually saved.

3.3 Workflow

This thesis proposes a new tracking algorithm for electron counting in the TS, utilizing a data-driven LUT to reduce runtime. LUT-based tracking is added as an optional feature controlled by a boolean flag in the existing software tracker's skeleton. This allows switching between the traditional tolerance-based (looping) method and the LUT-based approach within the same software environment. Each possible combination of clusters is validated against the text-file LUT: if found in the table, it is accepted as a *valid* track, and if not, it is rejected.

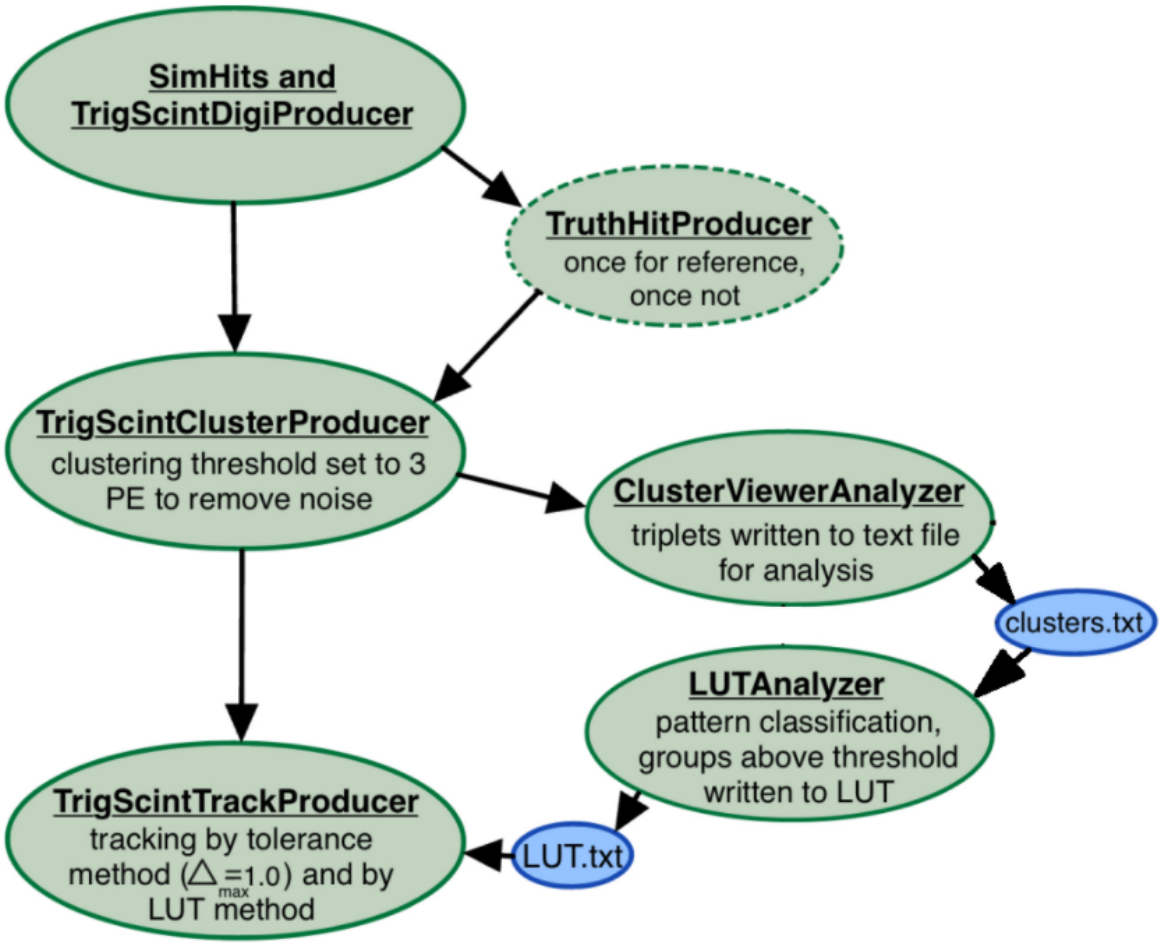


Figure 8: Event processor steps from SimHits to tracks, first performed on Truth SimHits and then repeated without **TruthHitProducer**. Cluster triplets are written to "clusters.txt" and different LUT thresholds produce different "LUT.txt" files. The files **ClusterViewerAnalyzer** and **LUTAnalyzer** were developed specifically as a result of this project[11].

A data-driven LUT is, naturally, constructed from (in this case, simulated) data, generated from detector geometry `ldmx-det-v14-8gev` in sets of 10000 events. As a proof-of-principle, truth-level information is used to study cluster patterns which correspond to real electron trajectories. Here, truth-level information refers to *uncorrupted* data entries (`SimHits`, clusters, etc. which originate from true beam electrons), verified as such by Monte Carlo information. In the realized LDMX, the STT (see Fig. 2) can provide a

similar reference since it will also record electron positions between pads 2 and 3. Here, one of `ldmx-sw`'s *event processors*—files which perform analysis operations to visualize, convert, or otherwise manipulate simulated data— **TruthHitProducer**, provides this information, by isolating hits from beam electrons using Monte Carlo information like particle ID. Events then proceed through the chain of event processors and analysis as shown in Fig. 8 for digi-making, clustering, and tolerance-based tracking[3].

Events are processed once with filtration through **TruthHitProducer**, and once without. Though the data-driven approach is validated by using **TruthHitProducer**, the final algorithm must operate using standalone unfiltered simulation data. When analyzing the cluster combinations written to the clusters text-file by **ClusterViewerAnalyzer**, shown in Fig. 8, the clustering efficiency of a simulation set can be calculated, defined as

$$\epsilon_{Cls} = \frac{n_{Cls}}{n_{ev}},$$

where n_{Cls} is the number of cluster triplets found in a set of simulated data and n_{ev} is the total number of events in that set. Comparing truth- and unfiltered patterns shows that *valid* patterns appear most frequently in the unfiltered samples, while *non-valid* combinations appear only as often as combinatorics allow.[4] This motivates implementing a pattern frequency threshold (a *LUT Threshold*): only patterns with a frequency above this threshold are written to the LUT. **LUTAnalyzer** contains the LUT threshold as a configurable parameter, and sorts triplets into groups characterized by pattern coordinates $(\Delta_{12}, \Delta_{23})$, as done in Figs. 6 and 7. A pattern's frequency is then calculated as the fraction of the total number of cluster combinations which are of that pattern.

$$\text{Frequency fraction} = \frac{n_{group}}{n_{Cls}}.$$

Different LUT thresholds are evaluated to determine which value most closely matches the truth-level results. The pattern-LUT tracking performance is evaluated based on tracking efficiency ϵ_{Trk} and fake rate r_{fake}:

$$\epsilon_{Trk} = \frac{n_{Trk}}{n_{Cls}} \quad \text{and} \quad r_{fake} = \frac{n_{fakes}}{n_{Cls}},$$

where n_{fakes} is the number of fake tracks and n_{Trk} represents the total number of tracks created. If two or more tracks are created from the same single-electron event, the spare track is classified as a *fake*. Each event contains one electron and should therefore only produce one track. A perfect algorithm would achieve $\epsilon_{Trk} = 1.0$ and $r_{fake} = 0.0$. Here, we compare the performance of LUT-method tracking for different LUT thresholds with the goal of matching that of the tolerance tracker.

[3] All steps use default parameters unless otherwise specified.

[4] Patterns which are likely to be true even in the unfiltered set, due to their frequency, are labeled *valid*, while infrequent, illogical combinations are labeled *non-valid*.

Amplitude-weighted centroid values are not LUT-compatible. A practicable LUT requires exact values: integer and half-integer. Though weighting centroid positions improves spatial resolution, it is not crucial to the TS. It is primarily applicable to "charge-sharing" detectors to account for overlapping signals[15]. The reflective film around each scintillator bar in LDMX's TS isolates the channels from each other, eliminating such effects. Amplitude weighting is therefore also made optional in clustering, partially for LUT-writing and partially to evaluate its overall importance in tracking.

4 Results and Discussion

4.1 Impact of Amplitude Weighting

The effect of amplitude weighting as a boolean variable on tolerance tracking is briefly evaluated independently of LUT-method tracking. Disallowing PE-weighted centroid values has no effect on the number of clusters formed in a pad, and it also has no effect on the number of hits within each cluster. Its effect on tolerance tracking applies to the number of tracks created, and also their patterns (measured by residual values). Table 1 and Fig. 9 represent a set n_{ev}=10000, passed through **TruthHitProducer**, with a clustering efficiency ϵ_{Cls}=97.31%.

Amplitude Weighting	n_{Trk}	ϵ_{Trk}
`True`	9643	99.10%
`False`	9566	98.30 %

Table 1: Tracking with amplitude-weighted vs. non-weighted cluster centroid values, for n_{ev}=10000 and $\epsilon_{Cls} = 97.31\%$. $\Delta_{max} = 1$; truth tracks only.

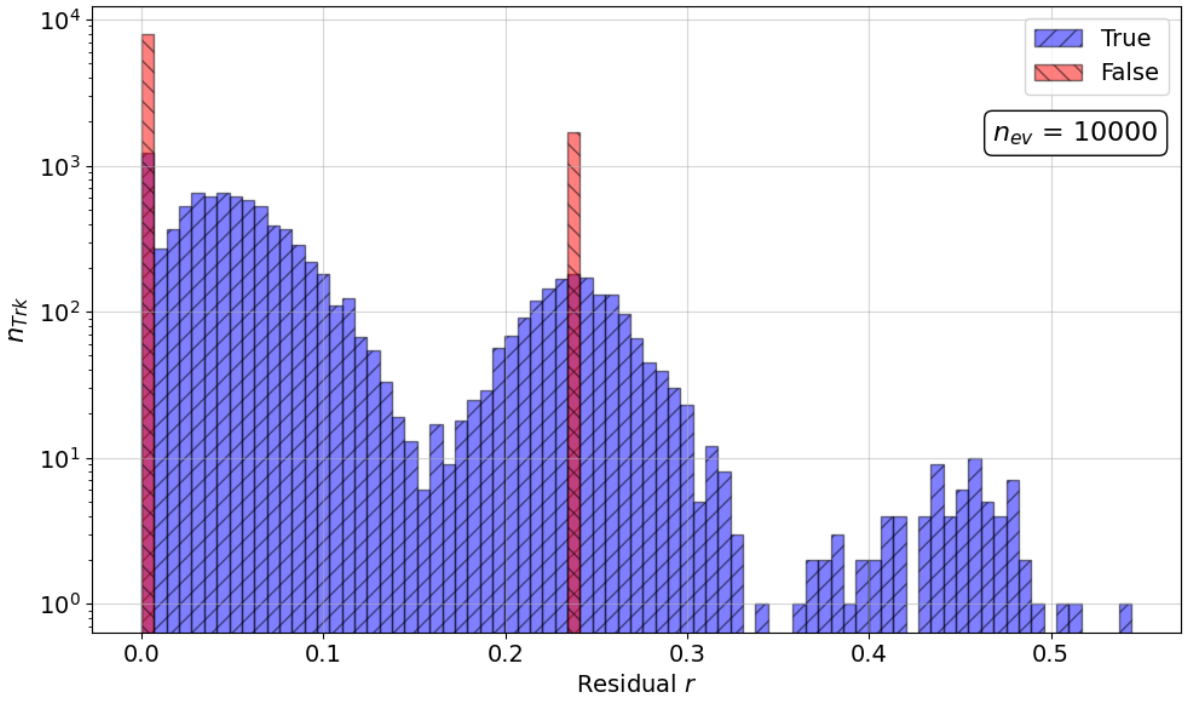


Figure 9: Residual values r of tracks created when amplitude weighting flag is set to `True` and `False`. Truth tracks only.

The wide distribution in r-values when PE-weighting (Fig. 9) is exactly as expected, stemming from the slight variations in centroid positions that originate from individual PE differences. When disallowing weighting, the spread in residuals is reduced to two values:

$r = 0$ and $r = 0.2357$. The residuals become fixed, which originates from now-discrete track patterns—in this case $(0, 0)$ and $(0, 0.5)$, respectively. Table 1 displays a slight decrease (0.8 percentage points) in tracking efficiency ϵ_{Trk} due to removal of amplitude weighting, but this is deemed acceptable for the tradeoff of enabling LUT-matching. For all further results (including tracking by the tolerance tracker), centroid values are not amplitude weighted.

4.2 Clusters Analysis

Examining the cluster triplets written to "clusters.txt" (step from Fig. 8) results in the finding of events with *extra* clusters. This refers to events where more than one cluster was detected in a single pad, observed in the unfiltered simulated events in Fig. 11. In the truth-level case (Fig. 10), an averaged 98.63% of events produce only one cluster per pad (compared to 94.01% in the unfiltered case). This leads to the conclusion that beam electrons are only capable of leaving one true cluster in each pad, while extras may be attributed to scattering or recoil. In both the truth-filtered and unfiltered cases, a noticeable number of events produce no clusters, especially visible in pad 3. A likely explanation is that those electrons simply traverse beyond the bounds of the TS modules, and are therefore not detected. While the second pad is positioned immediately following the first, the distance between pads 2 and 3 is significantly larger (see Fig. 3). This would further suggest that the electrons propagate beyond the edges of the third pad, resulting in no clusters. The STT, positioned between pads 2 and 3, likely also contributes to this effect by causing additional scattering. Hence, 4.12% of electrons have propagated beyond the module boundaries by the time they reach the third pad.

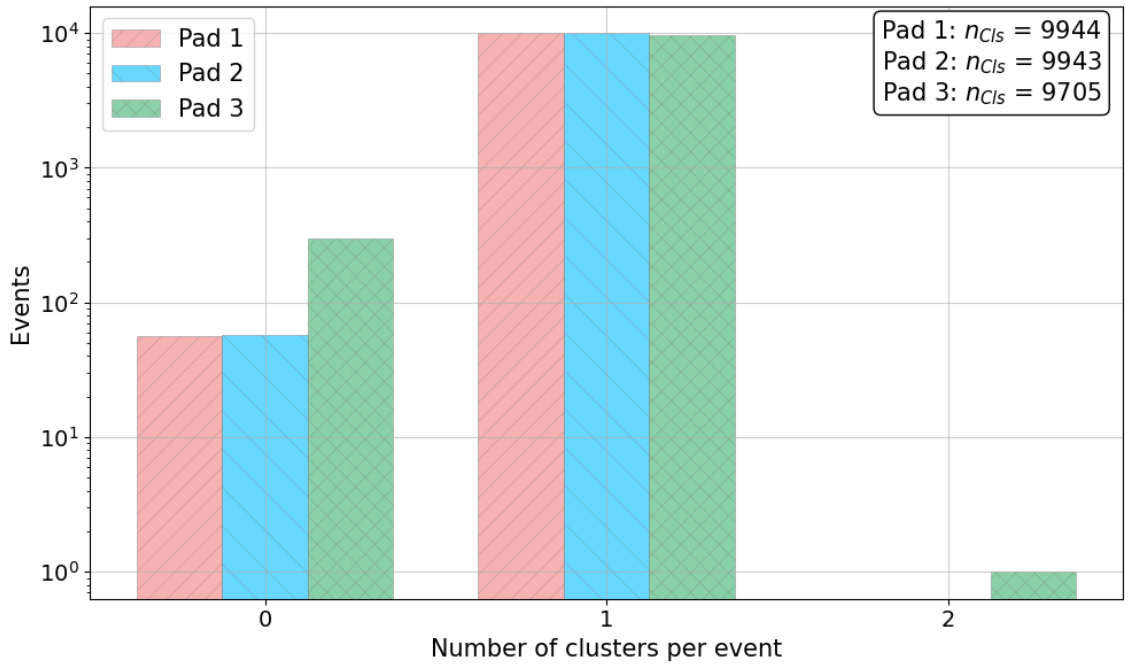


Figure 10: Number of clusters created per event (truth clusters only).

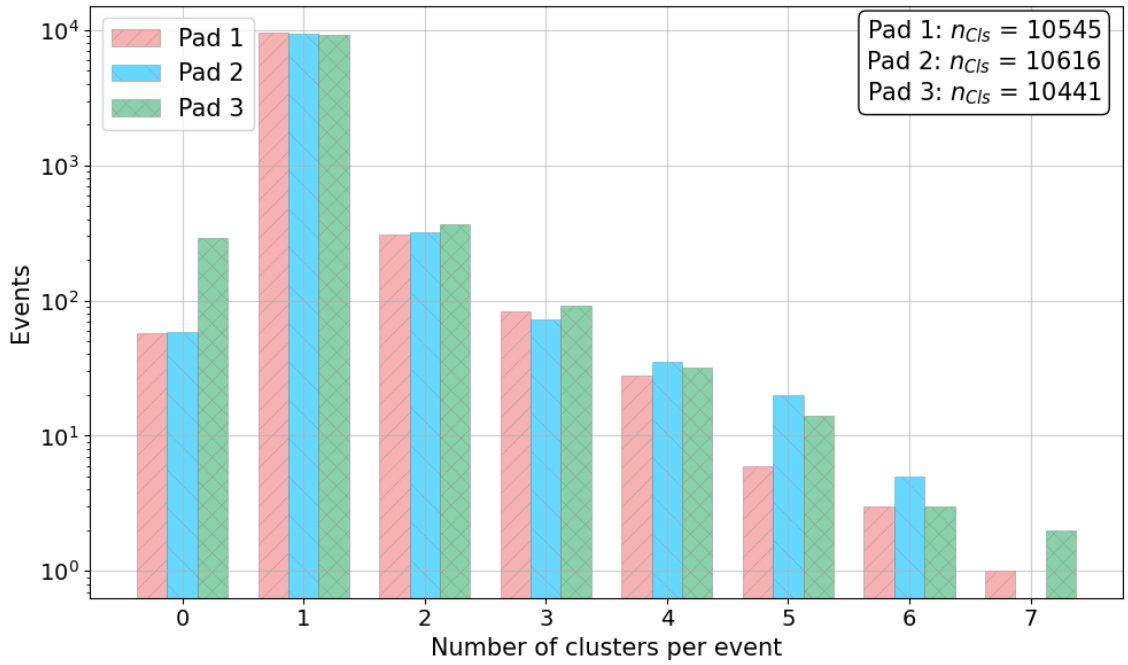
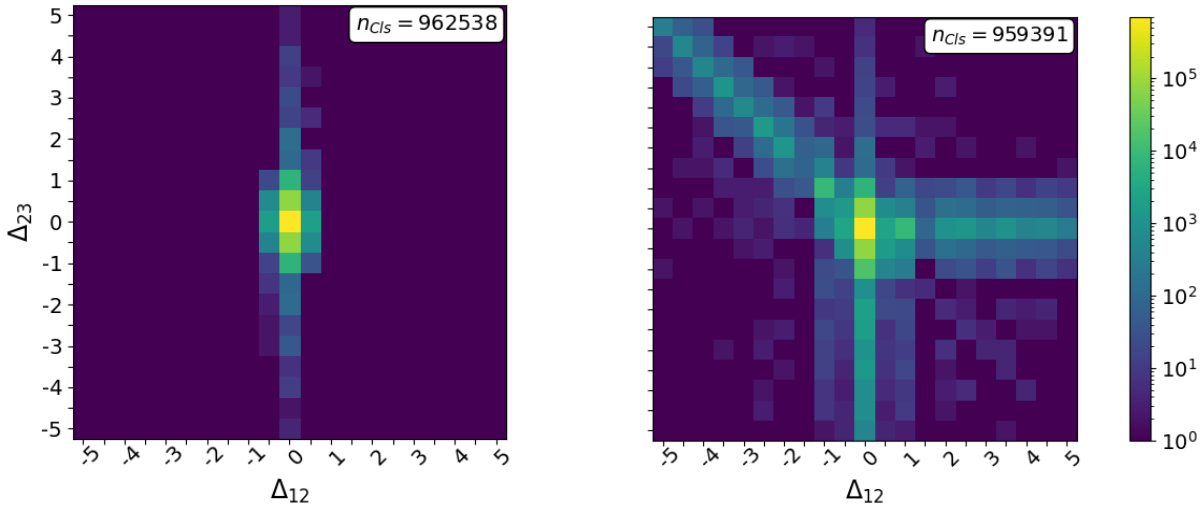


Figure 11: Number of clusters created per event (all clusters included).

The single event in Fig. 10 with two clusters in pad 3 is curious. The event constituents are $c_1 = 17.5$, $c_2 = 17.5$, and $c_3 = [18, 22.5]$, eliminating any three-barred cluster explanations. That **TruthHitProducer** accepted two (same-pad) clusters from the same event as *true*—incident with the beam timing and of correct particle ID—seems theoretically impossible. Further investigation may be required here.



(a) Including truth clusters only.

(b) Including all clusters.

Figure 12: Track patterns as seen in simulation, based on Δ_{12} and Δ_{23}, the vertical propagation between pads 1 and 2, and pads 2 and 3, respectively.

Translating the cluster triplets in the produced text file, with the help of **ClusterViewerAnalyzer**, to (Δ_{12}, Δ_{23}) patterns results in the distributions seen in Figs. 12a and 12b. Though n_{Cls} is larger in the truth-filtered case, this is only due to the number of

events also being larger for this set—it does not imply that isolating truth-events results in *more* clusters. n_{Cls} is included merely as a reference to the scale of the simulated sets of events. The truth-filtered (ϵ_{Cls}=0.9581) and unfiltered (ϵ_{Cls}=0.9588) data both exhibit (on average) 4.16% of events not resulting in cluster triplets (potential tracks), further indicating that these events likely propagate outside of the TS.

Figs. 12a and 12b both clearly display a strong concentration of combinations at pattern $(\Delta_{12}, \Delta_{23}) = (0, 0)$, implying that most electrons traverse through the TS with perfectly horizontal paths. Considering first only the truth clusters, one immediately observes the significantly broader Δ_{23} distribution, in comparison to that in Δ_{12}. Electrons are able to traverse greater distances between pads 2 and 3 as they positioned farther apart than pads 1 and 2. The first two pads are positioned so closely together that real electron patterns are unlikely to have a Δ_{12} greater than 0.5.

The set of *all* cluster combinations (Fig. 12b) contains significantly more patterns than visible in the truth-isolated set. Since these additional combinations are not included amongst the truth-level patterns, we conclude that they must be non-valid and not from real electrons. However, they occur with overall lower frequencies than the true patterns, which are still strongly enhanced in the set containing all events. The true trajectories dominate the cluster patterns strongly enough to prompt introducing a frequency threshold, which could suppress non-valid patterns without requiring **TruthHitProducer** filtering.

Fig. 12b exhibits three highlighted regions of non-valid propagation patterns seeming to stretch out from the center, $(\Delta_{12}, \Delta_{23}) = (0, 0)$, in the downward, rightward, and upper left diagonal directions. These "arms" are absent among the truth-level patterns, and they originate from the implementation of **ClusterViewerAnalyzer**. In the case of extra clusters, the analyzer is designed to select the cluster with the lowest centroid value, resulting in patterns which would (in the case of beam electron selection), not otherwise have occurred. Though one could propose redefining **ClusterViewerAnalyzer** to select the cluster which results in the most horizontal trajectory, this is precisely the distance-based logic that the existing tolerance tracker utilizes: ineffective in the case of a misalignment. This once again incentivizes a method of filtration that is not reliant on fixed bar positions, such as a frequency threshold.

The frequency threshold filtration method is investigated by grouping cluster triplets according to their propagation patterns (Δ_{12}, Δ_{23}). Organizing the data in this way results in Figs. 13 and 14, both visibly displaying the dominant pattern (0,0), also seen as strongly concentrated in Figs. 12a and 12b. While the truth-level cluster triplets result in 79 total pattern groups, the unfiltered combinations contain 1413 distinct patterns, meaning most do not originate from real beam electrons. The greater variety in patterns with low frequency among the unfiltered combinations, seen in Fig. 14, results in a smoother distribution, as compared to the more rigid Fig. 13.

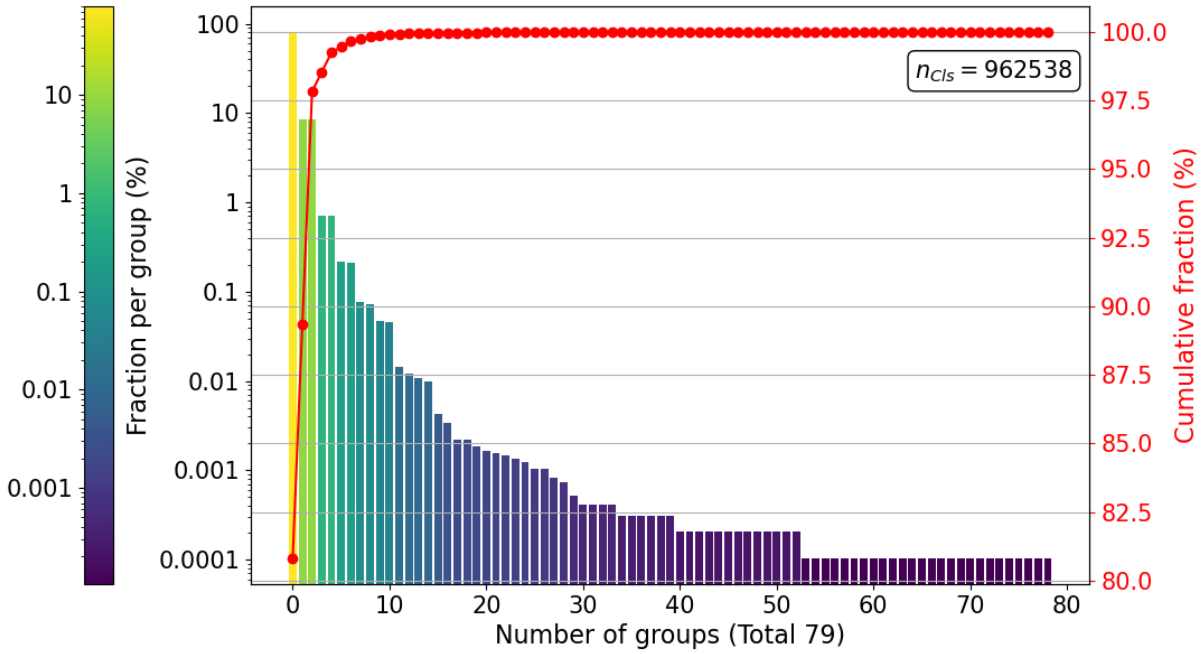


Figure 13: Groups of truth track candidates based on propagation patterns, sorted individually by size (percentage of total n_{Cls}). Cumulative sum also shown. Each bar represents one pattern group.

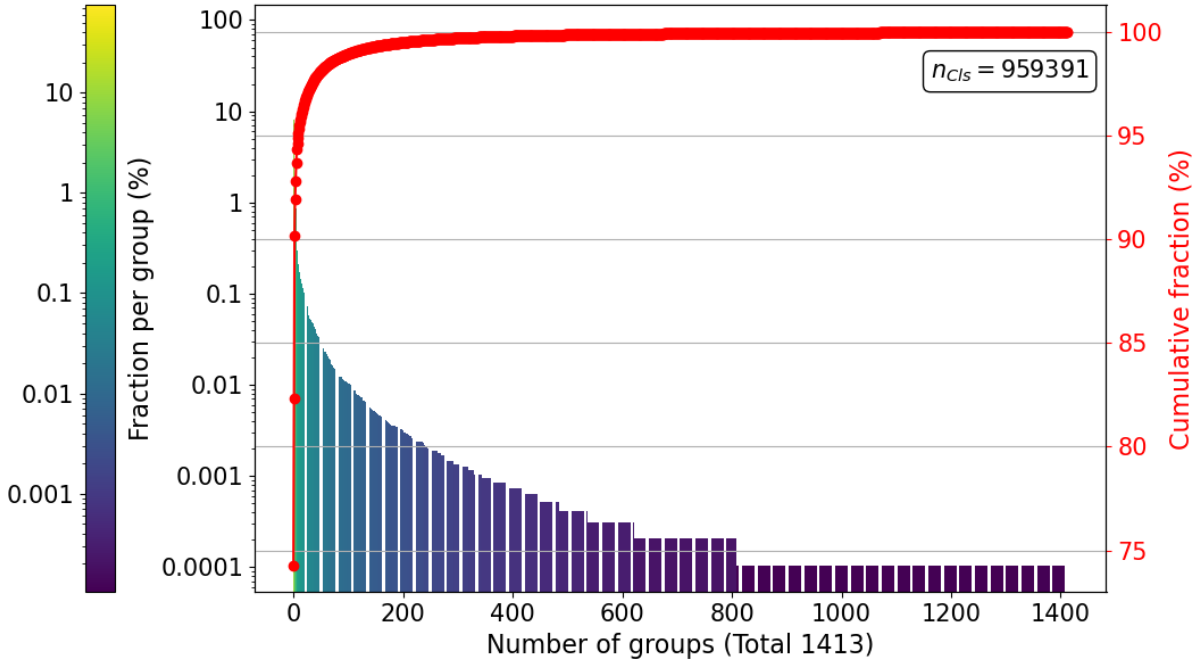


Figure 14: Groups of all track candidates based on propagation patterns, sorted individually by size (percentage of total n_{Cls}). Cumulative sum also shown. Each bar represents one pattern group.

4.3 Evaluation of Pattern-LUT Tracking

Each pattern group is then assigned a frequency proportional to the number of cluster combinations with that pattern. Different LUT thresholds are tested to evaluate the feasibility of a frequency threshold and if so, where the optimal threshold lies.

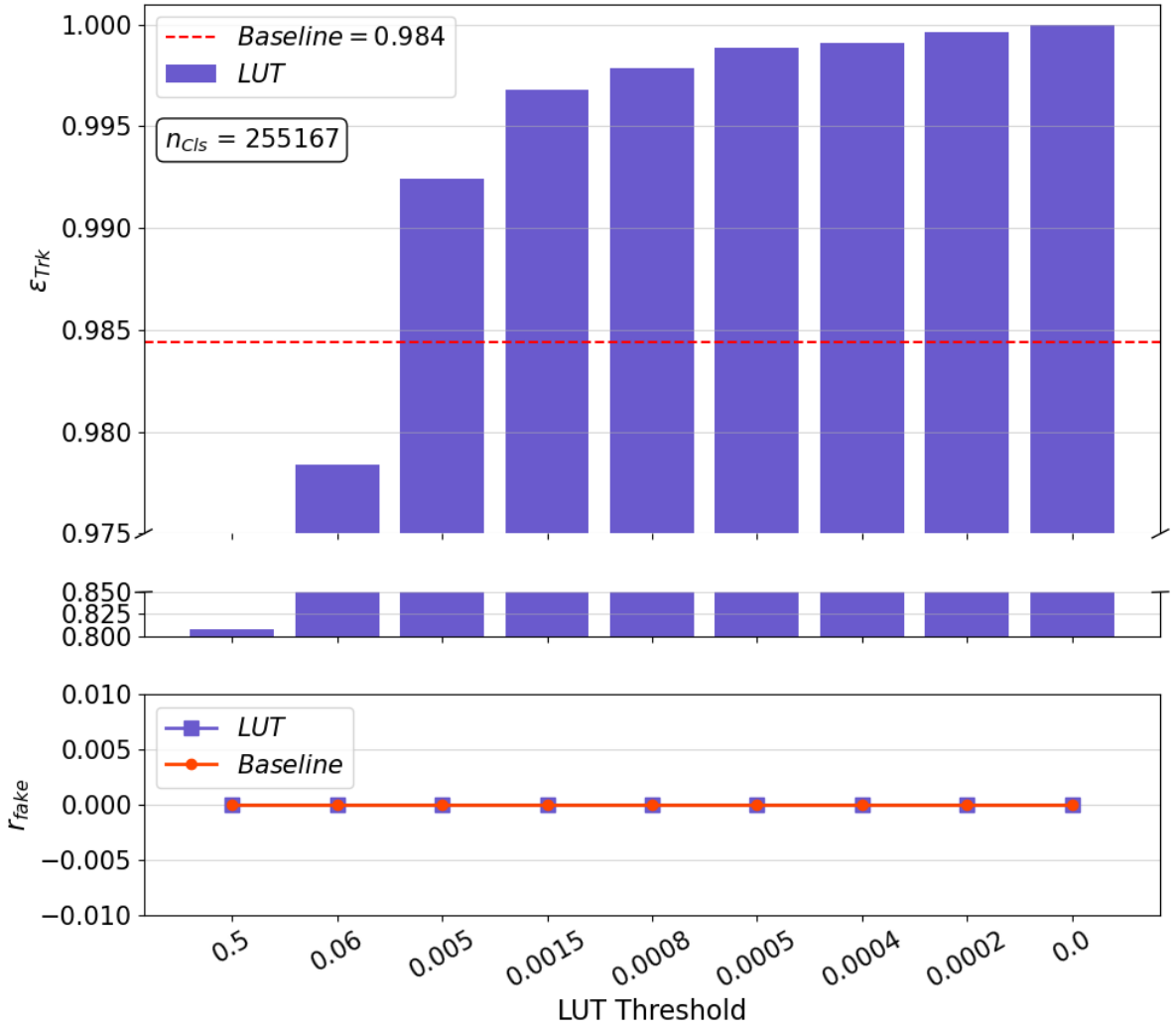


Figure 15: Tracking efficiency ϵ_{Trk} (upper panel) and fake rate r_{fake} (lower panel) for pattern-LUT tracking method, shown for different LUT thresholds (blue). ϵ_{Trk} (dashed red line) and r_{fake} (red points) also displayed for tolerance-method tracking (labeled "Baseline"). Truth hits only.

n_{Cls} in the truth sample (Fig. 15) is considerably smaller than the unfiltered sample (Fig. 16), though the fake rate r_{fake} of zero is not correlated to the size of the data set. Rather, it is a product of the truth-level combinations written to the LUT, resulting in only real tracks. Accepting all true cluster combinations as patterns (LUT threshold=0.0), when they are all confirmed to be true, results in $\epsilon_{Trk} = 1.0$ and r_{fake}=0.0, confirming that a data-driven LUT is a viable method for tracking. We now observe the frequency threshold results in the unfiltered sample, shown in Fig. 16.

LUT Threshold	ϵ_{Trk} /%	r_{fake} /%
0.0	99.73	0.2604
0.0002	99.72	0.09464
0.0004	99.65	0.01324
0.0005	99.60	0.004065
0.0008	99.37	0.001668
0.0015	99.03	0.001563
0.005	98.15	0.001355
Tolerance algorithm	98.00	0.0007296
0.06	94.57	0.0007296
0.5	76.81	0.0006254

Table 2: Tracking efficiency ϵ_{Trk} and fake rate r_{fake} as found for different LUT thresholds (and tolerance algorithm in red), as seen in Fig. 16 for all hits included.

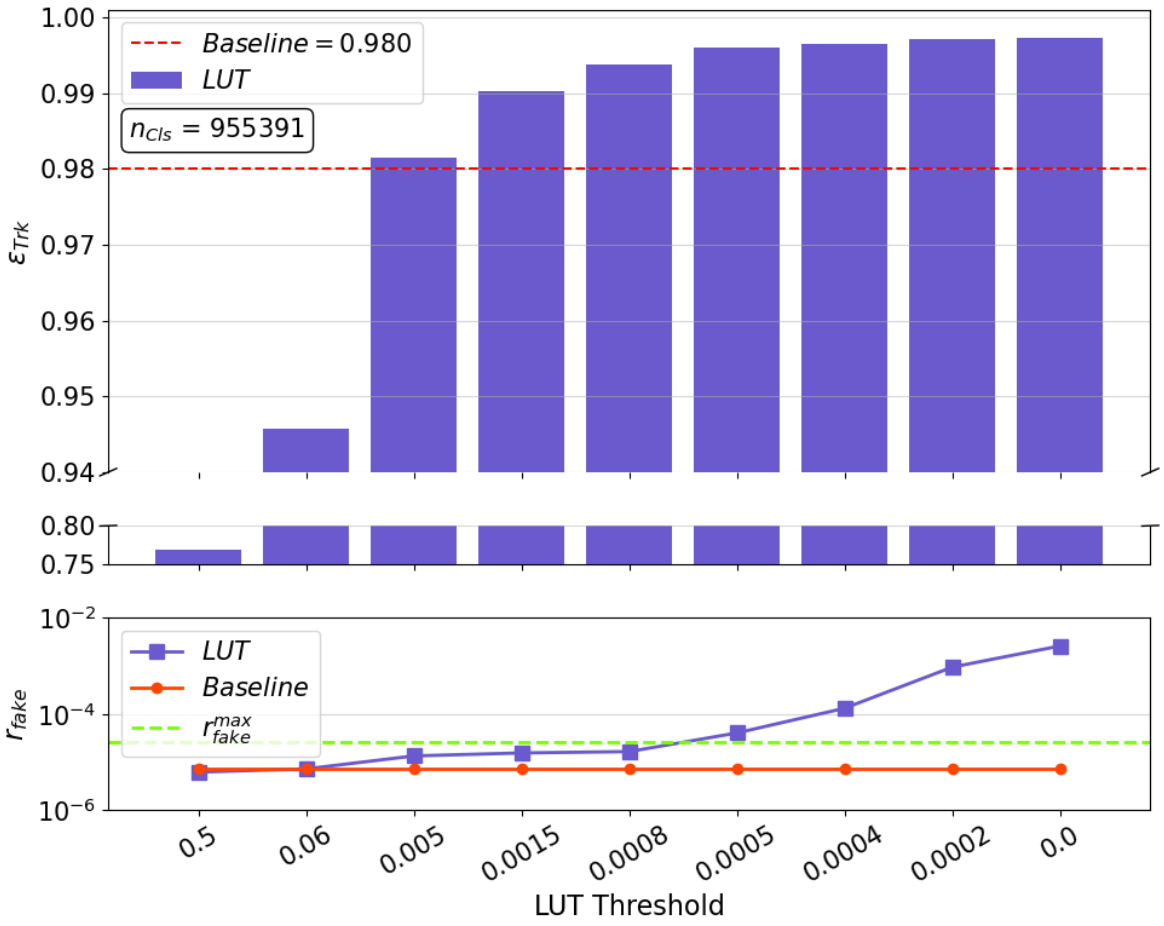


Figure 16: Tracking efficiency ϵ_{Trk} (upper panel) and fake rate r_{fake} (lower panel) for pattern-LUT tracking method, tested for different LUT thresholds (blue). ϵ_{Trk} (dashed red line) and r_{fake} (red points) also displayed for tolerance-method tracking (labeled "Baseline"). All hits included. The maximum permitted fake rate r_{fake}^{max} labeled (lower panel) in green.

The efficiency of the tolerance tracker is relatively consistent both with ($\epsilon_{Trk} = 98.40\%$) and without ($\epsilon_{Trk} = 98.00\%$) access to truth-level information, with a fake rate r_{fake}= 0.0007296% in the latter case. Its performance is not severely affected by lack of truth-information; it performs well (in the ideal geometric case). Examining the pattern-LUT method, one observes that the ϵ_{Trk} increases together with r_{fake}. The optimal LUT threshold can then be selected according to different criteria: matching the tolerance algorithm's efficiency, matching its fake rate, or maximizing efficiency within the fake rate constraints by the TS performance requirements. The maximum fake rate r_{fake}^{max} can be derived from the beam pulse frequency (37.14 MHz) and the TS maximum overcounting rate (1 kHz) as r_{fake}^{max}=0.002693%. Given these criteria, the optimal LUT threshold candidates are:

1. A LUT threshold of 0.005 to match the tolerance tracker's ϵ_{Trk}=98.00%. LUT-method achieves ϵ_{Trk}=98.15% with r_{fake}=0.001355%.

2. A LUT threshold of 0.06 to match the LUT-method fake rate ($r_{fake} = 0.0007296\%$) to the tolerance tracker's $r_{fake} = 0.0007296\%$. Though this selection matches the fake rate exactly, ϵ_{Trk}=94.57% is lower than the tolerance method.

3. A LUT threshold of 0.0008 achieves the highest ϵ_{Trk}=99.37%, for r_{fake}=0.001668%. This is the highest possible efficiency with a fake rate still acceptable by the TS performance requirements constraints.

One can evaluate the distribution of residual values as well.

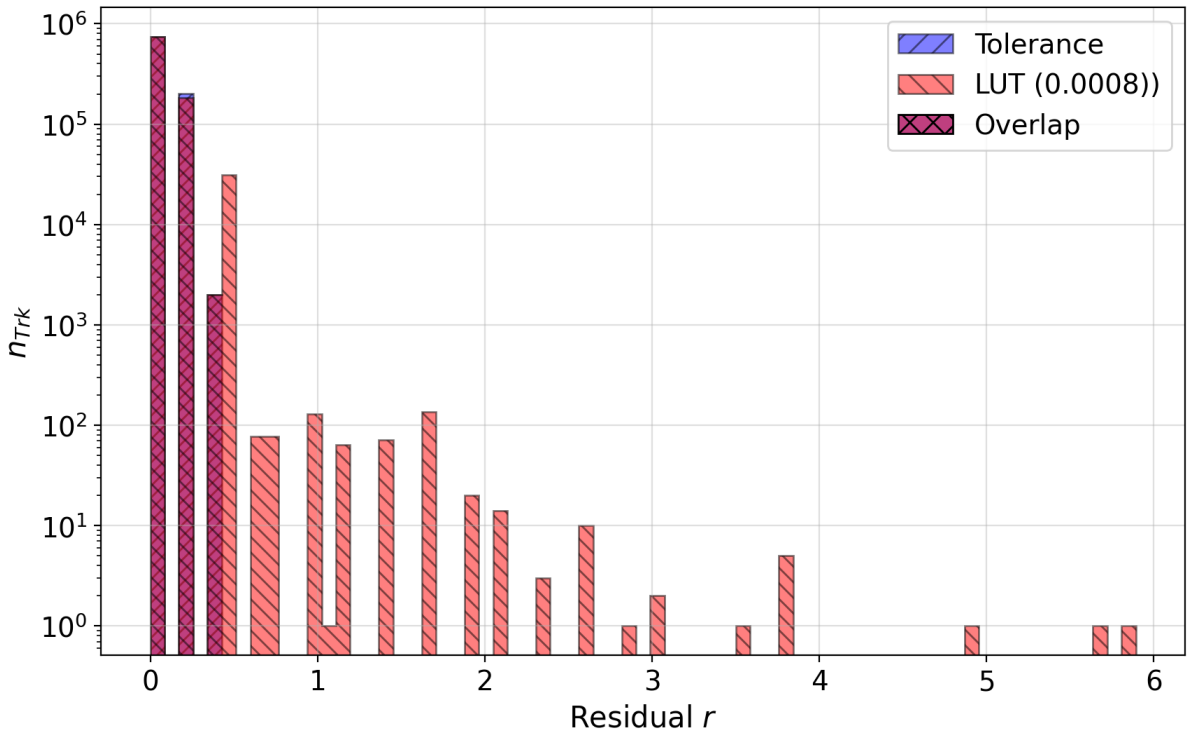


Figure 17: Residual values r of tracks created in tolerance tracking and LUT tracking (LUT threshold of 0.0008). All tracks included. Overlap indicates both tolerance and LUT method.

The track residuals for a LUT threshold of 0.0008 are as displayed in Fig. 17, including a comparison to those of the tolerance-method. Naturally, the tolerance-based algorithm creates tracks with residuals no greater than $r = 0.471$, the greatest possible residual value for a track for the greatest possible propagation (Δ_{max}=1.0) between pads 1 and 3. While tracks from the tolerance method are concentrated to a fixed range of residuals, those of the pattern-based LUT method are spread over a wider range. The higher fake rate among the pattern-based tracks suggests that those with higher residuals are likely fakes, since such tracks are absent among the tolerance sample. The broader range of LUT-method track residuals is therefore likely a result of fake tracks and coincidental combinations of scattered extra clusters.

A LUT threshold of 0.0008 results in the highest possible tracking efficiency (99.37%) within the constraints provided by the TS performance requirements. The optimal LUT threshold depends on balancing efficiency and fake rate, but both 0.06 and 0.005 are also demonstrated as viable threshold candidates.

5 Conclusion and Outlook

This study has thus proven that a pattern-based, data-driven LUT tracking method is sufficiently robust for track reconstruction without (excessively) over-counting electrons and should, in theory, still perform well in case of detector misalignment. Furthermore, a feasible approach to constructing such a tracking algorithm, that can be reproduced under real experimental conditions, has been demonstrated. Isolating valid track patterns with a frequency threshold has been proven effective.

Beginning with the observed clusters, investigation of the single extra true cluster in pad 3 (seen in Fig. 10) is encouraged. Under the assumption that both clusters from the same single-electron event originated from beam electrons (according to Monte Carlo information), and that both clusters were incident with the beam (according to timing constraints), then there seems to be no physical explanation for one beam electron to cross a module in two different locations at the same time. The origin of this phenomenon should be determined by studying this event in detail to rule out any issues in **TruthHitProducer** or **TrigScintClusterProducer**. The anomaly occurred in event 7098 for a simulated set of 10000 events with run number `p.run_number`=2 (see [11] for details).

Regarding the pattern-based tracking method, future continuation of this project should further evaluate LUT thresholds between 0.005 and 0.06. Though the data-driven algorithm derived here is able to achieve the required fake rate, it would be beneficial to reduce it further to match the existing tolerance tracker. The steps of event processors and analysis described in this paper should then simply be repeated simply for more LUT threshold values. To further refine the threshold, one should repeat this study with more events n_{ev}. Greater sets of events may more cleanly display a clearer cutoff between frequent, valid patterns and uncommon combinations. One could also propose investigating a frequency cut on residuals to reduce the fake rate, though this would require detailed analysis in determining such a definition, similarly to the method described in this report.

While this study contained only single-electron events, future investigations should research data-driven tracking in the case of multiple electrons per event, since such cases will occur in LDMX. In such a study, a LUT would be derived from single-electron events, in the same way as described here, and referenced when reconstructing tracks from simulated events with two or more electrons each. One would then aim to produce two or more tracks from each event, the same number of tracks as contained in the event.

Though the primary objective of a data-driven tracking algorithm is to naturally accommodate a detector displacement, the detector geometry utilized in this study assumed perfectly aligned TS modules. The performance of the method derived in this paper should be further investigated to confirm its consistency with a detector misalignment. To do this, one would define a distorted detector geometry, such as displacing one pad a certain distance off-aligned, in GEANT4 and proceed with the analysis steps described here (as

shown in Fig. 8). Files of `SimHits` would then be produced with the skewed geometry already encoded in them. If this were to be done, one should, theoretically, achieve a similar performance (in terms of tracking efficiency and fake rate) by using LUT-method tracking. In comparison, the distance-based tolerance tracker would likely form tracks from non-valid cluster combinations, resulting in a decreased tracking efficiency (depending on the offset length defined in the geometry). Future work should then also observe the cluster triplets text-list to confirm that the misalignment can be learned from the (simulated) data. These steps should be repeated for various misalignment types, depending on the size of the displacement and the number of displaced bars.

Offsetting individual bars generates another scenario of interest. In the case of a *global* misalignment—uniform displacement of a whole module—pattern sorting, we predict, performs within the TS performance requirements. Despite this, a pattern-based approach would be obsolete in the presence of individual bar displacements (*local* displacements), since classification by track patterns relies on bars within a module maintaining uniform positions relative to each other. To account for nuance between individual bar IDs, a detector geometry should, once again, be defined in GEANT4. The methodology as described in Fig. 8 should be repeated, only with a modification to **LUTAnalyzer**. Rather than sorting by pattern, it should classify cluster combinations individually by exact bar ID combinations to construct a LUT. The frequency threshold definition would then likely require repeated optimization studies to determine the ideal cutoff. One must then ensure that deriving a LUT with this method yields an appropriate tracking efficiency and fake rate within the TS performance requirements.

The LUT would certainly become much longer in this scenario, due to the increased number of possible valid cluster combinations. Though the firmware tracker demonstrates the sufficient speed at which the LUT operates at, a significantly longer LUT may result in the total memory outweighing its "speedy" computational advantage. Further investigations should attempt to locate the point at which this tradeoff is no longer quick enough to suit firmware requirements.

References

[1] Knut Lundmark, "Über die Bestimmung der Entfernungen, Dimensionen, Massen und Dichtigkeit fur die nächstgelegenen anagalacktischen Sternsysteme.", Meddelanden från Lunds Astronomiska Observatorium Serie I **125**, pp. 1–13 (1930), https://inspirehep.net/files/7ee66385e7792b3c2890f67e3df88e16.

[2] Marco Cirelli, Alessandro Strumia, and Jure Zupan, "Dark matter", SciPost Physics Reviews, pp. 11–33, 84–98 (2026) 10.21468/scipostphysrev.1, http://dx.doi.org/10.21468/SciPostPhysRev.1.

[3] Torsten Åkesson et. al., *LDMX – The Light Dark Matter eXperiment*, 2025, arXiv:2508.11833 [hep-ex], https://arxiv.org/abs/2508.11833.

[4] Elizabeth Berzin et. al., "Measurement of the LCLS-II dark current using the LDMX Trigger Scintillator Prototype", (2026), arXiv:2601.08090 [hep-ex].

[5] Keith M. Ashman, "Dark Matter in Galaxies", Astronomical Society of the Pacific (1992).

[6] Shao-Ping Li, *Observability of CMB spectrum distortions from dark matter annihilation*, 2024, arXiv:2402.16708 [hep-ph], https://arxiv.org/abs/2402.16708.

[7] C. Skordis, D. F. Mota, P. G. Ferreira, and C. Bœhm, "Large Scale Structure in Bekenstein's Theory of Relativistic Modified Newtonian Dynamics", Physical Review Letters **96** (2006) 10.1103/physrevlett.96.011301, http://dx.doi.org/10.1103/PhysRevLett.96.011301.

[8] Lars Bergström, "Dark Matter Candidates", New Journal of Physics **11** (2009) 10.1088/1367-2630/11/10/105006, http://dx.doi.org/10.1088/1367-2630/11/10/105006.

[9] Jim Alexander et. al., *Dark Sectors 2016 Workshop: Community Report*, 2016, arXiv:1608.08632 [hep-ph], https://arxiv.org/abs/1608.08632.

[10] Torsten Åkesson et. al., "Photon-rejection power of the Light Dark Matter eXperiment in an 8 GeV beam", Journal of High Energy Physics **2020**, pp. 003 (2020) 10.1007/JHEP04(2020)003, arXiv:1912.05535 [physics.ins-det].

[11] Lucia Kvarnström, *Pattern-LUT Thesis Repository*, https://github.com/kvarnla/PatternLUT-thesis/tree/main, 2026, (Accessed 06/07/2026).

[12] LDMX Collaboration, *LDMX Software Documentation*, https://ldmx-software.github.io/using/analysis/ldmx-sw.html, 2026, (Accessed 05/01/2026).

[13] John Ballantyne, *Simulation of dark bremsstrahlung in Geant4*, 2023, 10.17632/3PXFRZ8CMN.1, https://data.mendeley.com/datasets/3pxfrz8cmn/1.

[14] Rene Brun and Fons Rademakers, "ROOT – An Object Oriented Data Analysis Framework", in Proceedings AIHENP'96 Workshop, Lausanne, Vol. 389, Nuclear Instruments and Methods in Physics Research Section A (Sept. 1997), pp. 81–86, 10.1016/S0168-9002(97)00048-X.

[15] Lene Kristian Bryngemark, Private correspondence, 2026.

[16] LDMX Collaboration, *TrigScintClusterProducer.cxx in the ldmx-sw repository*, https://github.com/LDMX-Software/ldmx-sw/blob/trunk/TrigScint/src/TrigScint/TrigScintClusterProducer.cxx, 2026, (Accessed 05/13/2026).

[17] LDMX Collaboration, *Ldmx-sw*, https://github.com/LDMX-Software/ldmx-sw, 2026, (Accessed 05/13/2026).

[18] Ascher Opler, "Fourth-Generation Software", Datamation **13**, pp. 22–24 (1967).

[19] LDMX Collaboration, *Firmware source files in the ldmx-sw repository*, https://github.com/LDMX-Software/ldmx-sw/tree/trunk/TrigScint/src/TrigScint/Firmware, 2026, (Accessed 04/29/2026).

[20] BLT Inc., *What is a Lookup Table (LUT)?*, https://bltinc.com/2025/08/20/what-is-a-lookup-table-lut/ (Accessed 05/05/2026).