ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Mánek**	Jméno: **Petr**	Osobní číslo: **420263**	

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Rozpoznávání ionizujících částic pomocí hybridních aktivních pixelových detektorů

Název diplomové práce anglicky:

Machine learning approach to ionizing particle recognition using hybrid active pixel detectors

Pokyny pro vypracování:

Hybrid active pixel detectors (Timepix, Timepix3) record 2D projections of ionizing radiation particles in a frame-based (comercial camera like) or event-driven readout scheme. A basic categorization of such tracks was started by Holy [1]. However, recent data taken in Space and in particle accelerator facilities have shown that the defined categories are not sufficient.
Furthermore, advances in detector technology have lead to Timepix3?s improved features which can even reconstruct the 3D particle trajectory [2].
The goal of the thesis is to utilize methods of computer vision for feature detection and machine learning for particle species identification to achieve a particle species separation from the 2D track projections recorded by the Timepix. Both approaches have not been applied to such data before.
Referring to previous research of used methodology, the thesis will:
- define appropriate model for primary detector events; such model shall encompass both intrinsic (i. e. particle species) and extrinsic parameters (i. e. particle trajectory),
- propose and implement a robust track detection method for 2D Time-over-Threshold Timepix frames (e. g. Hough transform or a RANSAC variant [3,4]), optionally generalize such method for telescopic sensor configuration or combined Time-over-Threshold and Time-of-Arrival measurements obtained from Timepix3,
- select and define various features in detected tracks and analyze their properties on experimental or simulated ground truth data by means of Principal Component Analysis or a similar algorithm,
- propose and implement a machine learning algorithm for robust fitting of the defined model from detected 2D tracks, utilizing e. g. Linear Classifiers (or their combination),
Decision Trees, Markov Chain Monte Carlo or other suitable method,
- evaluate accuracy of such algorithm on experimental or simulated ground truth data using correctly detected 2D tracks and the output of the proposed detection method; discuss the influence of detection error on fitting error.
The result of the work will be well-documented implementation of the proposed techniques, preferably distributed as a reusable library package with a demonstration application capable of showcasing various features of the work in a graphical user interface. For training of artificial intelligence algorithms, Timepix and Timepix3 data from various experimental applications will be made available.

Seznam doporučené literatury:

[1] T. Holy, E. Heijne, J. Jakubek, S. Pospisil, J. Uher, Z. Vykydal, Pattern recognition of tracks induced by individual quanta of ionizing radiation in Medipix2 silicon detector, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume 591, Issue 1, 2008,
Pages 287-290, ISSN 0168-9002, https://doi.org/10.1016/j.nima.2008.03.074.
(http://www.sciencedirect.com/science/article/pii/S0168900208004592) Keywords: Medipix2; Pattern recognition; Online tracks analysis; Dosimetry
[2] Bergmann, B et al., ?3D track reconstruction capability of a silicon hybrid active pixel detector?, The European Physical Journal C, 2017,
https://doi.org/10.1140/epjc/s10052-017- 4993-4
[3] RS Stephens, Probabilistic approach to the Hough transform, Image and Vision Computing, Volume 9, Issue 1, 1991, Pages 66-71, ISSN 0262-8856,
https://doi.org/10.1016/0262-8856(91)90051-P.

(http://www.sciencedirect.com/science/article/pii/026288569190051P) Keywords: Hough transform; Probabilistic Hough Transform; maximum likelihood method; tracking
[4] Martin A. Fischler and Robert C. Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, In Readings in Computer Vision, Morgan Kaufmann, San Francisco (CA), 1987, Pages
726-740, ISBN 9780080515816,
https://doi.org/10.1016/B978-0-08-051581-6.50070-2.
(https://www.sciencedirect.com/science/article/pii/B9780080515816500702)
[5] X. Llopart, R. Ballabriga, M. Campbell, L. Tlustos, W. Wong, Timepix, a 65k
programmable pixel readout chip for arrival time, energy and/or photon counting measurements, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume 581, Issues 1?2, 2007, Pages 485-494, ISSN 0168-9002,
https://doi.org/10.1016/j.nima.2007.08.079.
(http://www.sciencedirect.com/science/article/pii/S0168900207017020) Keywords: Pixel; Photon counting; CMOS; Arrival time; Medipix; Micro-pattern gas detectors

Jméno a pracoviště vedoucí(ho) diplomové práce:

MSc. Benedikt Ludwig Bergmann,

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **08.02.2018** Termín odevzdání diplomové práce: **25.05.2018**

Platnost zadání diplomové práce: **30.09.2019**

MSc. Benedikt Ludwig Bergmann podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.

Datum převzetí zadání	Podpis studenta

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

Master's thesis

Machine learning approach to ionizing particle recognition using hybrid active pixel detectors

Bc. Petr Mánek

Supervisor: Benedikt Bergmann, M.Sc.

24th May 2018

Acknowledgements

I would like to express my deep sense of gratitude to the staff of IEAP for their support and for providing me with stimulating working environment. I would also like to thank my supervisor Benedikt Bergmann, MSc. for proposing and supervising this project. Lastly, I would like to thank my friends and family for their support, for without them, this work would not have been possible.

Declaration

In Prague on 24th May 2018 . . . .

Citation of this thesis

Mánek, Petr. *Machine learning approach to ionizing particle recognition using hybrid active pixel detectors.* Master's thesis. Czech Technical University in Prague, Faculty of Electrical Engineering, 2018.

Abstrakt

Detektory Timepix zaznamenávají snímky obsahující charakteristické obrazy částic ionizujícího záření, které procházejí vrstvou polovodičového materiálu. Od jejich vzniku ve skupině Medipix probíhaly pokusy o rekonstrukci trajektorií a rozpoznávání druhů částic z pozorovaných obrazů. Cílem této práce je návrh nových postupů řešení obou úloh, inspirovaných nedávnými články a metodami počítačového vidění.

Navržené algoritmy pro rekonstrukci trajektorií jsou robustní a dosahují subpixelového rozlišení díky využití lokální optimalizace a informace o energii dostupné v operačním režimu Time-over-Threshold. Narozdíl od jiných postupů umožňují úspěšnou detekci a oddělení až 10 překrývajících se obrazů. Rozpoznávání druhů částic je dosaženo algoritmem strojového učení, který provádí klasifikaci na základě modelu atributů ztráty energie částic. Křížovou validací dat s obrazy těžkých iontů se podařilo ukázat, že navržený klasifikátor plní požadovanou funkci.

Za účelem demonstrace využití navržených metod ve výzkumných aplikacích byla provedena analýza dat z detektorů Timepix rozmístěných v experimentu MoEDAL v LHCb, CERN.

Klíčová slova Medipix, Timepix, aktivní pixelový detektor, rozpoznávání, strojové učení, klasifikace, ionizující částice, rekonstrukce trajektorie, RANSAC, k-NN, simulované žíhání, MoEDAL

Abstract

Timepix detectors record frames containing characteristic patterns corresponding to particles of ionizing radiation passing through a layer of semiconductive material. Since their inception by the Medipix collaboration at CERN, attempts have been made at reconstruction of particle trajectories and recognition of particle species based on the observed patterns. This thesis proposes novel approaches to both problems inspired by the recent works and methods used in computer vision applications.

The presented algorithms for trajectory reconstruction are robust and combine local optimization with energies available in the Time-over-Threshold operation mode to achieve subpixel precision. Unlike alternate approaches, the proposed methods have been found to successfully detect and separate up to 10 overlapping patterns. A supervised machine learning algorithm is presented for particle species classification based on the energy loss feature model. Cross-validated evaluation of the classifier with heavy ion dataset indicates that the proposed model is viable and can accurately determine particle species.

An analysis of recent Timepix data from the MoEDAL Experiment at LHCb, CERN has been conducted to demonstrate usage of the presented methods in research applications.

Keywords Medipix, Timepix, active pixel detector, recognition, machine learning, classification, ionizing particle, trajectory reconstruction, RANSAC, k-NN, simulated annealing, MoEDAL

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Hybrid active pixel detectors are sensitive to particles of ionizing radiation. Much like chips of commercially available photographic cameras, they are comprised of a layer[1] of semiconductive material, which is divided into a square matrix of pixels. Individually, pixels are connected to readout electronics, which synchronize their operation and record their outputs. During data acquisition, charged particles induce signals in a subset of the pixels and thus leave characteristic imprints in the pixel matrix.

1.1 Timepix

Timepix [Llo+07] is a hybrid active pixel detector, developed within the MPX collaboration at CERN as a successor to the older Medipix2 model. The detector has 256×256 pixels with a 55 μm pitch. Each pixel has its own CMOS-based readout chain with a 14-bit counter register, which can be controlled independently of other pixels.

TPX detectors are operated by opening and closing a shutter – a binary signal controlling the state of the detector. Similar to photographic cameras, opening the shutter marks the beginning of a data acquisition period, during which the counter registers are sequentially incremented by interactions with incident charged particles. After the shutter is closed, values of counter registers are read out, producing so-called *frames* [Tur+11a]. Returning to the camera analogy, frames may be viewed as equivalent to digital pictures.

Among others, the behavior of pixel counter registers during data acquisition depends on *the operation mode* of the detector. Its selection usually depends on the application and effectively determines the semantics of counter values. In particular, earlier works have shown that in the *Time-over-Threshold* operation mode, counter value of a pixel can be calibrated to correspond with energy deposited in it by incident particles [Jak11].

[1]It is important to note that CCD chips used commercial photographic cameras are usually significantly thinner than those used in TPX detectors.

1.2 Patterns Produced by Charged Particles

In frames taken by calibrated TPX detectors, characteristic patterns may be observed. These patterns are produced by charged particles and may be interpreted as traces of their trajectories through the semiconductive sensor layer of the detector. In the past, efforts were made to identify such patterns, examine their features and reason about unknown properties of their corresponding particles [Hol+08].

Due to fast advances in pixel detector technology, some such methods have over time become insufficient or outdated. For instance, methods originally developed for older detector models fail to utilize increased pixel resolution and energy loss information available in the ToT operation mode.

Furthermore, some methods have been observed to not behave sufficiently robust in less-than-optimal settings. In practical operation, TPX frames are adversely affected by hardware malfunctions of individual pixels and effects of wrongly selected configuration parameters. As a consequence, these influences distort observed patterns, affecting the quality of their information.

1.3 Thesis Overview

The goal of this thesis is to utilize methods of computer vision to update existing methods for robust trajectory reconstruction, and to propose a machine learning algorithm for particle species classification. All presented methods are applied and evaluated in a set of experiments on real and synthetic data.

The rest of the text is divided in 5 chapters as follows:

- Chapter 2 gives relevant information on the hardware architecture and operation of TPX detectors. It also describes previous work done and defines a formal framework used in the rest of the thesis.
- Chapter 3 derives multiple algorithms for detection of known patterns left by ionizing particles in TPX detectors and robust randomized algorithms fitting particle trajectories.
- The focus of Chapter 4 is on selection and extraction of discriminative features for particle species classification. It also formally defines the classification task and proposes appropriate machine learning algorithm for solving it.
- Chapter 5 describes a multitude of experiments performed to evaluate the behavior of the presented detection and classification methods in a decoupled and a joint setting. It also proposes several modifications to improve the evaluation metrics.
- Chapter 6 discusses the results of the work, proposes various applications and outlines several possibilities for future research.

Chapter 2

Background

The aim of this chapter is to extend the introduction into Timepix detectors, give more background about interaction of ionizing radiation with matter and briefly summarize relevant previous works in this field.

2.1 Physics Background

Active pixel detectors use semiconductive materials to measure ionizing radiation in pixelated matrices. Hybrid detectors distinguish two major physically separate components – the semiconductive detection medium (called *the sensor layer*) and the readout electronics (constituted by an ASIC[2]). The separation between these components enables modular construction of detectors with various desirable properties by selection of appropriate semiconductive media. For Timepix detectors in particular, this approach enables versatile composition of detector assemblies depending on their application.

For the purposes of this work, hardware parameters of TPX detector assemblies are considered to be particularly relevant. These parameters include:

- Sensor layer thickness (usually given in μm),
- Sensor layer material (often used materials include e.g. Si, GaAs, CdTe).

2.1.1 Measuring Ionizing Radiation with Timepix

In order to fully deplete the active volume, a potential difference is applied to the backside contact and the pixelated electrodes. The potential is the so-called *bias voltage.* While the *back-side* electrode is common to all pixels, the *pixel site* electrodes are separately connected to individual pixels by means of so-called *solder bonds* (or *bump bonds*). This is illustrated in Figure 2.1.

[2]Application Specific Integrated Circuit

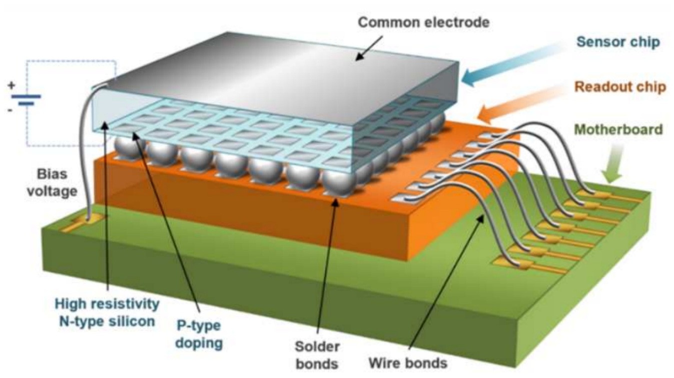


Figure 2.1: Cross-section rendering of various components of a TPX detector assembly.

During measurement, free charge carriers (electrons and holes) are created within the sensor layer. The applied potential difference results in their drift[3] through the sensor layer material, inducing currents at pixel electrodes closest to their positions. These are converted into analog voltage pulses and processed by the pixel electronics in the ASIC readout component. During drift and collection, the charge carriers induce measurable currents at the pixel electrodes, which are converted into voltage pulses, shaped and analysed. The received voltage pulses encode information about interactions occurring within the sensor layer.

To retain valuable information from the pulse, the ASIC uses a 14-bit integral counter register for every pixel. After amplification with a charge-sensitive preamplifier, the pulse is compared with a globally adjustable *threshold level* (THL). At every clock cycle, the value stored in the register may be incremented depending on the result of the comparison and *the operation mode* of the detector.

For TPX detectors, three operation modes are available (illustrated in Figure 2.2):

Counting Mode In this mode, the counter is incremented at every clock cycle in which the voltage pulse crosses[4] the THL level.

Time-of-Arrival Mode In this mode, the counter is incremented at every clock cycle starting at the first crossing of the THL level until the end of the acquisition (frame).

Time-over-Threshold Mode In this mode, the counter is incremented at every clock cycle in which the pulse exceeds the THL level.

TPX detectors are operated in a frame-based data acquisition scheme. In an analogy to commercially available photographic cameras, by opening and closing a shutter acquisition is started and stopped. At acquisition start, all counters are zeroed and the

[3]Depending on the sensor layer material, either electrons or holes are collected.

[4]The instance of *crossing* THL level is regarded to be a pair of consecutive clock cycles in which the voltage pulse is first observed below the THL level, then above it.

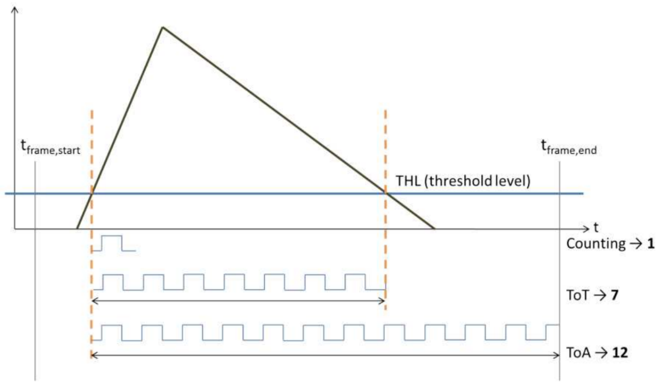


Figure 2.2: The behavior of the counter register in various TPX operation modes.

pixel matrix becomes active. After the shutter is closed, the detector stops registering events and the value of all counters is retrieved, constituting *a frame*.

The time elapsed between opening and closing of the shutter is called *the acquisition time*. Its value must be carefully determined in order to avoid numerical overflows in the 14-bit counter register (the register depth is 11810).

2.1.2 ToT Energy Measurements

In this work, focus is put on TPX frames taken in the Time-over-Threshold mode. As the name suggests, counter values in this mode track the amount of clock cycles the analog voltage pulse was observed to exceed the THL level.

The ToT mode has a particular significance since it has been shown that ToT counter values can be related to the energy deposited in pixels by incident particles [Jak11]. By calibration with characteristic X-ray fluorescence lines, four constants $a, b, c, t \in \mathbb{R}$ can be fitted for every pixel [Beg16]. The energy deposited in a calibrated pixel of counter value x is then approximated by function (illustrated in Figure 2.3)

$$f(x) = ax + b - \frac{c}{x - t} \text{ [keV]}. \tag{2.1}$$

For the reasons of simplicity, from this point onward all TPX frames are assumed to be taken in the ToT mode. Furthermore, energy calibration information is expected to be well-measured for every pixel and used to obtain energies from the corresponding counter values. This leads to the following definition of a frame.

Definition 2.1.1. A frame is a finite set of observed pixels $F = \{(p_i, e_i) \mid 1 \leq i \leq k\}$ where $p_i \in \mathbb{N}_0^2$ denote spatial coordinates within the pixel matrix of a TPX chip, $e_i \in \mathbb{R}^+$ denote the corresponding pixel energy in keV, and $k \in \mathbb{N}$ is the number of observed pixels.

This frame definition is designed to closely model a sparse list of pixels, which is the preferred transfer format for many readouts (e.g. FITPix) [Kra+11]. In addition,

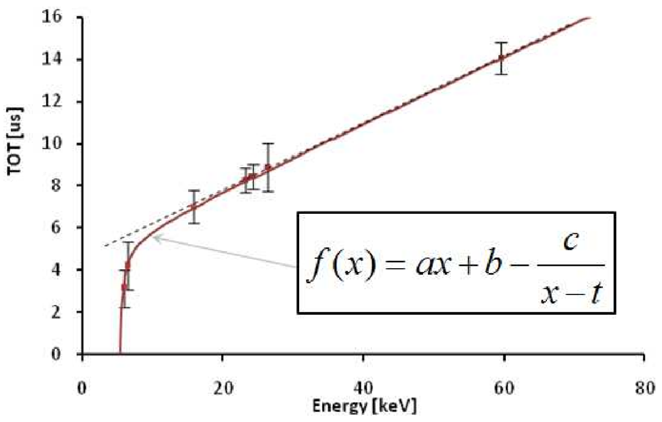


Figure 2.3: The dependence of ToT counter value (vertical axis) on the deposited energy in a pixel (horizontal axis) [Jak11].

this representation can be quickly constructed without any overhead from commonly used ASCII data file format produced by the Pixelman software package [Tur+11a]. For the purposes of this work, however, it is more practical to utilize the concept of image function used in computer vision.

Definition 2.1.2. Let $F = \{(p_i, e_i) \mid 1 \leq i \leq k\}$ be a frame. Function $f_F : P \to \mathbb{R}$ is the corresponding image function of F on pixel lattice P, if and only if the both of the following conditions are satisfied:

1. For all $i \leq k$, $f_F(p_i) = e_i$.
2. For all $p \in P$ such that $p \neq p_i$ for all $i \leq k$, $f_F(p) = 0$ keV.

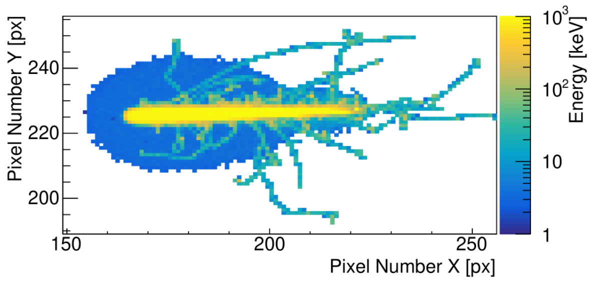
Remark. In this work, all image functions are assumed to operate on a lattice, which models the dimensions of a standard TPX chip. Such lattice is defined as $P_S = \{0, 1, 2, \ldots, 255\}^2$ and may be omitted in further mentions for the reason of clarity.

Remark. For a pixel $p = (x, y) \in P$, image function notations $f_F(p)$ and $f_F(x, y)$ are considered equivalent and thus used interchangeably.

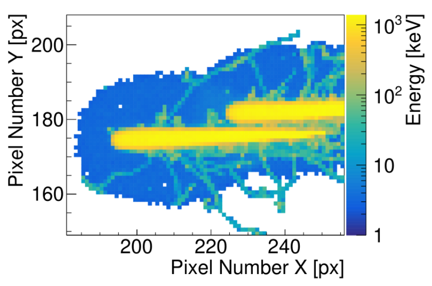
Image functions of TPX frames are similar to that of conventional images taken by commercial photographic cameras. For that reason, comparisons may be drawn between TPX image function energy values and pixel intensities.

2.2 Patterns Produced by Particles

When an ionizing particle traverses a thin layer of semiconductive material such as the TPX sensor layer, its energy is deposited in atoms of pixels surrounding its trajectory. [Boh13; Bet30; Blo33; Lan44] While specifics of this process (explained in further



(a) A cluster corresponding to a single track.



(b) Overlap cluster of two tracks.

Figure 2.4: Clusters produced by $^{36}_{18}$Ar ions of momentum 75 GeV/c.

detail in Section 4.1) depend on multiple variables, the effect produced in TPX frames captured in ToT operation mode can be summarized as follows.

In the frame, pixels directly incident with the particle trajectory are ionized the most, reporting the largest energy values. Due to charge sharing, adjacent pixels receive fractions of the deposited charge, reporting energy values which decrease as their distance from the particle trajectory increases. At some distance from the trajectory, deposited charge drops[5] below the counter THL level, producing zero energy values for any pixel from that point onward (this decrease is shown in Figure 2.6). The result of this process is a *track* – a local pattern of connected pixels with non-zero energy values. In ideally sparse frames, tracks correspond to *clusters* – isolated pixel sets bounded by pixels of zero energy values (example shown in Figure 2.4a).

In practical measurements with TPX detectors, multiple particles usually pass through the TPX sensor layer during a single frame acquisition period. Depending on their trajectories and detector configuration, tracks produced by such particles may share pixels and thus appear as one cluster in the frame. Clusters formed by such tracks are called *overlaps* (shown in Figure 2.4b). In the ToT operation mode, pixels of overlaps ionized by multiple particles report the sum of all energy values received.

2.2.1 Morphology of Ion Tracks

When examining tracks produced by ions, it is possible to identify several characteristic regions of interest: [Hoa+12]

Core: A region containing pixels with the largest energy values. As such pixels are known to be directly incident (or closely adjacent) to the particle trajectory, it encodes location information about its traversal through the TPX sensor layer.

Halo: A lower-energy region surrounding the core. Depending on the particle energy and configuration of the THL level parameter, area of the halo region may vary.

[5]In this chapter, TPX frames are assumed to be mostly sparse. Since the occupancy of a frame is in part determined by the number of particles traversing the sensor layer during the acquisition period, it is possible to achieve this result by lowering the acquisition duration parameter.

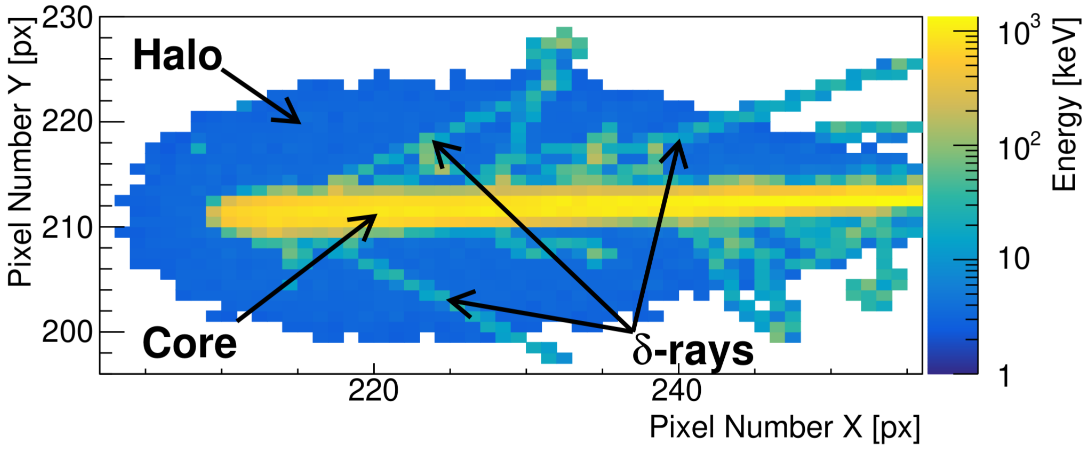


Figure 2.5: Ion of ${}^{36}_{18}$Ar with annotated core region, halo region and δ-rays.

δ-rays: Electrons emitted by the particle at random during its traversal through the sensor layer material. As such, they usually appear to be one pixel wide, start within the core and move away through the halo region.

An example of the above listed regions is shown in Figure 2.5.

2.3 Review of Previous Works

The task of particle detection and classification in hybrid active pixel detectors has been tackled in various forms since their inception.

Holy et al. [Hol+08] first approached the problem in older Medipix2 detectors where only binary information is available for every pixel. In his work, the detection task was solved by means of morphological cluster analysis. This approach relied on the assumption that detector acquisition time is configured to a sufficiently short duration. After their separation into clusters, particles were classified by a morphological classifier into 6 characteristic cluster classes related to particle species.

While in the work, the presented method has been shown to achieve accurate discrimination between individual components of radiation fields composed of α, β and γ sources, it is important to note that the proposed cluster types were not in one-to-one correspondence with classified particle species. In fact, a particle could be assigned various classes depending on the angle of incidence of its trajectory. Furthermore, any change in detector acquisition parameters (e.g. THL level or bias voltage) heavily affected the properties of patterns produced in frames, possibly prompting further manual adjustments of classification thresholds which were not automatically adapted or learned in any way. The adverse effect of this was later diminished by Granja et al. [Gra+11] who has shown that for certain charged particle measurements, optimal bias voltage and THL level can be determined.

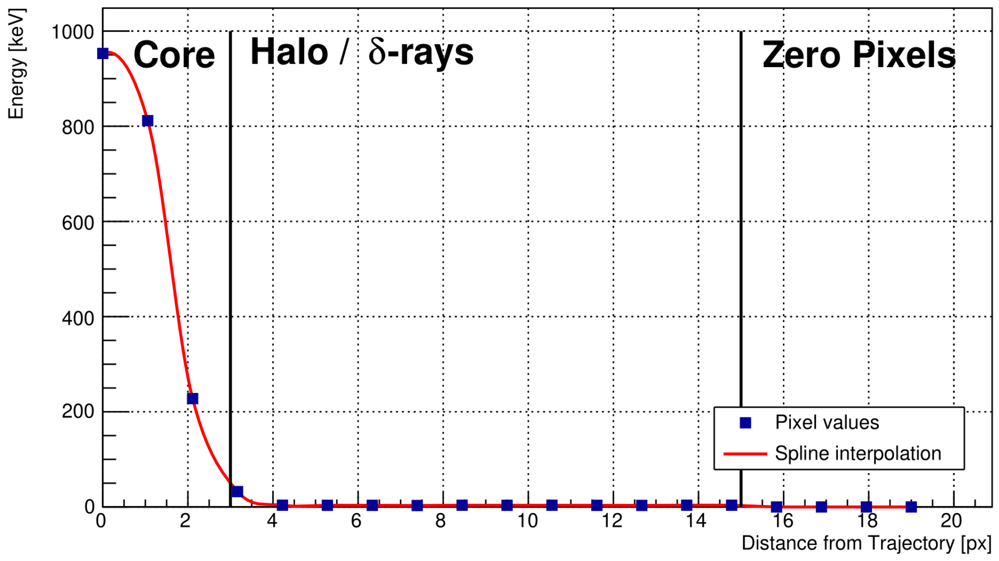


Figure 2.6: Dependence of pixel energy on distance from particle trajectory. The plot shows sampled energy for a cluster produced by $^{36}_{18}$Ar ion of momentum 75 GeV/c at pixels of increasing distance. Boundaries between regions defined in Section 2.2.1 are labeled.

Transitioning from Medipix2 to modern Timepix hardware, the presented morphological approach can still be applied. Since TPX detectors however offer more information in every pixel than just binary values, the method obviously underutilizes the amount of data available.

In an attempt to utilize counter information in TPX detectors, Vilalta et al. [Vil+11; Kuc11] updated and extended Holy's morphological cluster analysis algorithm for uses motivated by machine learning applications. With a detector operating in ToT mode, deposited energy could be extracted from frames by prior energy calibration [Jak11]. While this prospect opened a way towards spectral – not only morphological – feature extraction, it was also attractive since it was previously shown that energy information, especially in heavy charged particles, could help achieve subpixel spatial resolution [Jak+08].

In his work, Vilalta et al. proposed 6 cluster properties for use in classification, some of them obtained by fitting clusters with elliptical shapes. The author has evaluated his proposed feature model using various classifiers, in particular: decision trees, support vector machines, artificial neural networks and naive Bayesian classifiers. In the conclusion of the work, classification accuracy was enhanced by including spectral histograms in the feature model.

Further interest in particle classification was prompted by various applications in dosimetry. In his article, Turecek et al. [Tur+11b] used Holy's cluster analysis algorithm for the calculation of linear energy transfer (LET). As the value of this quantity is dependent on the length of the particle trajectory traversed in the sensor layer material, his algorithm used morphological skeleton for its estimation in certain clusters.

While at the time, LET values were primarily utilized for the estimation of the equivalent dose rate, the quantity soon found more practical applications. For instance, Stoffle et al. [Sto+12] used LET for calculation of stopping power of heavy ions. To enhance the accuracy of his estimation method, he proposed distinguishing between 2 characteristic regions in clusters, whereby only *the central region* – the most energetic part of a cluster – would be used for the stopping power estimation while the remainder of the cluster would be discarded.

LET estimation was later further extended by Hoang et al. [Hoa+12] who also studied clusters produced by heavy charged particles and proposed alternate, more sophisticated division into 3 distinctive regions of interest. In his work, the outer region was divided into *the skirt* and a multitude of *δ-rays*. In addition to LET estimation, the central region was also used to obtain spatial information about the incident particle trajectory. The fitting procedure used in the work utilized azimuthal estimation with reprojection to determine cluster boundaries in orthogonal directions.

In his later works, Hoang et al. [Hoa13; Hoa+14] adapted his method for conventional machine learning by extracting viable features not only from the central cluster region and LET, but also from the count of delta rays estimated by applying distance transform on the corresponding cluster region. For particle species classification, the author proposed a sophisticated method utilizing an ensemble of classifiers in a decision tree scheme. In addition to the previously mentioned classifiers, his work also tested boosted decision trees and k-NN classifiers showing that these particular methods can be used with sufficient accuracy.

Stoffle et al. [Sto+15] later used a similar method for the enhancement of the equivalent dose rate estimations. In his study of stopping ions [SP18], he experimented with fitting so-called Bragg curves and using goodness-of-fit tests in order to determine whether an ion has been completely absorbed in the sensor layer material. While this approach achieved the stated objective,s in the conclusion to his work the author noted that the results yielded were less than optimal and the entire fitting procedure was computationally intensive, offering many points for further improvements.

2.4 Formal Framework

For due diligence, this section introduces standard terminology which is utilized later in the work. Note that the most of the presented terms are either considered to be commonly known or have been previously used in other literature.

Definition 2.4.1. Function $d : P^2 \to \mathbb{R}$ is called distance if, and only if it satisfies all of the following conditions for all $p, q, r \in P$.

1. Non-negativity: $d(p, q) \geq 0$ (specially $d(p, p) = 0$),

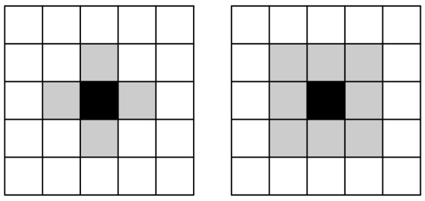


Figure 2.7: Examples of pixel 4-neighborhood (in the left) and 8-neighborhood (in the right). While the pixel is filled in black color, its respective neighborhood is filled with gray.

2. Symmetry: $d(p, q) = d(q, p)$,
3. Triangle inequality: $d(p, r) \leq d(p, q) + d(q, r)$.

This work utilizes the Euclidean distance, which is defined as follows.

Definition 2.4.2. The Euclidean distance between two pixels (x, y) and $(h, k) \in P$ is defined as $d_E((x, y), (h, k)) = \sqrt{(x-h)^2 + (y-k)^2}$.

Another utilized concept is that of pixel neighborhood (illustrated in Figure 2.7).

Definition 2.4.3. The 4- and 8-neighborhood are set functions $\mathcal{N}_4, \mathcal{N}_8 : P \to 2^P$ respectively, such that for all pixels $(x, y) \in P$ the following conditions hold:

- $\mathcal{N}_4((x, y)) = P \cap \{(x+h, y+k) \mid h, k \in \{-1, 0, 1\} : |h| + |k| \leq 1\}$,
- $\mathcal{N}_8((x, y)) = P \cap \{(x+h, y+k) \mid h, k \in \{-1, 0, 1\}\}$.

Remark. Note that every pixel is a part of its 4- and 8-neighborhood.

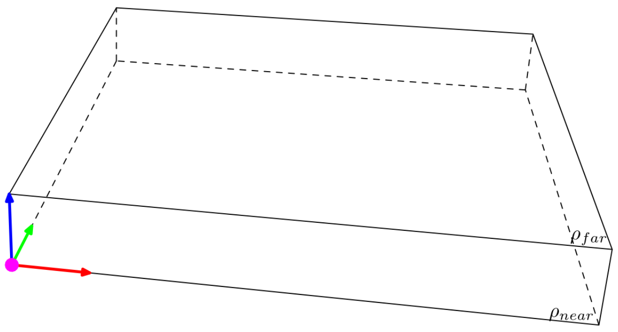
2.4.1 Coordinate System

The coordinate system used in this work is defined consistently with Timepix pixel numbering [Llo+07]. The X, Y and Z axes form a Cartesian coordinate system, where the X and Y axes lie in the plane of the ASIC, while the Z axis points from the plane of the ASIC towards the incidence plane in normal direction. Consequently, the origin is placed in the plane of the ASIC in the left bottom corner.

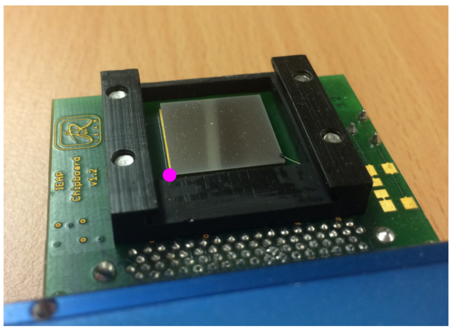
In the coordinate system, two planes are used to describe incidence events with the TPX sensor plane. The plane parallel to the X-Y plane at $z = 0$ is labeled ρ_{near} due to its close distance to the ASIC. For detector of sensor chip thickness t, the plane parallel to the X-Y plane at $z = t$ is labeled ρ_{far}. The coordinate system as well as both planes are illustrated in Figure 2.8.

Based on these planes, two major directions of three-dimensional trajectories intersecting the TPX sensor layer are distinguished:

Towards ASIC (TA) Trajectories of this direction intersect ρ_{far} at earlier point in time than ρ_{near}.



(a) The used coordinate system with the unit vectors e_i, e_j, e_k of axes X, Y, Z displayed in red, green and blue respectively. The planes ρ_{near} and ρ_{far} are denoted.



(b) GaAs TPX detector of thickness 500 μm connected to a FITPix readout interface [Kra+11].

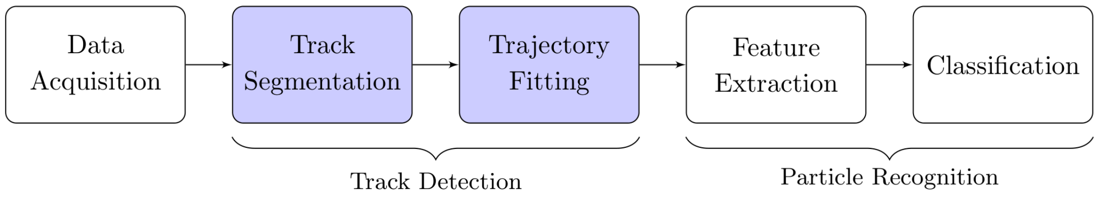
Figure 2.8: Correspondence of the defined coordinate system (shown in the left) with real TPX detector hardware (shown in the right). The magenta dot marks the same spatial point in both pictures.

From ASIC (FA) Conversely, trajectories of this direction intersect ρ_{near} at earlier point in time than ρ_{far}.

Chapter **3**

Track Detection

This and the next chapter describe the presented system in detail. Following its individual components, the first one is *track detection*. Note that here the word *detection* is considered rather in the context of image processing than physics, referring to the task of identification and characterization of local features of interest in TPX frames.



Track detection is decoupled into two stages:

Track Segmentation: In this stage, frames taken by a calibrated TPX detector are analyzed. The output is a multitude of extracted pixel sets likely to correspond with individual tracks (i.e. particle trajectories).

Trajectory Fitting: The purpose of this stage is to estimate parameters of spatial trajectory model for every pixel set produced by the previous stage.

Before describing the specifics of both stages, it is necessary to note the advantages and disadvantages of a decoupled approach. One obvious advantage is that by decomposition into two separate subproblems, the detection problem becomes simplified. For instance, since the dimensionality of the problem is determined in track segmentation, the solution domain for trajectory fitting is significantly diminished. While this lowers the computational complexity, in practice this is also desirable for parallelization. In a multi-threaded environment, segmented pixel sets can be fitted independently of each other, presumably utilizing multiple processing cores at the same time.

On the other hand, the most significant disadvantage of a decoupled approach is the possibility of *cascade failures* – undesirable situations where an incorrect result of an earlier stage of the algorithm adversely affects all of its subsequent stages. To decrease their probability, it is important to minimize the failure rate of the segmentation stage and maximize the robustness of the fitting stage.

3.1 Track Segmentation

The focus of this section is on the segmentation stage. As stated before, segmentation partitions the set of pixels of a TPX frame into subsets likely corresponding to individual tracks. In this aspect, TPX segmentations differ from conventional[6] segmentations as they do not require partitioned sets to be disjoint in any way.

The aim of segmentation is to ensure that no pair of pixels ionized by the same particle are partitioned into different sets. While this technically permits trivial solutions (e.g. one large pixel set for the entire), it is desirable for a segmentation partitioning to yield as many subsets as possible, possibly having a great influence on the complexity and accuracy of all subsequently applied algorithms.

In this work, two segmentation methods are used. A preliminary segmentation is produced by *morphological connectivity checking* and then refined by *the Hough Transformation.*

3.1.1 Morphological Connectivity Checking

Morphological connectivity checking algorithms have originally found use in computer graphics and are commonly used methods of obtaining segmentation from TPX frames [Tur+11b]. They exploit sparsity of pixel data which has been shown to be easily attainable by setting sufficiently short duration as frame acquisition time [Hol+08]. Since in sparse frames, clusters are bounded by multiple pixels of zero energy value, it is possible to separate them by propagating a search wave over non-zero pixels. This line of reasoning leads to algorithms such as Flood fill (or Seed fill) [Hec90].

Adapted version of the algorithm repetitively finds an unvisited *seed pixel* of a non-zero energy value. Once such pixel is located, its neighbors satisfying the same criteria are found and recursively explored. This process continues as long as possible. When no more pixels can be discovered, the explored pixel set is marked as a new cluster and the algorithm resumes its search for a new seed pixel, exploring the rest of the frame. This approach is summarized in Algorithm 1.

While morphological connectivity checking never partitions pixels ionized by the same particle into multiple clusters under the assumption that no pixel is *dead* (falsely reports zero energy values e.g. due to a detector malfunction or radiation damage), it is unable to detect overlaps. This motivates the use of the Hough Transformation described in the next section.

[6]In conventional segmentations (e.g. foreground vs. background), every pixel is usually assigned exactly one class.

Algorithm 1 Segmentation by Morphological Connectivity Checking

```
1: $V \leftarrow \{\}$, $r \leftarrow 1$
2: while $F \setminus V$ is not empty do
3:     Select seed pixel $p_s \in F \setminus V$.
4:     $C_r \leftarrow \{\}$, $Q \leftarrow \{\}$, $A \leftarrow \{p_s\}$
5:     while $A$ is not empty do
6:         $Q \leftarrow Q \cup \{(x, e) \in A \setminus V \mid e > e_0\}$
7:         $V \leftarrow V \cup A$
8:         $A \leftarrow \{\}$
9:         if $Q$ is not empty then
10:            Select $p \in Q$, $Q \leftarrow Q \setminus \{p\}$
11:            $C_r \leftarrow C_r \cup \{p\}$
12:            $A \leftarrow \mathcal{N}(p)$
13:    if $C_r$ is not empty then
14:        Report $C_r$ as new cluster, $r \leftarrow r + 1$.
```

3.1.2 Hough Transformation

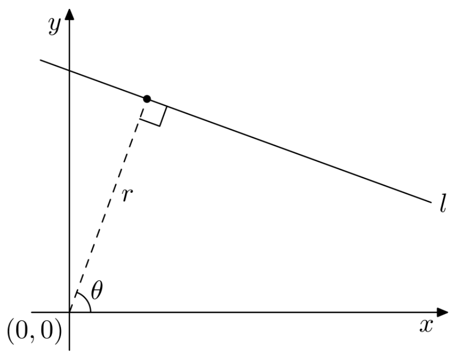
Hough Transformation [Hou62] is a general method used for detection of complex, parametric shapes and patterns (e.g. lines, circles, etc.) in computer vision. Its major advantages include efficient tractability and robustness with respect to a great amount of input noise.

The general algorithm behind the transformation is straightforward. In order to identify particular instances of a parametric model, an accumulator space is constructed. The accumulator space, typically represented by an array, is of the same dimensionality as the parameter domain of the model and is addressed by its elements which may be quantized to achieve more stable results. When an image is presented to the algorithm, its pixels increment a multitude of bins in the accumulator space, casting *votes* for consistent model instances. After all pixels are discounted, local maxima are identified in the accumulator space, marking the most probable model instances. While this general algorithm can be adapted to various parametric shapes, this work uses Hough in a specific algorithm proposed for line detection [DH72].

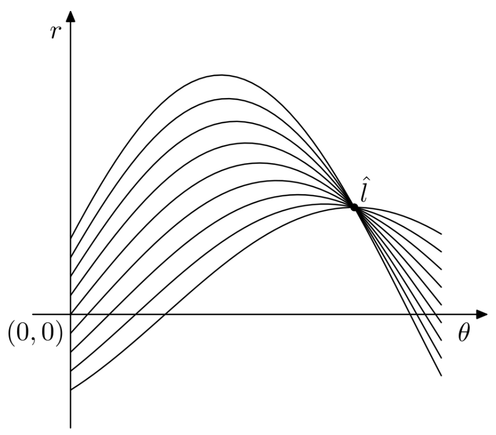
To avoid problems caused by possibly unbounded values of conventional linear parameterizations, the proposed algorithm uses Hesse normal form, which parameterizes lines as

$$r = x_1 \cos\theta + x_2 \sin\theta \qquad \text{where } \theta \in [0, \pi], r \in [-r_{\max}, r_{\max}] \tag{3.1}$$

This parametrization (illustrated in Figure 3.1a) ensures that all (including vertical) lines can be encoded, given sufficiently large value of $r_{\max}$. For a point $x = (x_1, x_2)$, the set $\mathcal{L}(x)$ of all consistent lines can be efficiently enumerated as



(a) Parametrization of line l.



(b) Hough image $\hat{l}$ of line l.

Figure 3.1: Parametrization in Hough transformation for lines. [DH72] The left diagram shows line l (solid) parameterized by a distance r from the origin and angle θ of the corresponding normal (dashed). In the right, the Hough images of lines intersecting various points from l are displayed. Note that Hough image $\hat{l}$ lies in their intersection.

$$\mathcal{L}(x) = \{(\theta, x_1 \cos\theta + x_2 \sin\theta) \mid \theta \in [0, \pi]\} \tag{3.2}$$

The set $\mathcal{L}(x)$ describes a continuous planar curve in the accumulator domain. It holds that the Hough image $\hat{l}$ of any line l lies in the intersection $\mathcal{L}(x) \cap \mathcal{L}(y)$ of any two points $x, y \in l$. This property (illustrated in Figure 3.1b) is exploited by the line detection algorithm.

3.1.3 Non-maxima Suppression

While the original Hough Transformation algorithm detects lines by exhaustive search for local maxima in the accumulator space, it is known that this approach may lead to duplicate and possibly inaccurate detections caused by redundant identifications of adjacent points in the accumulator. For this reason, a non-maxima suppression method is proposed to resolve this issue.

Instead of a set of local maxima, the presented algorithm identifies a single global maximum over the entire accumulator space, ensuring that the most prevalent line is chosen. The maximum is then suppressed along with other relevant non-maxima and the search for another global maximum is resumed. The search terminates when the value of the global maximum of the accumulator space surpasses an adjustable detection threshold A_{min}.

The suppression procedure for a global maximum (θ, r) relies on correct identification of its inlier pixels in the original image. In this work, such pixels are identified by thresholding on maximum distance from the line. Once found, the votes of the inliers can be simply subtracted from the accumulator, suppressing the maximum along with its adjacent non-maxima.

3.1.4 Usage in Segmentation

The presented variant of the Hough Transformation algorithm for line detection may be directly applied to TPX data as an alternative to connectivity checking. Since the algorithm however detects analytical lines, additional reasoning would be required to obtain correct pixel partitioning for its detections in the scope of a TPX frame. Furthermore, it may not be desirable to only detect lines due to the fact that some particle trajectories with incident angle $\theta << 90°$ produce tracks of more circular than linear shape. For these reasons, the Hough algorithm is rather used for refinement after morphological connectivity checking, possibly increasing the number of subsets produced in cases where overlaps are detected.

When presented with an input set of pixels, the implemented algorithm initializes and fills the accumulator space with pixel votes. Then, sequentially, global maxima are identified, removed and suppressed until a threshold vote count A_{min} is reached or maximum number of iterations s_{max} is exceeded. The inliers of identified maxima are segmented into separate pixel sets and reported. This algorithm is summarized in Algorithm 2. An example of its run is shown in Figure 3.2.

Algorithm 2 Segmentation Refinement by Hough Transformation

```
1: $A \leftarrow$ empty accumulator, $s \leftarrow 1$
2: **for** $(x, e) \in F$ **do**
3:     **for** $(\theta, r) \in \mathcal{L}(x)$ **do**
4:         Cast vote $A(\theta, r) \leftarrow A(\theta, r) + \Phi(e)$.
5: **repeat**
6:     $(\theta^s, r^s) \leftarrow \text{argmax}_{(\theta,r)} A(\theta, r)$
7:     $A^s \leftarrow A(\theta^s, r^s)$
8:     $C_s \leftarrow \{(x, e) \in F \mid x \text{ consistent with } (\theta^s, r^s)\}$
9:     **for** $(x, e) \in C_s$ **do**
10:        **for** $(\theta, r) \in \mathcal{L}(x)$ **do**
11:            Suppress inlier vote $A(\theta, r) \leftarrow A(\theta, r) - \Phi(e)$.
12:    Report subset $C_s$.
13:    $s \leftarrow s + 1$
14: **until** $A^s \leq A_{\text{min}}$ or $s \geq s_{\text{max}}$
```

3.2 Trajectory Parametrization

Based on observed tracks left in TPX detectors by particles of interest, it is possible to derive and describe a viable model characterizing particle trajectory traversing the

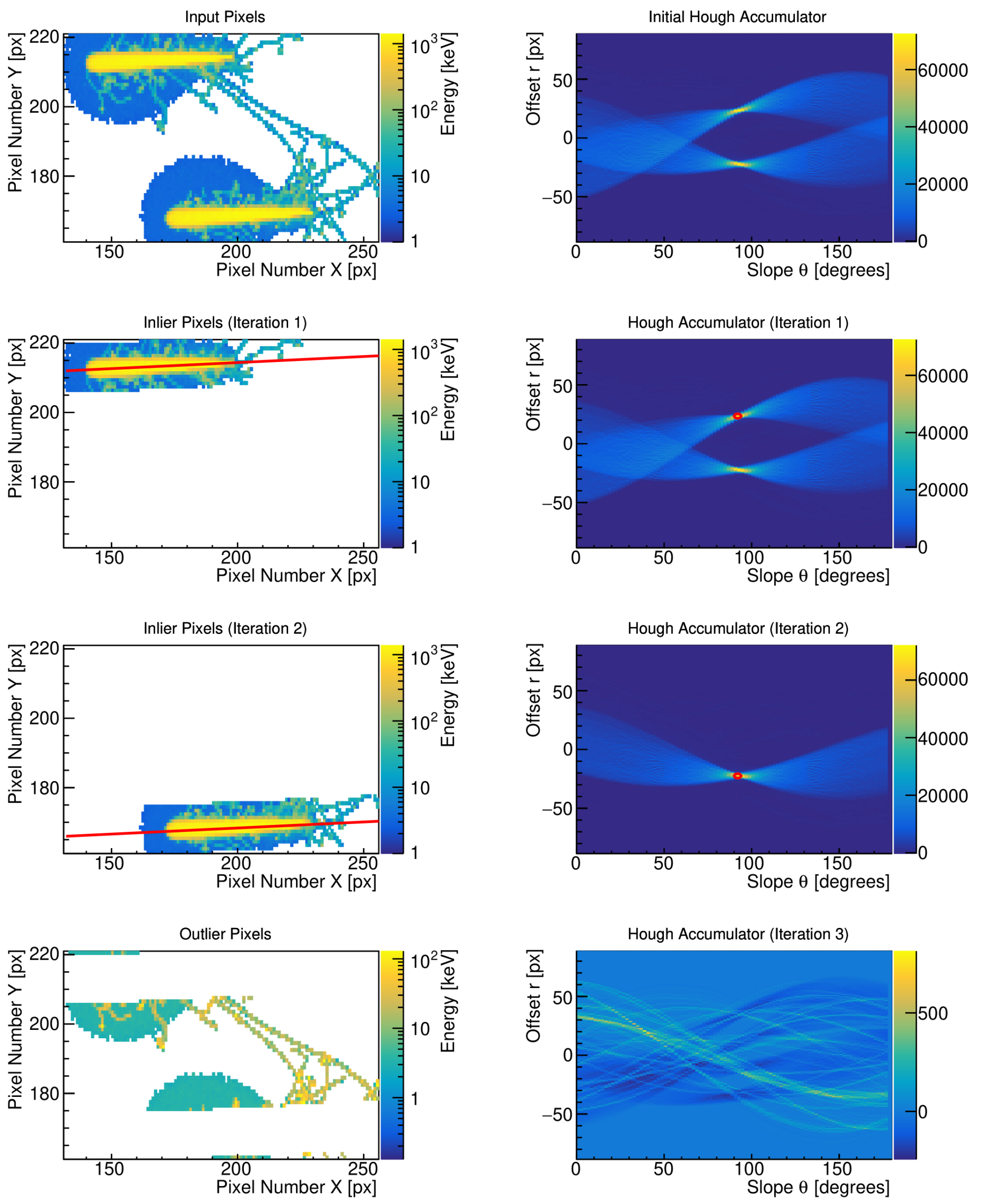


Figure 3.2: Usage of the presented non-maxima suppression method shown on an overlap cluster of two ${}^{36}_{18}$Ar ions. In each iteration (displayed in rows), a global maximum (marked by a red circle) is found in the accumulator (right column). The inliers (left column) of the corresponding line (indicated in red) are found and their votes are subtracted for the next iteration. Once the accumulator maximum drops below the threshold of 5000, the algorithm terminates to prevent redundant detections (last row). Note that the offset r is expressed with respect to the center of the bounding box.

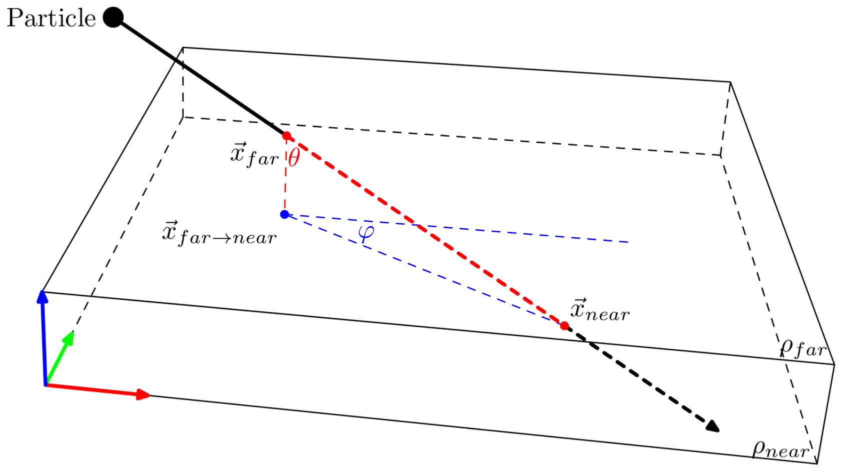


Figure 3.3: A perspective illustration of the core model. A linear particle trajectory passing through a TPX sensor layer is depicted as a thick arrow. The angles θ and φ are annotated and colored in red and blue, respectively.

sensor layer of a TPX detector. This model is the subject of the trajectory fitting stage, and is used for feature extraction at later stages of the system.

Let $x(t)$ denote the position of a particle at time t. Then the presented model is derived under the following assumptions:

(A1) Particle trajectory intersects planes ρ_{near} and ρ_{far} within pixel matrix bounds.[7]

(A2) Only a small fraction of the initial energy of the particle is lost in the sensor.

Under the stated assumptions, the trajectory of the particle is linear within the sensor layer. It can thus be described by two intersection points $\vec{x}_{\text{far}}, \vec{x}_{\text{near}}$ with planes $\rho_{\text{far}}, \rho_{\text{near}}$ respectively, or by spherical coordinates consisting of a single intersection point and two angles. In this work, the latter representation is used.

Let $\Delta\vec{x} = \vec{x}_{\text{near}} - \vec{x}_{\text{far}}$, the angle θ (also referred to as the *incidence angle*) is defined as the angle closed by $\vec{v}$ and the normal vector of ρ_{far}. Let $\vec{x}_{\text{far}\to\text{near}}$ be the orthogonal projection of $\vec{x}_{\text{far}}$ into ρ_{near} and $\vec{w} = \vec{x}_{\text{far}\to\text{near}} - \vec{x}_{\text{near}}$. The angle φ (also referred to as *the azimuth*) is defined as the angle closed by $\vec{w}$ and the X-axis.

With angles φ and θ, particle trajectory within the sensor layer is parameterized by the particle core model as tuple $(\vec{x}_{\text{far}}, \varphi, \theta)$. Perspective illustration of this parametrization is shown in Figure 3.3.

[7]Note that this assumption forbids any trajectories entering or exiting the TPX sensor layer from a side. The occurrence rate of such trajectories depends on the environment and the orientation of the detector.

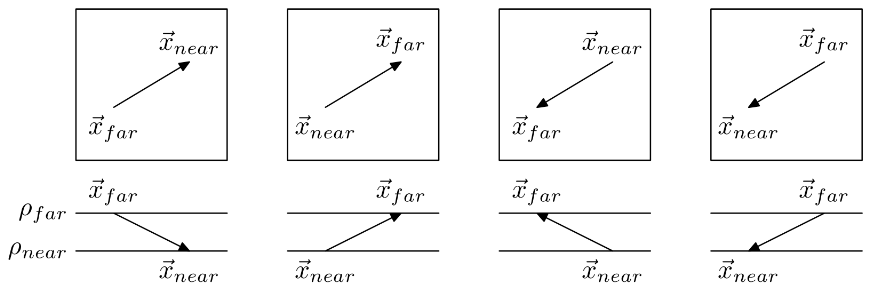


Figure 3.4: Four ambiguous particle trajectories indistinguishable when estimating the proposed model from a point pair. In each case, the illustration shows top view of the detector as well as the corresponding horizontal cross-section of the sensor layer.

3.2.1 Degrees of Ambiguity

Clearly, the core model parametrization does not capture particle trajectory completely. It is, however, a sufficiently descriptive approximation for the purposes of this work. In addition, one major advantage is that this particular parametrization can be efficiently estimated from a TPX frame by means of sampling point pairs (details are given in Section 3.3.1).

Nevertheless, it is important to note that given an arbitrary point pair, multiple ambiguous parameterizations exist due to two degrees of ambiguity:

Near-Far Ambiguity This ambiguity is caused by two possible ways of labeling sampled points as either projections of $(\vec{x}_{\text{far}}, \vec{x}_{\text{near}})$ or $(\vec{x}_{\text{near}}, \vec{x}_{\text{far}})$.

Entry Point Ambiguity This ambiguity is caused by two possible directions of trajectory (TA and FA), yielding the entry point into the sensor layer either at $\vec{x}_{\text{far}}$ or at $\vec{x}_{\text{near}}$. Note that this ambiguity may be resolved by the analysis of the lateral expansions of heavy tracks and by the charge carrier drift time measurement in TPX3 detectors [Ber+17].

All four possible cases arising from this are depicted in Figure 3.4.

3.3 Trajectory Fitting

At the end of the segmentation stage, the TPX frame is partitioned into a multitude of pixel sets. Assuming that, with high probability, these sets correspond with individual tracks, and in turn particles traversing the TPX sensor layer, the next stage of the detection process consists of fitting core model parameters for every such set.

To solve the fitting task, previous works utilized various different approaches. For instance, linear regression with reprojection [Hoa+12] or fitting the pixel set with elliptical shapes [Vil+11]. In this work, the primary concern is to maximize robustness in order to minimize the probability of a cascade failure. For that reason, the selected fitting method is a randomized algorithm based on the well-known RANSAC scheme [FB87]. Having various applications in conventional computer vision, a major advantage of such approach is the simplicity of its implementation and the robustness of its output in spite of possibly numerous outliers.

Short for *Random Sample Consensus*, RANSAC algorithms repeatedly sample observed data at random, and fit samples with some parametric model, counting the number of inliers (which is sometimes also referred to as *the support*). Throughout their run, RANSAC algorithms keep the model with the largest support, possibly improving it at the end of each iteration. Convergence to the model with the largest support is based on the notion that given a sufficient number of iterations, inliers of such model are eventually sampled and the model is accepted.

For track detection in a TPX frame, the domain from which samples are drawn is a pixel set and the fitted model is a single particle core, as defined in Section 3.2.

3.3.1 Fitting Cores from Sampled Pixels

To fit a core model, two pixel samples are required. Let $\vec{a} = (a_1, a_2)^T$ and $\vec{b} = (b_1, b_2)^T$ denote sampled coordinates in the TPX pixel matrix respectively, and let t be the sensor layer thickness. Then, without the loss of generality, the far point is $\vec{x}_{\text{far}} = (a_1, a_2)^T$.

The azimuthal angle φ satisfies

$$|\Delta\vec{x}| \cos\varphi = \Delta x_1, \qquad |\Delta\vec{x}| \sin\varphi = \Delta x_2, \qquad \text{where } \Delta\vec{x} = (\Delta x_1, \Delta x_2) = \vec{b} - \vec{a} \tag{3.3}$$

Furthermore, following from the right triangle determined by points $(a_1, a_2, t)^T$, $(a_1, a_2, 0)^T$ and $(b_1, b_2, 0)^T$ which is shown in Figure 3.5, the incidence angle θ satisfies

$$\tan\theta = \frac{|(b_1, b_2, 0)^T - (a_1, a_2, 0)^T|}{|(a_1, a_2, t)^T - (a_1, a_2, 0)^T|} = \frac{|\Delta\vec{x}|}{t} \tag{3.4}$$

From equations 3.3 and 3.4, angles φ and θ can be uniquely determined, yielding a fully fitted core model $\mathcal{M} = (x_{\text{far}}, \varphi, \theta)$.

3.3.2 Utility Function

Algorithms following the conventional RANSAC scheme use a discrete number of inliers (i.e. samples, which are consistent with the fitted model in some sense) to evaluate model quality. While this quantity has a significant advantage in its calculation time, it poses a problem for the local optimization extension discussed in detail in Section 3.3.6.

For that reason, the presented algorithm maximizes a more general utility function:

$$U_{w,\Phi}(\mathcal{M} \mid F) = \int_{s=0}^{1} \int_{t=-t_{\max}}^{t_{\max}} w(|t|)\Phi(f_F((1-s)\vec{a} + s\vec{b} + t\vec{n}))dtds. \tag{3.5}$$

Here, $t_{\max}$ is a sufficiently large distance (e.g. the largest distance between a pair of pixels), $\vec{n}$ is a normalized vector normal to $\Delta\vec{x}$, $w : \mathbb{R}_0^+ \to [0, 1]$ is a distance weight

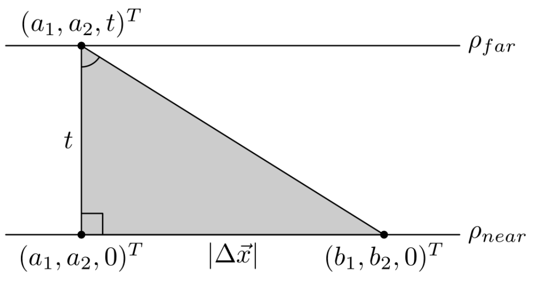


Figure 3.5: A cross-section of the TPX sensor layer showing a right triangle (gray) used in model fitting with its corresponding right angle (bottom left). The legs of the triangle are detector thickness t and projected core length $|\Delta\vec{x}|$. The angle θ is marked in the top left.

function and $\Phi : \mathbb{R}_0^+ \to \mathbb{R}$ is an energy kernel function. While the parameters $t_{\max}$ and n have constant assigned values, the choice of suitable functions w and Φ is discussed in Sections 3.3.3 and 3.3.4.

3.3.3 Energy Kernel Function

The purpose of the energy kernel function Φ is to determine pixel utility for core model fitting based only on its observed energy value. This notion is motivated by observation that pixels in the core region can be thresholded (or otherwise segmented) from the energy spectrum (illustrated in Figure 2.6). For that reason, it is expected that a well-chosen energy kernel shall identify portions of the spectrum characteristic for pixels ionized by desirable particles from those characteristic for pixels ionized by undesirable particles or no particles at all.

Since the proposed algorithm maximizes $U_{w,\Phi}$ (and in turn Φ), the function can thus reward favorable energies by assigning them positive values and punish unfavorable energies by assigning them zero or negative values. Furthermore, the magnitude of assigned values can be used to determine the relative importance of the energies observed. In this section, several candidate families of simplistic energy kernels are proposed along with their parametrization:

$$\Phi_{\text{step}}(e) = \text{sgn}(e - E_0) \qquad \text{where } E_0 \in \mathbb{R} \tag{3.6}$$
$$\Phi_{\text{linear}}(e) = ke + q \qquad \text{where } k, q \in \mathbb{R} \tag{3.7}$$
$$\Phi_{\log}(e) = a \cdot \log(be + c) \qquad \text{where } a, b, c \in \mathbb{R} \tag{3.8}$$
$$\Phi_{\text{ReLU}}(e) = e_0 + \max\{0, k(e - E_0)\} \qquad \text{where } E_0, e_0, k \in \mathbb{R} \tag{3.9}$$

Example plots of the proposed kernels are shown in Figure 3.6.

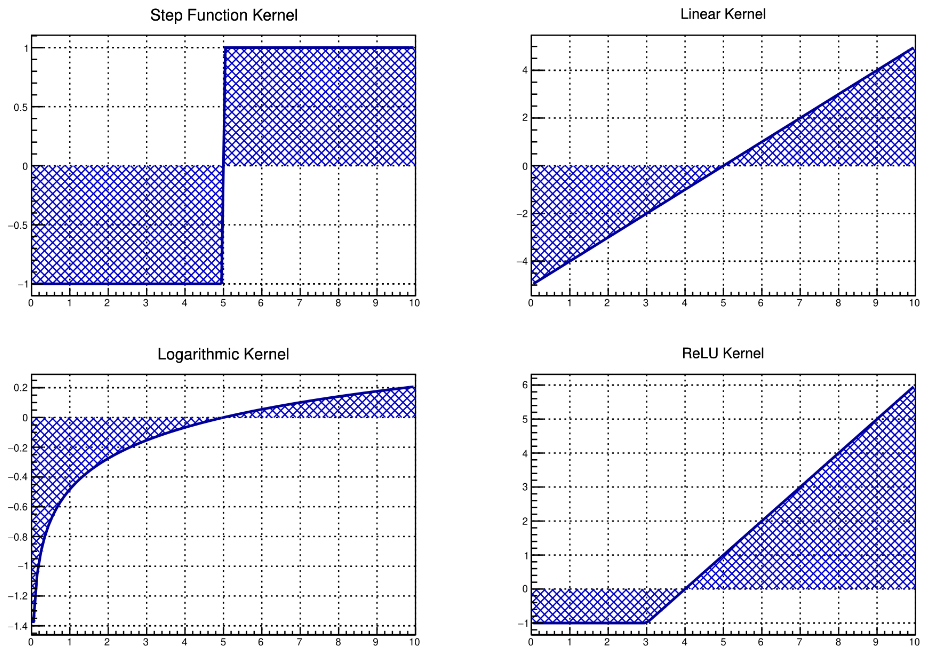


Figure 3.6: Various proposed energy kernel functions. The plots show Φ_{step} with $E_0 = 5$ (upper left), Φ_{linear} with $k = 1, q = -5$ (upper right), $\Phi_{\log}$ with $a = 0.3, b = 0.2, c = 0$ (lower left) and Φ_{ReLU} with $E_0 = 3, e_0 = -1, k = 1$ (lower right).

3.3.4 Distance Weight Function

While the energy kernel function is by definition completely invariant to pixel location, the aim of a distance weight function is to assign pixels with various weights based on their location with respect to the fitted core model, thereby giving their corresponding energy kernel values more or less importance. In this way, the distance function can encode what could be considered a shape model in other applications.

Definition 3.3.1. The distance of a pixel located at point x to a core model $\mathcal{M}$ is defined as the distance between x and line segment ab.

The distance weight function maps the above defined distance to weights from the $[0, 1]$ interval, assigning greater values to pixels which are closer to the core, and lower values to those farther from it. Similarly to energy kernels, several simplistic weight functions are proposed along with their parametrization:

$$w_{\text{step}}(d) = \begin{cases} 1 & \text{if } d < d_0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{where } d_0 \in \mathbb{R} \tag{3.10}$$

$$w_{\text{Gauss}}(d) = \exp\left(-\frac{d^2}{\sigma^2}\right) \qquad \text{where } \sigma \in \mathbb{R} \tag{3.11}$$

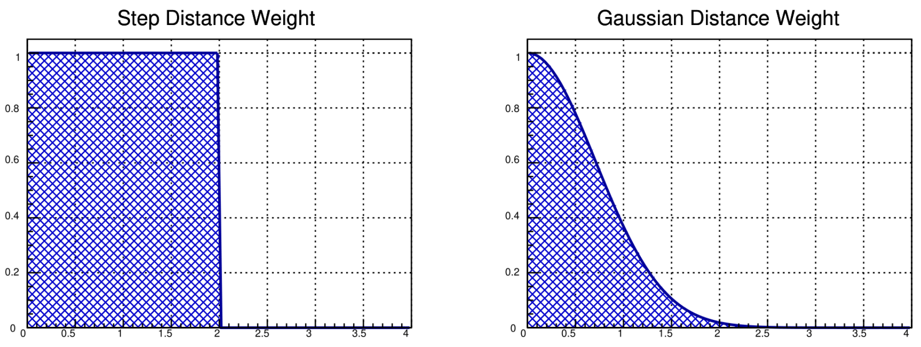


Figure 3.7: Various proposed distance weight functions. The plots show w_{step} with $d_0 = 2$ (left), w_{Gauss} with $\sigma = 3$ (right).

Example plots of the proposed distance weight functions are shown in Figure 3.7.

3.3.5 Proposition Strategy

The proposition strategy Ψ determines how samples $(\vec{a}, \vec{b})$ are drawn. In this work, three strategies are implemented:

Uniform Sampling Strategy The uniform sampling strategy selects pixels sampled independently at random from a uniform discrete distribution.

Informed Sampling Strategy Similarly to the uniform strategy, the informed sampling strategy also samples pixels independently from a discrete distribution. Pixel sampling probability is however proportional to the corresponding energy kernel value Φ, making pixels with greater kernel values more likely to be sampled.

Exhaustive Strategy The exhaustive strategy is the most simple and the only deterministic proposition strategy implemented. It selects all pixel pairs in lexicographic ordering, making it complete but hardly tractable for larger sets of pixels.

The definition of the proposition strategy concludes the definition of the basic fitting procedure. The entire algorithm is recapitulated in Algorithm 3.

3.3.6 Local Optimization

The presented algorithm returns a model, which is fitted from pixels selected by the proposition strategy Ψ. Given a sufficient number of iterations, this approach leads towards convergence to a model $\mathcal{M}^*$ globally maximizing the utility $U_{w,\Phi}$. While this claim holds under the lattice of sampled pixels, if continuous spatial coordinates are considered, the algorithm might possibly return suboptimal solutions. For that reason, an enhancement by means of local optimization has been implemented, leading to algorithm known as *Locally Optimized RANSAC* (or LO-RANSAC) [CMK03].

Algorithm 3 RANSAC

```
1: $U^* \leftarrow -\infty$, $r \leftarrow 1$
2: **repeat**
3:     Select pixel coordinates $\vec{a}^r, \vec{b}^r$ according to the proposition strategy $\Psi$.
4:     Fit model $\mathcal{M}^r \leftarrow (\vec{x}^r_{\text{far}}, \varphi^r, \theta^r)$ from $\vec{a}^r, \vec{b}^r$ by algorithm described in Section 3.3.1.
5:     Evaluate the utility of the model $U^r \leftarrow U_{w,\Phi}(\mathcal{M}^r \mid F)$.
6:     **if** $U^r > U^*$ **then**
7:         Improve $U^* \leftarrow U^r$, $\mathcal{M}^* \leftarrow \mathcal{M}^r$.
8:     $r \leftarrow r + 1$
9: **until** $r \geq r_{\max}$
```

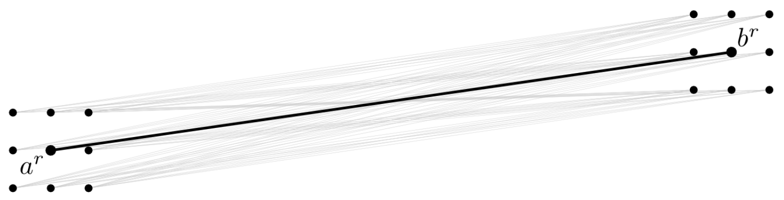


Figure 3.8: The local optimizer window $\mathcal{W}(\mathcal{M})$ evaluated for a model $\mathcal{M}$ (drawn in black). The figure shows $|\Delta|^4 = 81$ similar models (drawn in thin gray) considered during a single iteration of the local optimization procedure for $\Delta = \{-0.2, 0, 0.2\}$.

The modified algorithm is identical to Algorithm 3 up to line 7, where the best model is improved. If a better model $\mathcal{M}$ is discovered, the algorithm considers a set of similar models $\mathcal{W}(\mathcal{M})$ known as *the local optimizer window*. If any of its elements offers a better utility, it replaces $\mathcal{M}$ and repeatedly, a set of its similar models is considered in the next iteration. The algorithm continues in greedy manner until the maximum number of iterations $s_{\max}$ is exhausted or until no further improvement can be performed.

The window function $\mathcal{W}(\mathcal{M})$ maps a model to a set of similar models (example shown in Figure 3.8). In the provided implementation, a finite set of sufficiently small distances Δ is selected. For a model $\mathcal{M}$, $\mathcal{W}(\mathcal{M})$ is defined as the set of all models fitted from points $\vec{a} + \vec{h}$, $\vec{b} + \vec{k}$ such that $\vec{h}, \vec{k} \in \Delta^2$. Note that as $|\mathcal{W}(\mathcal{M})| = \Theta(|\Delta|^4)$, it is not tractable to choose Δ of large cardinality.

With the described modifications, the LO-RANSAC algorithm can greedily pursue improvements throughout its run, adding to its complexity, but also improving the utility of the found solutions and achieving possibly up to subpixel precision in turn. This is consistent with previous findings [Gra+11]. A summary is shown in Algorithm 4.

3.3.7 Simulated Annealing

To improve the convergence rate and decrease the probability of producing suboptimal models in the LO-RANSAC local optimization component, a variant of the algorithm based on the simulated annealing scheme has been implemented [KGV83].

Algorithm 4 LO-RANSAC

```
 1: $U^* \leftarrow -\infty$, $r \leftarrow 1$
 2: repeat
 3:     $s \leftarrow 1$
 4:     Select pixel coordinates $\vec{a}^r, \vec{b}^r$ according to the proposition strategy $\Psi$.
 5:     Fit model $\mathcal{M}^{r,s} \leftarrow (\vec{x}^r_{\text{far}}, \varphi^r, \theta^r)$ from $\vec{a}^r, \vec{b}^r$ by algorithm from Section 3.3.1.
 6:     Evaluate the utility of the model $U^{r,s} \leftarrow U_{w,\Phi}(\mathcal{M}^{r,s} \mid F)$.
 7:     if $U^{r,s} > U^*$ then
 8:         repeat
 9:             $\mathcal{M}^{r,s+1} \leftarrow \text{argmax}_{\mathcal{N} \in \mathcal{W}(\mathcal{M}^{r,s})} U_{w,\Phi}(\mathcal{N} \mid F)$
10:             $U^{r,s+1} \leftarrow \max_{\mathcal{N} \in \mathcal{W}(\mathcal{M}^{r,s})} U_{w,\Phi}(\mathcal{N} \mid F)$
11:             $s \leftarrow s + 1$
12:         until $|U^{r,s} - U^{r,s-1}| < \varepsilon$ or $s \geq s_{\max}$
13:         Improve $U^* \leftarrow U^{r,s}$, $\mathcal{M}^* \leftarrow \mathcal{M}^{r,s}$.
14:     $r \leftarrow r + 1$
15: until $r \geq r_{\max}$
```

Inspired by the cooling process of a thermodynamic system, simulated annealing tracks the energy state of the local optimizer (or *temperature*) which determines its ability to transition into neighboring models of possibly greater utility. The local optimizer is initialized in a state of high energy and over time, the system simulates energy loss (hence the name annealing), gradually transitioning into states of lower energy and decreasing the ability to make further transitions. Eventually, the lowest energy state is reached and the optimization process converges on the final model.

Note that unlike LO-RANSAC, transitions occurring in the simulated annealing scheme are probabilistic and may not necessarily always lead into neighbor models of greater utility. This is justified as means of the algorithm to overcome suboptimal extrema. The key component controlling this behavior is the acceptance probability function $\mathcal{P}(U, U', T)$ and the temperature function $\tau(t)$. Based on the thermodynamic analogy, Kirkpatrick proposed the acceptance probability to be defined as $\mathcal{P} : \mathbb{R}^3 \to [0, 1]$ such that

$$\mathcal{P}(U, U', T) = \begin{cases} 1 & \text{if } U' > U \\ \exp\left(-\frac{U-U'}{T}\right) & \text{otherwise} \end{cases} \tag{3.12}$$

Under this definition, the optimizer always accepts all utility improvements, while smaller utility decreases are accepted with a greater probability than large decreases. Furthermore, the probability of accepting any utility decrease at all is inversely proportional to the temperature of the system.

The temperature is determined by a nonincreasing function $\tau : [0, 1] \to \mathbb{R}$ (sometimes referred to as *the annealing schedule*). In the implementation, two widely used annealing schedules are available:

$$\tau_{\exp}(t) = T_0 \alpha^t \qquad \text{where } T_0 \in \mathbb{R}, \alpha \in [0,1] \qquad (3.13)$$

$$\tau_{\text{linear}}(t) = T_0 - \eta t \qquad \text{where } T_0, \eta \in \mathbb{R} \qquad (3.14)$$

Retrospectively, the LO-RANSAC algorithm can be viewed as a special case of simulated annealing where $\tau(k) = 0$ for all k, forcing the optimizer to reject all downhill transitions, greedily pursuing utility improvements only. On the other hand, when $\tau(k) = 1$ for all k, the optimizer executes the Metropolis-Hastings step [Met+53] of a MCMC algorithm, simulating a random walk on the graph of adjacent models. As a summary, the simulated annealing variant of RANSAC is presented in Algorithm 5.

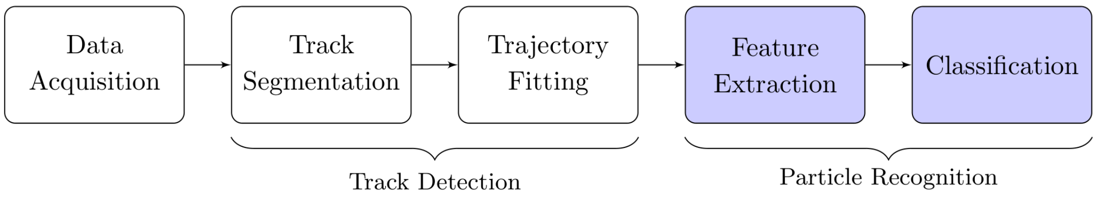
Algorithm 5 SA-RANSAC

```
1: $U^* \leftarrow -\infty$, $r \leftarrow 1$
2: **repeat**
3:     $s \leftarrow 1$
4:     Select pixel coordinates $\vec{a}^r, \vec{b}^r$ according to the proposition strategy $\Psi$.
5:     Fit model $\mathcal{M}^{r,s} \leftarrow (\vec{x}^r_{\text{far}}, \varphi^r, \theta^r)$ from $\vec{a}^r, \vec{b}^r$ by algorithm from Section 3.3.1.
6:     Evaluate the utility of the model $U^{r,s} \leftarrow U_{w,\Phi}(\mathcal{M}^{r,s} \mid F)$.
7:     **if** $U^{r,s} > U^*$ **then**
8:         Set accepted hypothesis index $q \leftarrow 1$.
9:         **repeat**
10:            Calculate the system temperature $T \leftarrow \tau(s)$.
11:            Increment hypothesis index $s \leftarrow s + 1$.
12:            Select $\mathcal{M}^{r,s} \leftarrow$ random element of $\mathcal{W}(\mathcal{M}^{r,q})$.
13:            Evaluate the utility of the model $U^{r,s} \leftarrow U_{w,\Phi}(\mathcal{M}^{r,s} \mid F)$.
14:            **if** accepted with probability $\mathcal{P}(U^{r,q}, U^{r,s}, T)$ **then**
15:                Set accepted hypothesis index $q \leftarrow s$.
16:        **until** $s \geq s_{\max}$
17:        Improve $U^* \leftarrow U^{r,q}$, $\mathcal{M}^* \leftarrow \mathcal{M}^{r,q}$.
18:    $r \leftarrow r + 1$
19: **until** $r \geq r_{\max}$
```

Chapter **4**

Particle Recognition

After detection, fitted instances of the core model undergo recognition by classification. This chapter gives information on the selected features, the feature extraction procedure and the used classification method.



The recognition task is divided in two separate subtasks:

Feature Extraction: Given a set of detected particles with corresponding models and frames $(\mathcal{M}^1, F^1), (\mathcal{M}^2, F^2), \ldots, (\mathcal{M}^N, F^N)$, produce corresponding feature descriptors $x^1, x^2, \ldots, x^N \in \mathcal{X}$.

Classification: For a set of feature descriptors $x^1, x^2, \ldots, x^N \in \mathcal{X}$, assign semantic labels $y^1, y^2, \ldots, y^N \in \mathcal{Y}$.

Looking past the above division, the main purpose of the recognition component is to label detected core models with the corresponding particle species. While in theoretical physics this term generally refers to a broad domain of possibilities, including elementary and composite particle types, in the context of Timepix detectors it may be significantly simplified due to the fact that not necessarily all particles are observable within TPX frames. For that reason, the proposed model contains the following classes:

p^+ This class corresponds with protons.

e^- This class corresponds with electrons.

γ This class corresponds with photons.

$^A_Z X$ Multiple classes parameterized by $A, Z \in \mathbb{N}$. An instance of this class corresponds with an atom of element X defined by the atomic number Z and mass number A (e.g. carbon for $Z = 6$, $A = 12$).

For later reference, the listed classes constitute the discrete set $\mathcal{Y}$ of semantic labels.

4.1 Feature Selection

This work presents a conventional approach to feature extraction, wherein feature descriptors x^j are vectors $(x^j_1, x^j_2, \ldots, x^j_n)$ for all $j \leq N$. The components $\{x^j_i\}^n_{i=1}$ of each vector are given by a set of corresponding functions $\{f_i\}^n_{i=1}$ such that $x^j_i = f_i(\mathcal{M}^j \mid F^j)$ for all $i \leq n$, $j \leq N$. The functions $\{f_i\}^n_{i=1}$ are referred to as *features*.

Since classification is performed based on the feature descriptors alone, features must be carefully chosen to be representative of the set of labels $\mathcal{Y}$. While in the past, morphological features were used to obtain viable classification of particle species [Hol+08; Vil+11], it is the aim of this work to propose features utilizing the energy deposition information available in the ToT operation mode as such features have already been used with success in more recent works [Hoa+12; Hoa+14]. For their derivation, the physics of particle interaction with TPX detector shall be revisited.

The theory of the energy loss process has been described by Bohr, Bethe, Bloch and Landau [Boh13; Bet30; Blo33; Lan44]. For the purposes of this work, it can be summarized as follows. During the traversal of a charged particle through a solid medium such as the TPX sensor layer, energy losses occur due to single elastic collision with the atomic electrons. In each such collision, energy W is transferred. Depending on whether the charged particle is completely absorbed or not, two possible cases may be distinguished.

4.1.1 Energy Loss of Particles at the End of Their Range

If the charged particle is completely absorbed in the medium, its energy loss (also referred to as *the stopping power*) is given by the Bethe formula: [Bet30; Frö15]

$$\frac{dE}{dx} = \frac{4\pi e^4 z^2}{m_e v^2} NZ \left[\ln \frac{2 m_e v^2}{I} - \ln\left(1 - \frac{v^2}{c^2}\right) - \frac{v^2}{c^2}\right]. \tag{4.1}$$

Here, e denotes the elementary charge, m_e electron rest mass, v and z is the velocity and charge of the incoming particle. Z is the atomic number, N the number density and I the mean excitation energy of the absorber material.

During the traversal of the medium, the velocity of the charged particle decreases, in turn affecting the value of $\frac{dE}{dx}$. This gives rise to a characteristic curve referred to as *the Bragg peak* (shown in Figure 4.1), which is present for any charged particle stopping in the sensor material.

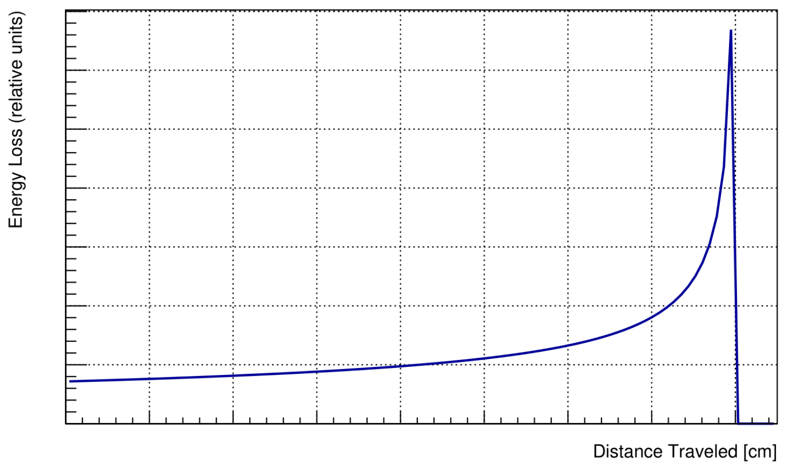


Figure 4.1: Analytical approximation of the Bragg peak for protons of momentum from 10 to 200 MeV/c in a solid medium [Bor97].

4.1.2 Energy Loss of Particles with Intermediate Energy

If the charged particle is not completely absorbed in the medium, it has been shown that the energy loss is constant. For intermediate energies ($\beta\gamma \in [0.1, 1000]$) and intermediate Z-materials, the Bethe-Bloch formula reads as [OG+14]:

$$\left\langle -\frac{dE}{dx} \right\rangle = Kz^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} \ln \frac{2m_e c^2 \beta^2 \gamma^2 W_{\max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right]. \tag{4.2}$$

In the formula, M denotes the incident particle mass, z its charge and $K = 4\pi N_A r_e^2 m_e c^2 = 0.037\ \mathrm{mol}^{-1}\mathrm{cm}^{-2}$. Furthermore, I denotes the mean excitation energy of the absorber material, Z its atomic number and A its atomic mass. $\delta(\beta\gamma)$ is a density effect correction to the ionizing energy loss. $W_{\max}$ is the maximum energy in a single collision given by:

$$W_{\max} = \frac{2m_e c^2 \beta^2 \gamma^2}{1 + 2\gamma m_e/M + (m_e/M)^2}. \tag{4.3}$$

While I, Z and A corresponding with the TPX detector hardware are assumed to be known as well as universal constants, M, z and β can be viewed as a part of the hidden state of the incident particle.

4.1.3 Selected Features

Since in both presented cases, information about particle species is linked to the energy loss (this is shown in Figure 4.4), energies observed in TPX frames may yield discriminative features for particle species classification.

Unfortunately, it has not been shown to be possible to directly regress[8] parameter values corresponding with the incident particle without other measurements or a priori assumptions. As noted by other authors, fitting such models is a computationally intensive task which produces less than optimal results [SP18]. Furthermore, for every detected particle, two possible models (Bethe-Bloch and Bragg) have to be considered leading to an additional layer of reasoning, which might convolute the accuracy of the results.

For these reasons, a different approach to feature selection has been adopted. Instead of attempting to learn the parameters of the two models, the features used for particle species classification are energy loss values sampled by a function δ from the frame F over various intervals $[p_i, q_i]$ along the projected particle trajectory defined by a core model $\mathcal{M}$.

$$f_i(\mathcal{M} \mid F) = \delta(\rho(p_i \mid \mathcal{M}), \rho(q_i \mid \mathcal{M}) \mid F) \qquad \text{where } i \leq n \tag{4.4}$$

$$\rho(r \mid \mathcal{M}) = \begin{cases} (1-r)\vec{x}_{\text{near}} + r\vec{x}_{\text{far}} & \text{for FA direction} \\ (1-r)\vec{x}_{\text{far}} + r\vec{x}_{\text{near}} & \text{for TA direction} \end{cases} \qquad \text{for } r \in [0,1] \tag{4.5}$$

The parameters $n, m \in \mathbb{N}$ and $\{p_i\}_{i=1}^n$, $\{q_i\}_{i=1}^n$ are called *a feature model* and determine the volume and the precision of the sampled information. Appropriate choice of their values is discussed later in this section.

In a desirable feature model, redundant information may be efficiently compressed by choosing a small number of wide intervals in regions where δ is nearly constant, and a large number of narrow intervals in regions of increased fluctuation.

4.2 Feature Extraction

In order to evaluate the sampling function $\delta(\vec{h}, \vec{k} \mid F)$ in a frame F over a portion of linear particle trajectory bounded by points h and k, an elaborate procedure is employed to improve the accuracy and to increase the invariance of the calculated result against the effects of charge sharing.

In theory, the energy loss along particle trajectory may be directly sampled from F by integrating pixels along the projected path. For this, the frame image function f_F can be used.

$$\delta(\vec{h}, \vec{k} \mid F) = \frac{1}{d_E(\vec{h}, \vec{k})} \int_{s=0}^{1} f_F((1-s)\vec{h} + s\vec{k})ds. \tag{4.6}$$

This approach, albeit correct in theory, does not yield sufficiently accurate results in practice. Due to charge sharing, the energy loss $\frac{dE}{dx}$ is deposited in a multitude of

[8]Note that direct regression of parameters M, z and β of the Bethe-Bloch Formula may yield an analytical solution to the task, rendering any subsequent classification efforts unnecessary.

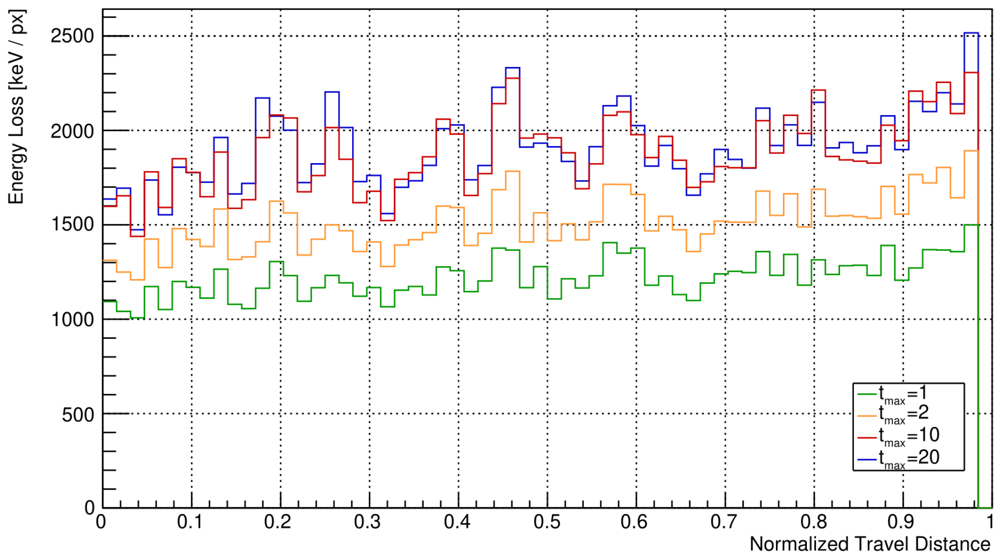


Figure 4.2: Effect of the $t_{\max}$ parameter. The plot shows $\frac{dE}{dx}$ of $^{16}_{8}$O ion at 430 MeV/c sampled on 64 uniformly spaced intervals of equal size with $e_0 = 10$ keV and various values of $t_{\max}$ (see legend).

adjacent pixels instead of just one. For that reason, a wider neighborhood possibly spanning the entire core region has to be considered at every point of the integral.

Let $\vec{v} = \vec{k} - \vec{h}$ and $\vec{n}$ be a normal vector to $\vec{v}$ of unit size. Then, $\frac{dE}{dx}$ is sampled by $\delta(\vec{h}, \vec{k} \mid F)$ as follows:

$$\delta(\vec{h}, \vec{k} \mid F) = \frac{1}{d_E(\vec{h}, \vec{k})} \int_{s=0}^{1} \int_{t=-t_{\max}}^{t_{\max}} f_F((1-s)\vec{h} + s\vec{k} + t\vec{n}) dt ds \tag{4.7}$$

This modification effectively increases the number of energy values integrated, resulting in overall larger values (shown in Figure 4.2). Unfortunately, due to this effect, the modified integral may also include undesirable energies from other particles in the frame. For this reason, a rudimentary suppression mechanism is included, resulting in further modification.

$$\delta(\vec{h}, \vec{k} \mid F) = \frac{1}{d_E(\vec{h}, \vec{k})} \int_{s=0}^{1} \int_{t=-t_{\max}}^{t_{\max}} w(f_F(\xi(s,t))) f_F(\xi(s,t)) dt ds \tag{4.8}$$

$$\xi(s,t) = (1-s)\vec{h} + s\vec{k} + t\vec{n} \tag{4.9}$$

As in Expression 3.5, $w : \mathbb{R} \to [0, 1]$ denotes a weight function. Here, however, the purpose of the function is to increase the precision of the calculation by suppressing

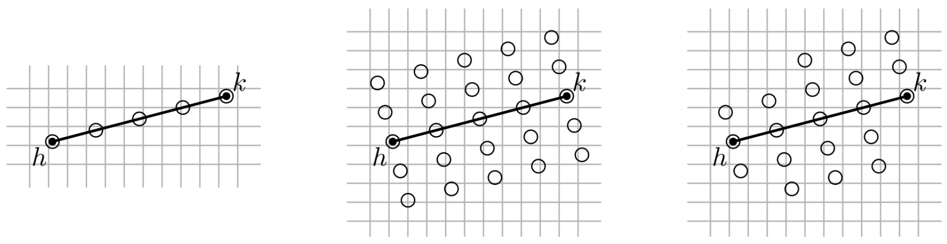


Figure 4.3: Illustration of the difference between various proposed δ sampling functions. Expression 4.6 (shown in the left) integrates pixel values along the projected trajectory. Expression 4.7 (shown in the center) integrates pixel values in a certain neighborhood of the trajectory. Expression 4.8 (shown in the right) also considers the neighborhood, suppressing low-energy pixel values.

energies from pixels ionized by other particles than the particle described by the core model. For that purpose, the function takes energy values instead of distance.

For the scope of this work, it is sufficient to define w as a step function:

$$w(e) = \begin{cases} 0 & \text{if } e < e_0 \\ 1 & \text{otherwise} \end{cases} \tag{4.10}$$

For illustration, a comparison between all three presented sampling methods is shown in Figure 4.3. As it is the most general, the δ function defined in Expression 4.8 is used for feature extraction from this point onward. An example of sampled values from various classes for possible use as classifier features is shown in Figure 4.4.

4.2.1 Performance Improvements

For feature models consisting of 2^k uniformly spaced intervals of equal width, the performance of multi-scale feature extraction can be enhanced. In situations where k is to be determined empirically, it is not necessary to perform feature extraction for each value of k independently.

Instead, feature extraction may be performed only for the largest $n = 2^k$. Then, given a feature descriptor $(x_1^j, x_2^j, \ldots, x_n^j) \in \mathcal{X}$ an alternate descriptor $(u_1^j, u_2^j, \ldots, u_w^j) \in \mathcal{X}'$ for $w = 2^{k-1}$ can be obtained as follows.

$$u_i = \frac{x_{2i} + x_{2i+1}}{2} \qquad \text{for } i \leq w \tag{4.11}$$

With this enhancement, a multitude of feature models may be efficiently tested at different scales.

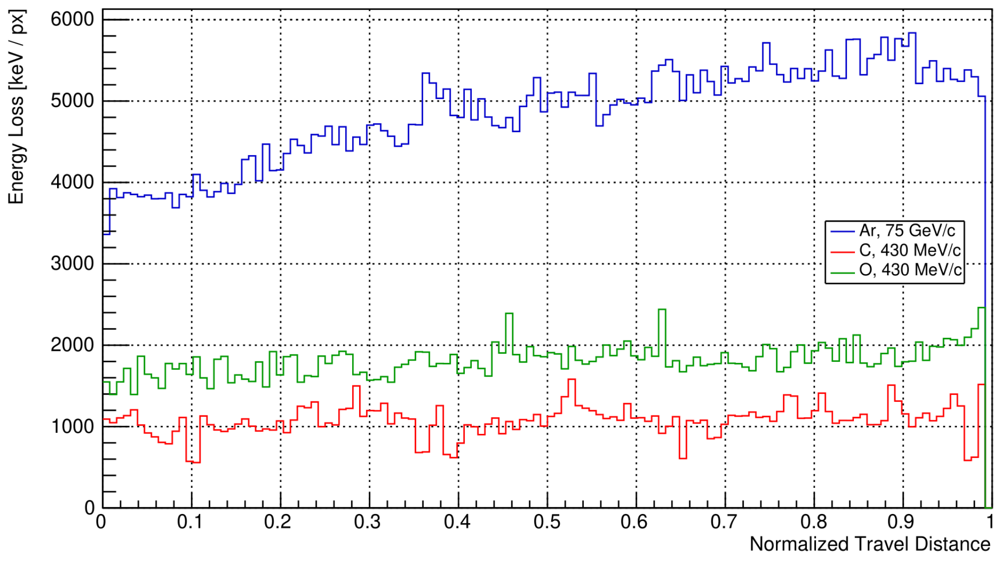


Figure 4.4: $\frac{dE}{dx}$ values of various ions (see legend) sampled on 128 uniformly spaced intervals of equal size using the presented method with $t_{\max} = 6$, $e_0 = 10$ keV.

4.3 Classification

For classification, a conventional supervised learning paradigm is used. This requires ground truth to be provided *a priori* in the form of a training set $\mathcal{D} = \{(x^j, y^j) \mid j \leq N\}$ containing correctly classified samples. A classifier is first presented with samples from $\mathcal{D}$ to *learn* its internal model. Later, the classifier uses the internal model to make classification decisions about unknown samples $\mathcal{X}$.

The learning process is motivated by the notion that with a set $\mathcal{D}$ containing sufficiently large number of independent training samples, a learned classifier is capable of producing accurate class labels for previously unobserved feature descriptors.

4.3.1 Nearest Neighbors

The selected classification method for the feature set proposed in the previous sections is the *k-nearest neighbor classifier* (abbreviated as k-NN) [Bar12]. The selection of this particular algorithm is prompted by its consistently high performance and by its lack of requirements on the knowledge of a priori distributions from which training samples are drawn. In addition, k-NN classifiers make no attempts at inference from the training data, which is desirable given the nature of the selected features.

The k-NN classification method requires a well-defined dissimilarity function d which can be thought of as a distance metric in the domain of feature descriptors. In addition, a number $1 \leq k \leq N$ must be specified. During the classification of a descriptor x,

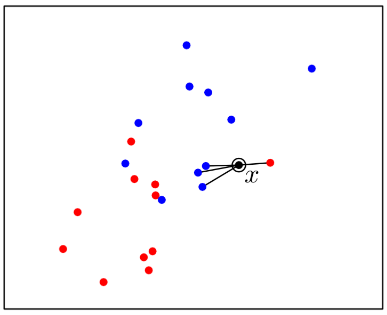


Figure 4.5: Example of 4-NN classification in binary environment with 2 features. The sample x is classified into the blue class because 3 out of its 4 nearest neighbours are blue, whereas only 1 neighbor is red.

dissimilarity is calculated between x and all training samples x^j, $j \leq N$. The k samples with the lowest dissimilarity (referred to as the nearest neighbors) then vote with their labels y^j and the sample x is classified with the label of the greatest support. This procedure is summarized in Algorithm 6.

Algorithm 6 k-Nearest Neighbor Classification

```
 1: for training sample $(x^j, y^j) \in \mathcal{D}$ do
 2:     Calculate dissimilarity $d^j \leftarrow d(x, x^j)$.
 3: Initialize empty voting set $V \leftarrow \{\}$.
 4: Let $C(y) \leftarrow 0$ for all class labels $y$.
 5: $r \leftarrow 1$
 6: repeat
 7:     Find the $r$-th nearest neighbor $v^r \leftarrow \text{argmin}_{j \leq N, j \notin V} d^j$.
 8:     Append $V \leftarrow V \cup \{v^r\}$.
 9:     Vote $C(y^{v^r}) \leftarrow C(y^{v^r}) + 1$.
10:     $r \leftarrow r + 1$
11: until $r \geq k$
12: Label $x$ with $\text{argmax}_y C(y)$.
```

In the context of this work, the dissimilarity function d of the k-NN classifier is defined as a conventional Euclidean metric d_E generalized to arbitrary number of dimensions.

The choice of a suitable neighbor count k is usually small and odd to break ties. While $k = 1$ enables use of various optimized data storage approaches, increasing the speed of the algorithm, choosing larger $k << N$ may improve the robustness of its output, requiring the consensus of multiple data points. Lastly, setting $k = N$ includes all training samples in the vote, rendering the dissimilarity function d completely irrelevant. In the next chapter, the value of k is determined empirically through cross-validation.

4.3.2 Performance Improvements

While an obvious advantage of the k-NN algorithm is zero training time, a drawback presents itself in the fact that learned classifiers are required to keep their entire training sets, thereby increasing memory consumption significantly. In addition, classification of a single sample requires the calculation of $\mathcal{O}(N)$ dissimilarities for a training set of possibly large size N.

For these reasons, dedicated data structures are used in order to achieve better performance. In particular, the k-NN implementation in this work uses k-d trees and ball trees [Ben75; Omo89]. Both of these data structures use various tree-based storage layouts to offer sublinear lookups of the nearest neighbors (resulting in speedup at the line 7 of Algorithm 6). This, however, also increases classifier training complexity due to the overhead caused by tree construction, making the improvement only viable for a small number of training cycles and a large number of classifications.

Chapter 5

Experiments

To get a better understanding of the behavior of the presented detection and recognition methods, a series of experiments was performed. This chapter gives details on the experimental data used, evaluated metrics, setup and results of the individual experiments.

5.1 Experimental Data

Due to the lack of a sufficiently accurate[9] simulation software, experimentally measured TPX data was used to evaluate the presented algorithms. The data consists of 75 datasets measured at the Heidelberg Ion Therapy Center[10] by researchers from IEAP CTU[11]. The experimental setup included a TPX detector equipped with a single ${}^{28}_{14}\mathrm{Si}$ sensor layer of thickness 1 mm. The bias voltage was set to approximately 500 V and the acquisition time was set to 0.1 s.

During measurements, charged particles were hitting the detector at various angles. The results are categorized in datasets, each constituted by multiple files containing frames taken with a specific beam configuration (particle species and momentum) and incidence angle. In the ground truth data set, the following particle classes are present: p^+, ${}^{4}_{2}\mathrm{He}$, ${}^{12}_{6}\mathrm{C}$, ${}^{16}_{8}\mathrm{O}$, ${}^{36}_{18}\mathrm{Ar}$.

Because of noise induced in the TPX detector by the background environment and interactions of the particle beam with surroundings of the experimental setup, datasets were processed before evaluation could be performed. Undesirably, frames contained tracks not necessarily limited to those produced by the particle beam. Ground truth data was therefore created by manually annotating frames from selected datasets with core

[9]Commonly available particle physics simulation systems such as GEANT4 [Ago+03] do not simulate charge sharing between pixels of TPX detectors. This cannot be neglected as the proposed track detection methods in part rely on this phenomenon for producing accurate results.

[10]Heidelberger Ionenstrahl-Therapiezentrum (HIT), Im Neuenheimer Feld 450, 69120 Heidelberg, `https://www.klinikum.uni-heidelberg.de/index.php?id=113005&L=1`

[11]Institute of Experimental and Applied Physics, Czech Technical University, `https://utef.cvut.cz`

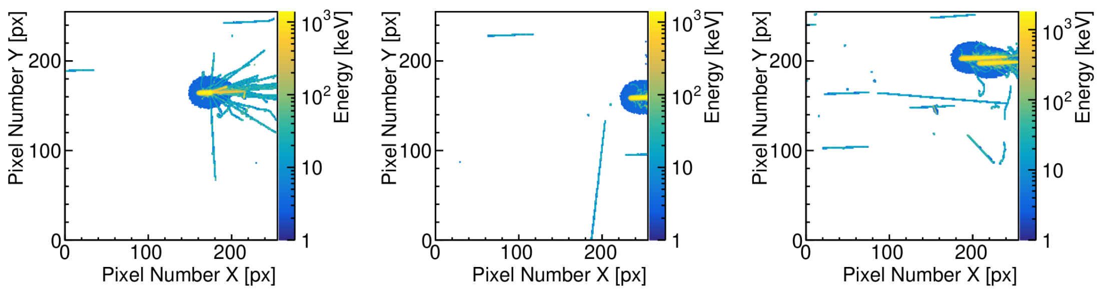


Figure 5.1: Examples of 3 frames containing excluded tracks. From the left to the right: particle collision, cropped track and overlap of two tracks.

model parameters. Even though this approach may not be as accurate as e.g. obtaining ground truth by means of artificial simulation, human-annotated model parameters are considered to be sufficiently accurate for the purposes of this work.

To increase the quality of the ground truth data, the following tracks were excluded (illustrated in Figure 5.1):

1. tracks likely produced by other particles than those of the beam (judged by energy loss, morphology, incidence angle and azimuth),
2. tracks of particles colliding within the detector,
3. tracks cropped by the edge of the frame,
4. tracks overlapping with other tracks.

In total, 937 samples have been manually prepared. The annotated tracks were stored along with model parameters hierarchically in a sparse list of pixels encoded in the ROOT binary format for further manipulation [Ant+09].

5.2 Detection Evaluation

In evaluation of the proposed track detection methods, two metrics are tracked – the failure rate and the accuracy.

Failure Rate: This metric describes a relative quantity of erroneous detections in the output of the tested method. Such detections are distinguished in two categories corresponding to errors in binary classification – either an actual particle is not detected (resulting in a *false negative*) or a particle is detected extraneously (yielding a *false positive*).

The value of failure rate is conventionally defined as the percentage of false detections in all detections:

$$FR = \frac{FP + FN}{TP + TN + FP + FN} \tag{5.1}$$

Here, the numbers FP and FN abbreviate the two aforementioned quantities. In the context of particle detection, *true negative* detections are not relevant, yielding $TN = 0$. Lastly, TP denotes the *true positive* detection count (i.e. correctly detected particles).

Error: Error of a parameter v is expressed as the root mean squared error of individual estimated model parameters $\hat{v}_i$ with respect to their ground truth values v_i.

$$E(v) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_i - \hat{v}_i)^2} \tag{5.2}$$

This way, parameters $x_{\text{far},1}$, $x_{\text{far},2}$, φ and θ of the particle trajectory model are evaluated.

5.2.1 Trajectory Fitting

The first experiment examines the behavior of the presented detection algorithms in a very basic setting comprised of frames containing only one track produced by a single particle. Although such a scenario is not likely to be often observed in reality, it is desirable for the purposes of evaluation. As the lack of overlaps renders segmentation unnecessary, it provides a valuable insight into the performance of proposed trajectory fitting methods.

For the experiment, 1,000 synthetic frames were generated from the ground truth data. This quantity is considered to be a good compromise between tractability and accuracy. In each frame, a track has been chosen uniformly at random from the processed ground truth datasets. The track has been placed at a random location within the pixel matrix such that it is not cropped by the border of the frame in any direction. To avoid biased results, the orientation of the track has been randomized as well. However, in order to avoid interpolation of pixel values, the angle of rotation has been quantized to 90 degrees. Several examples of the generated synthetic frames is shown in Figure 5.2.

The tested algorithms consist of RANSAC, LO-RANSAC and SA-RANSAC. Their selected configuration parameters are listed in Table 5.1.

Since only one track is present in every frame, the calculation of evaluation metrics may be simplified to improve performance. If the tested algorithm produced no detection at all, the instance is regarded as false negative. If the algorithm produced exactly one detection, the instance is regarded as true positive provided that the detected trajectory is sufficiently close to its ground truth counterpart. Lastly, if more than one detection

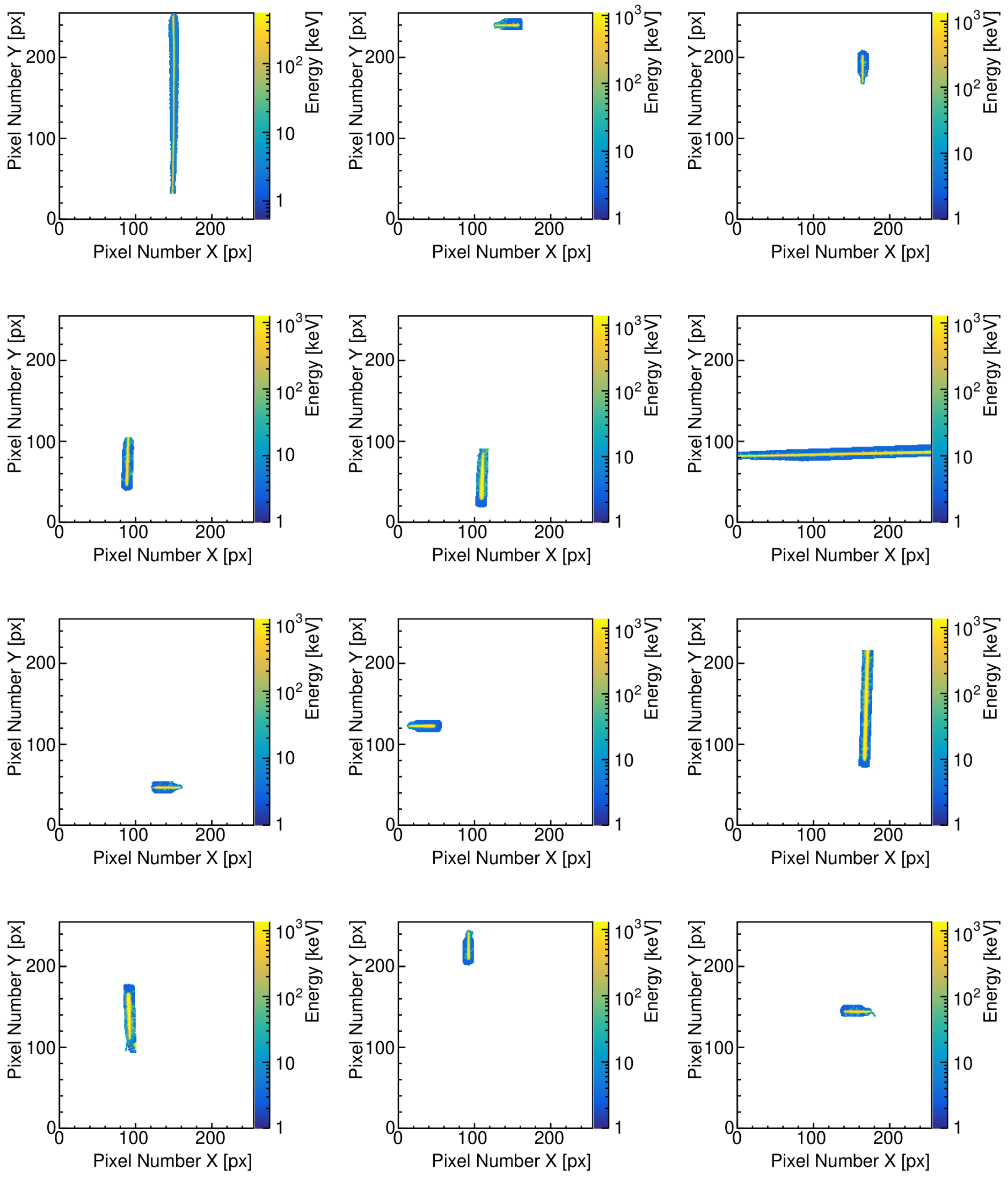


Figure 5.2: 12 synthetic TPX frames used for evaluation of detection methods in the trajectory fitting experiment.

Parameter	RANSAC	LO-RANSAC	SA-RANSAC
Proposition Strategy Ψ	informed	informed	informed
Energy Kernel Φ	ReLU	ReLU	ReLU
Distance Weight w	Gaussian	Gaussian	Gaussian
Sample Count	2,000	2,000	2,000
Local Optimizer	none	greedy	sim. annealing
L. O. Iteration Count	N/A	100	1,000
L. O. Window Δ	N/A	$\{0, \pm 0.2\}$	$\{0, \pm 0.2\}$
Annealing Schedule	N/A	N/A	linear

Table 5.1: Parameters of the evaluated algorithms.

is produced by the algorithm, the closest detection is assumed to be true positive while others are labeled false positives.

Since all algorithms are randomized, each frame has been repeatedly processed 3 times to suppress random fluctuations. The results shown in Table 5.2 are mean values over all runs.

Algorithm	$\langle FR \rangle$	$\langle E(x_{\text{far},1}) \rangle$	$\langle E(x_{\text{far},2}) \rangle$	$\langle E(\varphi) \rangle$	$\langle E(\theta) \rangle$	$\langle t \rangle$ [ms]
RANSAC	0.022	3.227	2.821	3.047	10.412	1,012.7
LO-RANSAC	0.021	2.898	2.873	2.886	11.863	5,538.6
SA-RANSAC	0.021	2.721	2.908	2.856	12.582	1,762.5

Table 5.2: Results of the trajectory fitting experiment.

The results indicate that all three algorithms have yielded outputs of comparable quality. Overall, the low failure rate shows that they have successfully achieved their main objective. While spatial errors in the directions of the X- and Y-axis seem to be consistent, angular trajectory parameter errors differ significantly. In particular, the azimuth error is consistently smaller than the incidence angle error, possibly hinting at poor resolution of the reconstructed trajectory in the direction of the Z-axis.

Of the three, the fastest algorithm was RANSAC. This was expected since the other two algorithms extend RANSAC by additional local optimization. The effects of the optimization on reducing parameter errors seem to be observable between the locally-optimized and non-optimized algorithms. However, the magnitude of the RMSE improvement is almost negligible and within the order of random fluctuations. This is consistent with the observation that given sufficient number of samples, local optimization may only yield a small[12] RMSE improvement over a non-optimized algorithm.

[12]The upper bound on the spatial RMSE improvement is given by $1 - \min \Delta = 0.8$ provided that a sufficient number of samples is drawn.

5.2.2 Overlap Detection

While the first experiment examined the behavior of the presented trajectory fitting algorithms in a very clean and predictable environment, the purpose of the second experiment is to evaluate their robustness under noisy conditions more similar to practical measurements. This is accomplished by testing them alongside the proposed track segmentation methods in frames of larger occupancy. As more overlaps occur more frequently, the quality of the output is expected to decrease. The goals of this experiment are to confirm this expectation and to determine the overall robustness of the algorithm with respect to track overlaps.

To effectively examine the behavior of the tested algorithms, an approach similar to that of the previous experiment has been adopted. Synthetic test datasets of frames have been generated with varying number of tracks $n \in \{5, 10, 20, 50\}$. These can be viewed as simulations of actual measurements with a TPX detector taken with increasing values of the acquisition time parameter.

The tested algorithms use the same configuration parameter values as in the previous experiment (see Table 5.1). For segmentation, the parameters used for all algorithms are listed in Table 5.3.

Segmentation Method	Parameter	Value
Flood Fill	Connectivity Threshold e_0	10^{-3} keV
Flood Fill	Propagation Neighborhood $\mathcal{N}$	8-neighborhood
Hough Transformation	Inlier Distance Threshold	7 pixels
Hough Transformation	Accumulator Threshold $A_{\min}$	5,000
Hough Transformation	Maximum Iterations $s_{\max}$	100

Table 5.3: Parameters of the segmentation methods (same for all algorithms).

In evaluation of every frame, a matching between the set of ground truth detections and detections provided by the algorithm is constructed. In order to be matched, a pair of detections must have sufficiently small distance. While successfully matched pairs are regarded as true positives, unmatched ground truth detections are treated as false negatives and unmatched detections provided by the tested algorithms are labeled as false positives.

As in the previous experiment, each generated testing set contains 1,000 frames and for all frames, algorithms are executed repeatedly 3 times to suppress randomized fluctuations. The results of the experiment are shown in Table 5.4.

Similarly to the previous experiment, all three algorithms have produced results of comparable quality. As expected, the failure rate increases along with the number n of tracks present in the frame. Overall, its values indicate that the tested methods succeed in the majority of cases if less than 20 tracks are present in the frame. As in the previous experiment, spatial errors are consistent and increase with n, whereas angular errors behave differently of each other. While the azimuthal error increases with n, the

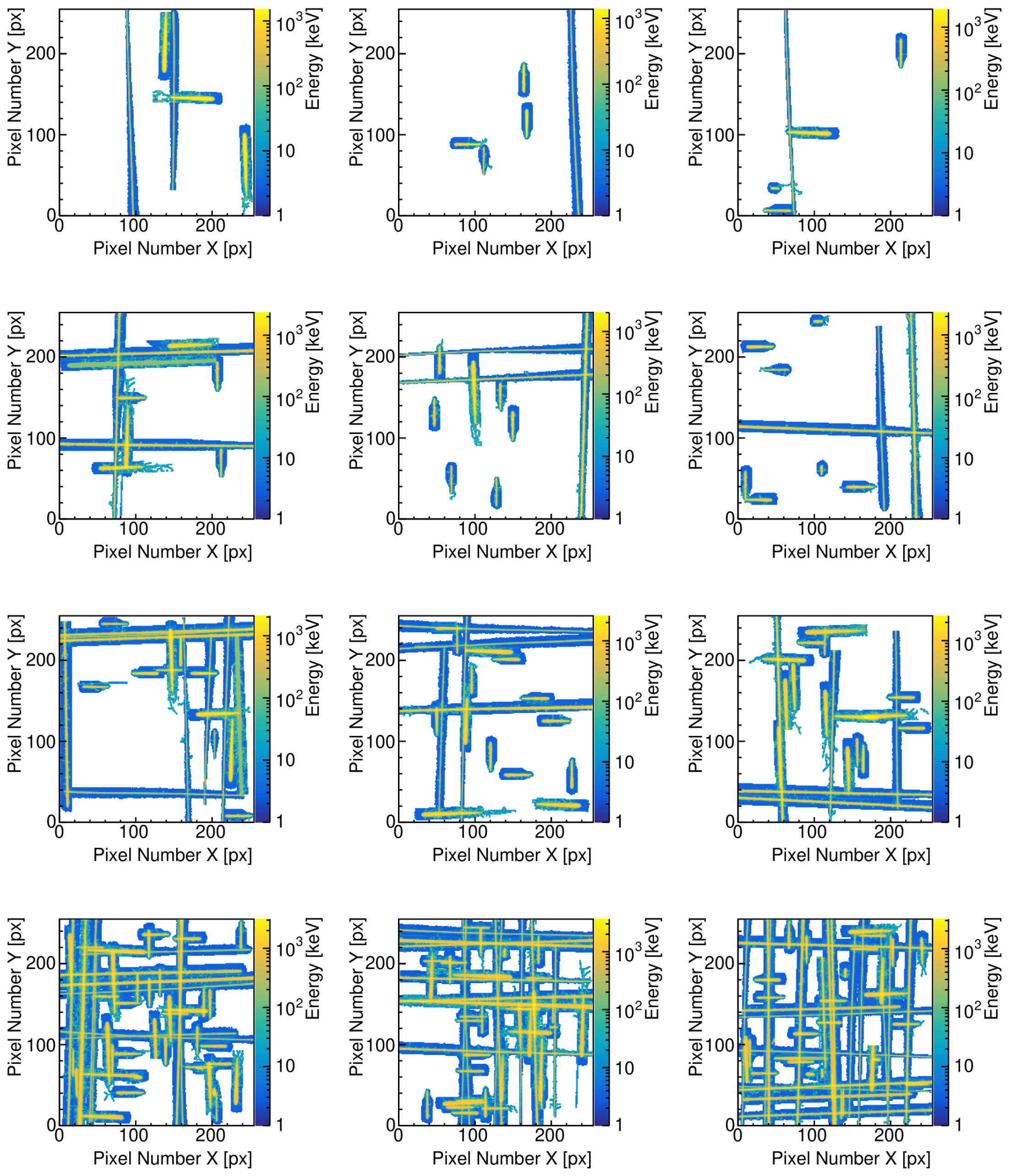


Figure 5.3: 12 synthetic TPX frames used for evaluation of detection methods in the overlap detection experiment. In every row from the top to the bottom, 3 frames are shown for particle counts $k \in \{5, 10, 20, 50\}$ respectively.

Algorithm	n	$\langle FR \rangle$	$\langle E(x_{\mathrm{far},1}) \rangle$	$\langle E(x_{\mathrm{far},2}) \rangle$	$\langle E(\varphi) \rangle$	$\langle E(\theta) \rangle$	$\langle t \rangle$ [ms]
RANSAC	5	0.107	4.344	4.266	5.244	10.085	4,492.2
	10	0.253	6.413	6.384	9.030	10.826	9,123.3
	20	0.544	10.264	9.776	12.012	11.786	20,626.7
	50	0.772	14.852	14.572	14.655	10.226	38,161.3
LO-RANSAC	5	0.104	4.194	3.953	5.169	11.709	24,105.5
	10	0.242	6.096	6.219	8.549	12.250	57,995.2
	20	0.532	9.777	9.311	11.620	12.682	125,795.0
	50	0.770	14.034	13.888	13.960	10.167	294,498.0
SA-RANSAC	5	0.105	4.026	3.938	5.115	12.205	20,882.6
	10	0.238	6.088	6.064	7.833	12.539	19,708.4
	20	0.528	9.768	9.163	10.912	12.578	39,827.8
	50	0.769	14.324	14.111	12.535	9.892	83,365.4

Table 5.4: Results of the overlap detection experiment.

number of tracks seems to have little or no effect on the error of the incidence angle parameter.

The measured running times confirm the result of the previous experiment that the fastest algorithm is RANSAC. The effects of the local optimization again appear consistently visible in comparisons of failure rates and errors between RANSAC and the remaining two algorithms. On the other hand, with more particles in the frame the time complexity of both locally optimized algorithms quickly grows to undesirable dimensions.

5.2.3 Adaptive Parameter Conditioning

In order to improve the performance of the presented detection algorithms, a modification is proposed. During testing, evaluated algorithms have been observed to needlessly waste time in unproductive attempts to segment clearly non-overlapping tracks. In addition, LO-RANSAC as well as SA-RANSAC frequently executed computationally intensive local optimizers, presumably in a large quantity of suboptimal samplings.

To address both issues, parameters of segmentation and fitting algorithms have been conditioned on morphological properties of input clusters. Morphological classifiers have been applied to MPX and TPX data in the past, mainly for the purposes of particle species classification and data quality analysis [Hol+08]. In this application, morphological classification serves to provide a discrimination between simple clusters, on which processing time can be saved by making additional assumptions, and complex clusters, which have to be examined closely due to suspected overlaps.

For this, the original classifier proposed by Holy et al. is reused. Its original morphological class labels (depicted with examples in Figure 5.4) are grouped as follows.

Simple Clusters: dots, straight tracks and small blobs

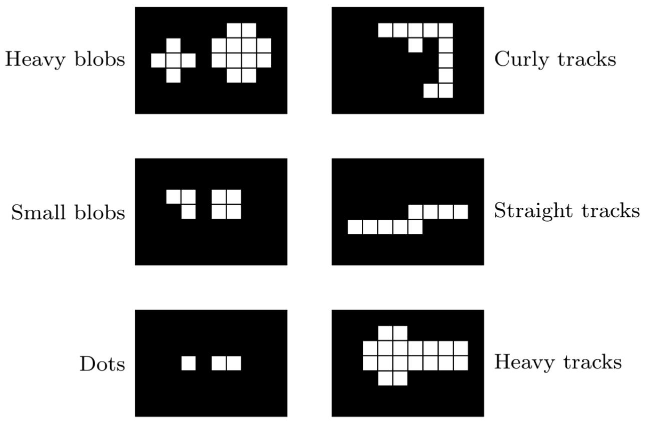


Figure 5.4: Examples of morphological cluster classes defined for pattern recognition in Medipix2 frames [Hol+08].

Complex Clusters: heavy tracks, heavy blobs and curly tracks

While the Flood Fill segmentation algorithm is executed for both classes, Hough Transformation refinement is withheld for simple clusters, assuming that such clusters are not overlaps. In addition, parameters of the trajectory fitting kernel function have been conditioned as shown in Table 5.5.

	Simple Clusters	Complex Clusters
Parameter		
Constant Value e_0	-1	-5
Energy Threshold E_0	1	50
Linear Slope k	1	1

Table 5.5: Energy kernel Φ_{ReLU} parameters conditioned on morphological classification.

With this modification, both experiments have been repeated. The results are shown in Tables 5.6 and 5.8, respectively. In addition, for both experiments relative changes in results were calculated. They are shown in Tables 5.7 and 5.9, respectively.

Algorithm	$\langle FR \rangle$	$\langle E(x_{\text{far},1}) \rangle$	$\langle E(x_{\text{far},2}) \rangle$	$\langle E(\varphi) \rangle$	$\langle E(\theta) \rangle$	$\langle t \rangle$ [ms]
RANSAC	0.037	3.270	3.075	3.051	10.516	677.1
LO-RANSAC	0.036	3.206	2.834	2.889	11.971	4,975.6
SA-RANSAC	0.037	2.638	2.894	2.887	12.046	1,378.0

Table 5.6: Results of the trajectory fitting detection experiment with parameter conditioning.

Algorithm	Relative Change $\langle x_{\text{conditioned}} \rangle$ / $\langle x \rangle$ FR	$E(x_{\text{far},1})$	$E(x_{\text{far},2})$	$E(\varphi)$	$E(\theta)$	t
RANSAC	1.681	1.013	1.090	1.001	1.010	0.669
LO-RANSAC	1.714	1.106	0.986	1.001	1.009	0.898
SA-RANSAC	1.762	0.970	0.995	1.011	0.957	0.782

Table 5.7: Relative change of the trajectory fitting experiment results between the algorithms without and with parameter conditioning.

Algorithm	n	$\langle FR \rangle$	$\langle E(x_{\text{far},1}) \rangle$	$\langle E(x_{\text{far},2}) \rangle$	$\langle E(\varphi) \rangle$	$\langle E(\theta) \rangle$	$\langle t \rangle$ [ms]
RANSAC	5	0.136	4.507	4.213	6.503	10.415	5,202.2
	10	0.273	6.737	6.727	11.127	11.667	11,550.7
	20	0.544	10.755	10.111	17.833	14.034	28,848.4
	50	0.764	15.058	14.867	23.408	14.569	65,918.3
LO-RANSAC	5	0.152	4.211	4.027	6.494	12.202	24,261.1
	10	0.264	6.420	6.438	10.495	12.892	78,843.0
	20	0.534	10.331	9.780	17.240	14.640	156,714.0
	50	0.762	14.353	14.248	22.637	14.482	321,088.0
SA-RANSAC	5	0.133	4.325	3.885	6.376	12.199	11,153.0
	10	0.260	6.379	6.419	10.173	13.182	24,063.9
	20	0.528	10.301	9.752	16.809	14.549	55,635.9
	50	0.761	14.589	14.508	22.054	14.191	105,597.0

Table 5.8: Results of the overlap detection experiment with parameter conditioning.

Algorithm	n	Relative Change $\langle x_{\text{conditioned}} \rangle$ / $\langle x \rangle$ FR	$E(x_{\text{far},1})$	$E(x_{\text{far},2})$	$E(\varphi)$	$E(\theta)$	t
RANSAC	5	1.271	1.037	0.988	1.240	1.033	1.158
	10	1.079	1.051	1.054	1.232	1.292	1.266
	20	1.000	1.048	1.034	1.485	1.191	1.399
	50	0.990	1.014	1.020	1.597	1.425	1.727
LO-RANSAC	5	1.462	1.004	1.019	1.256	1.042	1.007
	10	1.090	1.053	1.035	1.228	1.053	1.360
	20	1.004	1.054	1.050	1.484	1.154	1.246
	50	0.990	1.023	1.026	1.622	1.424	1.090
SA-RANSAC	5	1.267	1.074	0.987	1.247	1.000	0.534
	10	1.092	1.048	1.058	1.299	1.051	1.221
	20	1.000	1.054	1.064	1.540	1.157	1.397
	50	0.990	1.019	1.028	1.759	1.435	1.267

Table 5.9: Relative change of the overlap detection experiment results between the algorithms without and with parameter conditioning.

In the results of the repeated trajectory fitting experiment, failure rate as well as errors of the estimated model parameters have overall risen by several percentage units. This can be presumably attributed to the lack of Hough Transformation refinement in the segmentation stage and changes in the energy kernel function in the fitting stage. On the other hand though, the processing time has significantly decreased for all tested algorithms. In the best case, the parameter conditioning has saved over 30 percent of the original processing time. Since the relative increase of errors is rather marginal in comparison with the speedup obtained, the proposed enhancement is considered to have achieved its goal in this particular experiment.

Unlike its predecessor, the repeated overlap detection experiment has yielded no improvements at all. Conversely, errors as well as processing times have risen significantly in comparison with the unconditioned version of the experiment. This undesirable effect can be explained due to increased occupancy of frames in the synthetic test sets. With increased number of tracks in the frame, the probability of assignment of the simple cluster class decreases. Since performance improvements occur in part due to accelerated processing of simple clusters, the overall processing time is not improved.

In conclusion, the proposed parameter conditioning method has yielded a significant speedup in frames with low number of overlaps. In saturated frames, this approach has rather adverse effect on the processing time as well as on the accuracy of the results. It is, however, important to note that better results may possibly be obtained by conditioning different parameters on a different set of classes, or by using alternative adaptive parameter initialization methods.

5.2.4 Conclusion

The performed experiments have shown that the proposed track segmentation and trajectory fitting algorithms have achieved their design goals. The trajectory fitting experiment has indicated that estimated parameters have satisfactory accuracy in comparison with human-annotated ground truth, possibly hinting at poor three-dimensional resolution for reconstruction of particle trajectories. Of the three tested algorithms, the overall fastest one was conventional RANSAC without any local optimization. The effects of local optimizers, regardless of their type, can be observed in consistently decreased spatial errors of the estimated parameters. However, the improvements yielded by local optimizers do not seem to outweigh the additional increase in running time.

The overlap detection experiment has demonstrated that the presented solution is robust enough to effectively analyze frames of larger occupancy. It should however be noted that errors as well as processing times have increased rapidly with the number of tracks present in the frame. This renders the tested algorithms practically usable only for frames consisting of at most 10 overlapping tracks.

To address the issue of low performance, a solution based on adaptive parameter conditioning was devised. This alternate version of the presented algorithms has achieved over 30 percent speedup in the best case scenario. On the other hand, the modification has been observed to have rather adverse effects on the overall performance in cases of high frame occupancy.

5.3 Recognition Evaluation

To offer an insight into their performance, the presented recognition methods are evaluated separately from the detection methods. For this purpose, the ground truth information previously used for construction of synthetic frames is utilized.

To evaluate classifier properties, three metrics are used:

Accuracy: The recognition accuracy is conventionally defined as the fraction of correct classifier predictions:

$$\text{accuracy}(y, \hat{y}) = \frac{1}{N} \sum_{j=1}^{N} 1[y^j = \hat{y}^j] \tag{5.3}$$

Here, y^j and $\hat{y}^j$, $j \leq N$ denote true and predicted class labels, respectively, and $1[x]$ is the indicator function.

Confusion Matrix: For m classes, the (normalized) confusion matrix is defined as a $[0,1]^{m \times m}$ matrix, where cell at index $(i,j) \in [m]^2$ contains the fraction of samples of class j predicted in class i.

Note that by definition, the columns of the confusion matrix sum up to one. Furthermore, the quality of predictions can be assessed by examining dominance of the main diagonal (diagonal confusion matrix implies unit accuracy).

Rejection Rate: In classification with the rejection option, the rejection rate is given by:

$$\text{RR} = N_{\text{rejected}}/N. \tag{5.4}$$

In the formula, N_{rejected} is the number of rejected samples and N denotes the number of all samples. The rejection rate is thus a relative quantity representing the fraction of rejected samples in all tested samples.

To enhance the accuracy of the calculated metrics, *5-fold cross-validation* is employed. In K-fold cross-validation, the ground truth data set is randomly partitioned into K *folds* – subsets of equal cardinality. Repeatedly, one fold is used as a testing set while the other $K-1$ serve as training sets for the classifier. This procedure is repeated K times, so that all folds become test sets exactly once, yielding K results from which mean values are calculated for every evaluation metric.

During cross-validation, *stratification* is performed to ensure that even though the fold partitioning process is randomized, all sample classes are represented in every fold in similar quantities.

In all tasks related to classification, implemented algorithms utilize the SciKit Learn package for Python [Ped+11].

5.3.1 Classifier Configuration

The aim of the classifier configuration experiment is to test the accuracy and the performance of the k-NN classifier in various configurations. The experiment follows the conventional cross-validation scheme, wherein several pairs of training and testing sets are generated at random. For every pair, multiple classifiers with different configurations are learned and evaluated separately.

In this experiment, 15 classifier configurations are examined. They are obtained as a Cartesian product of 5 selected values of the k parameter and the following model storage layouts:

Naive: In this layout, the classifier stores the entire training set in a contiguous memory array. For prediction, the full array is enumerated in order to identify the nearest neighbors.

k-d Tree: In this layout, the classifier builds a k-d tree during learning [Ben75]. For prediction, the space partitioning memory data structure is utilized in order to constrain the set of samples, in which the nearest neighbors may be located.

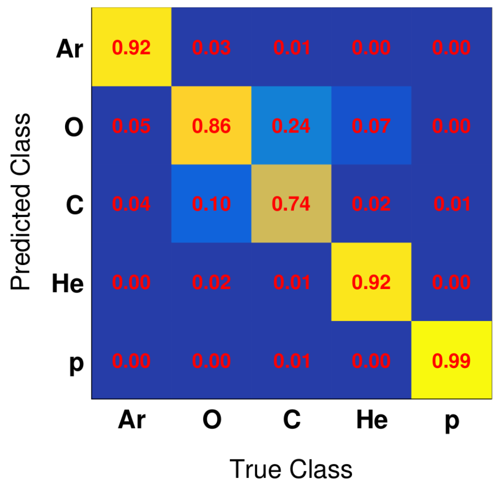
Ball Tree: In this layout, the classifier builds a ball tree during learning [Omo89]. Similarly to k-d trees, ball trees are used during prediction to reduce the number of enumerated samples.

Since the listed layouts use algorithms of different computational complexities for the training and the testing phase, the motivation for the experiment is to identify optimal classifier configurations in terms of prediction accuracy and performance. For that reason, in addition to prediction quality metrics training and testing times of the classifier are tracked. For feature extraction, 32 uniform sampling intervals are used. The cross-validated results of the experiment are shown in Table 5.10 and Figure 5.5.

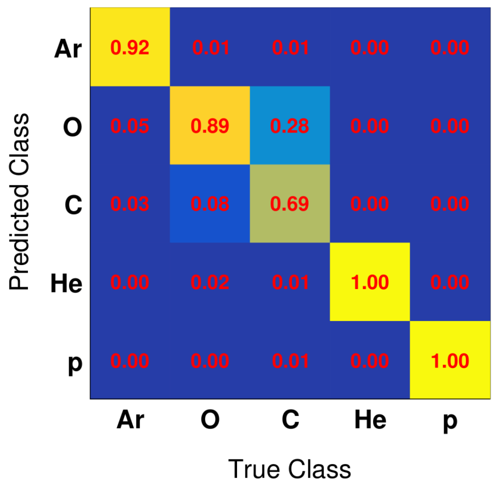
In all experiments, classifiers have exhibited sufficient accuracy in predictions. This is supported by the observation that confusion matrices (shown in Figure 5.5) appear predominantly diagonal. Curiously, the choice of the k parameter seems to only have a limited effect on the behavior of the classifier, achieving the best results for $k \in \{3, 7\}$.

On the other hand, performance indicators seem to follow expectations. In the training phase, the naive model storage layout consistently outperformed both tree-based layouts. This is however reversed in the testing phase, where the naive approach is the slowest of all three. Among layouts, the fastest predictions are produced by ball trees. It is however important to note that this result may be inconclusive due to a very narrow margin between both tree-based layouts.

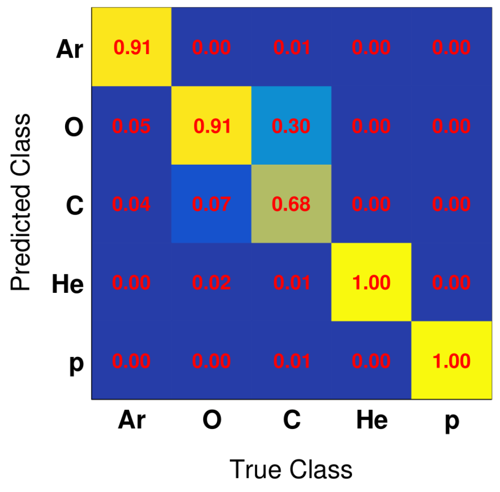
The choice of the k parameter seems to have insignificant effect on the performance of the classifier in the training phase. In the testing phase, the complexity of predictions in tree-based layouts increases for large values of k.



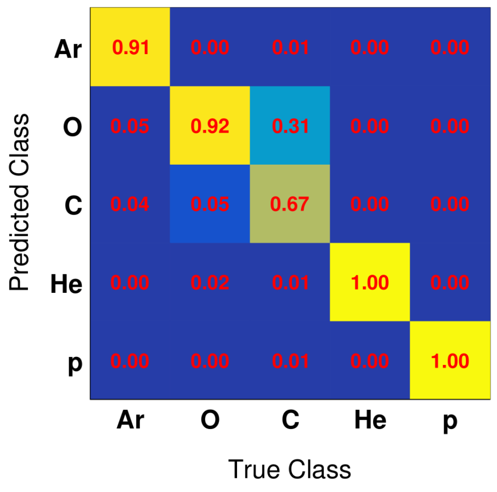
(a) $k = 1$, accuracy $= 0.880$



(b) $k = 3$, accuracy $= 0.892$



(c) $k = 5$, accuracy $= 0.891$



(d) $k = 7$, accuracy $= 0.892$

Figure 5.5: Selected confusion matrices of the k-NN classifier trained for various values of k in the classifier configuration experiment.

Layout	k	$\langle t_{\text{train}} \rangle$ [ms]	$\langle t_{\text{test}} \rangle$ [ms]	$\langle$accuracy$\rangle$	$\langle$RR$\rangle$
Naive	1	0.385	7.334	0.880	0
	3	0.396	7.374	0.892	0
	5	0.368	8.115	0.891	0
	7	0.641	7.562	0.892	0
	15	0.356	7.446	0.863	0
k-d Tree	1	0.979	4.074	0.880	0
	3	0.925	4.212	0.892	0
	5	1.000	4.521	0.891	0
	7	0.982	4.709	0.892	0
	15	0.953	5.013	0.863	0
Ball Tree	1	1.019	3.450	0.880	0
	3	1.032	3.515	0.892	0
	5	1.057	3.757	0.891	0
	7	0.998	3.741	0.892	0
	15	1.003	4.210	0.863	0

Table 5.10: Results of the k determination experiment.

5.3.2 Sampling Intervals

While the previous set of experiments has compared the performance and accuracy of the k-NN classifier in various configurations, the sampling interval experiments focus on the feature extraction process, which occurs prior to classification. In particular, the experiments compare multiple choices of interval configurations, from which features are sampled. While a multitude of configurations can be proposed, for the sake of tractability, all tested interval configurations are uniform and the only varying parameter is the interval count n.

In the experiments, 5 interval configurations are tested for corresponding interval counts $n \in \{1, 16, 32, 64, 128\}$. For these sets of intervals, the best-performing configuration from the previous experiment has been selected. The neighbor count has been fixed at $k = 7$ and the used model storage layout utilizes ball trees. Results have again been aggregated by means of stratified 5-fold cross-validation in order to reduce fluctuations due to randomized partitioning. The results of the experiment are shown in Table 5.11 and Figure 5.6.

Intervals	$\langle t_{\text{train}} \rangle$ [ms]	$\langle t_{\text{test}} \rangle$ [ms]	$\langle$accuracy$\rangle$	$\langle$RR$\rangle$
single value	0.906	1.054	0.851	0
16 uniform	0.645	1.917	0.893	0
32 uniform	1.023	5.315	0.892	0
64 uniform	2.051	6.065	0.885	0
128 uniform	1.901	11.225	0.887	0

Table 5.11: Results of the sampling interval experiments.

In the performance aspect, the results of the experiment are consistent with expectations. The processing times of the training phase as well as the testing phase seem to increase with the number of sampling intervals, and in turn the number of features. Unlike processing times, the effect of interval choice on classification accuracy seems to be much less significant. While the lowest number of intervals produces the least accurate classifications, it generally does not hold that more dense interval configurations achieve better accuracy.

5.3.3 Classification with Rejection Option

As evidenced by the results of the previous experiments, the accuracy of the presented classification method may be further improved. An effective approach to minimize the number of misclassified samples is to permit the classifier to withhold decision in cases where misclassification might likely occur. The act of withholding classifier decision is known as *the rejection option* and is advantageous especially in applications where the effects of misclassification are less desirable than those of no classification at all.

The practical implementation of classification with rejection option is quite straightforward. In addition to making a class decision, the k-NN classifier is extended to calculate *a confidence measure* for the decision. The value of this measure is then used to make a secondary decision for either acceptance or rejection, usually based on a so-called *confidence threshold*. Conventional classification can then be viewed as a special case of classification with rejection where the threshold is set to 0. For thresholds approaching 1, classifiers usually reject more samples, reducing the number of misclassified samples but also providing less information in turn.

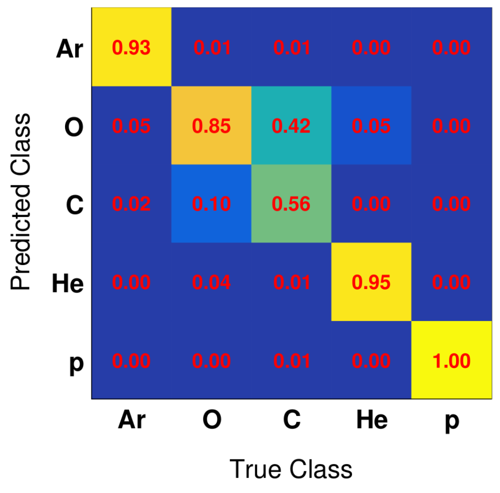
While there exist numerous approaches to calculating viable confidence measures in nearest neighbor classifiers, this work uses the most obvious option based on estimation of the posterior probability $P[y^j \mid x]$ for class y^j. This estimation is given by [CH67]

$$\hat{P}[y^j \mid x] = k^j/k. \tag{5.5}$$

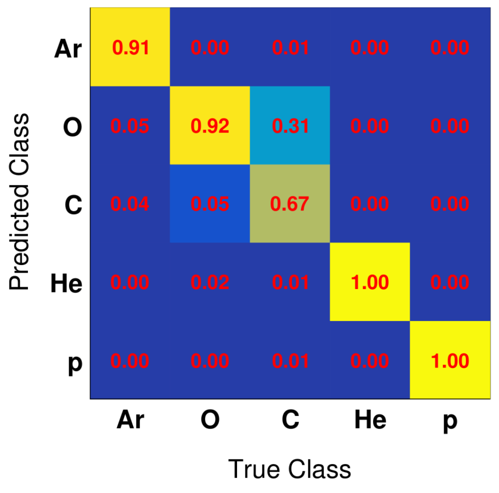
Here, k denotes the number of neighbors in k-NN and k^j denotes the number of training samples from the class y^j among the closest neighbors.

With the confidence measure defined, another experiment has been performed using various confidence thresholds $c \in \{0.5, 0.75, 0.9, 0.95\}$. In the experiment, the best-performing configuration from both previous experiments has been selected ($k = 7$, ball tree storage layout and 16 sampling intervals). The calculation of accuracy has been altered to only consider accepted classifications. The results of the experiment are shown in Table 5.12 and Figure 5.7.

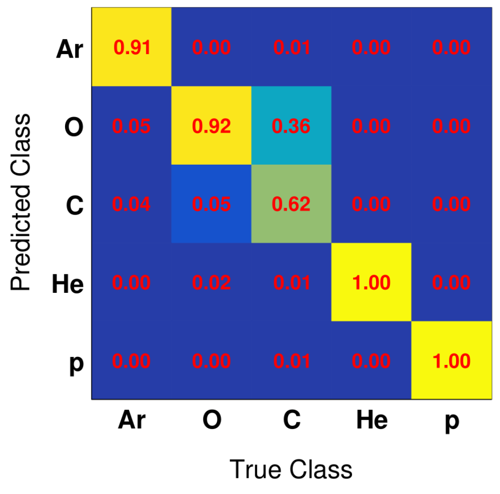
In the presented results, confidence thresholding has achieved quite a significant improvement of classification accuracy. Consistently with expectations, with higher thresholds, accuracy of the classifier as well as its rejection rate seem to increase. Nevertheless, even for the largest tested confidence threshold, rejection rate appears reasonably low for practical use. As expected, confidence thresholding seems to have little or no effect on the processing time at any classification stage.



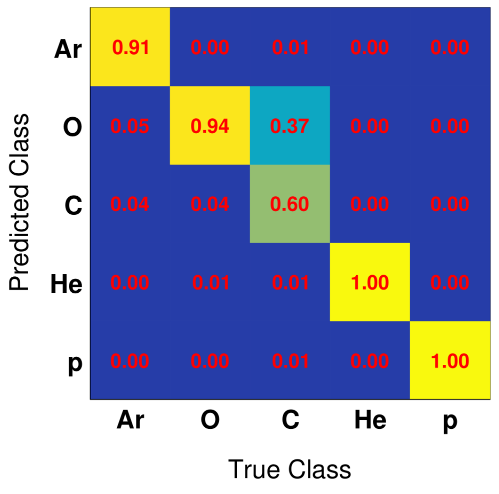
(a) single value, accuracy = 0.851



(b) 32 uniform, accuracy = 0.892

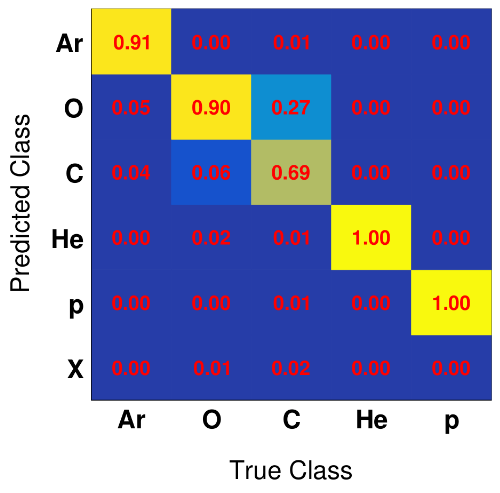


(c) 64 uniform, accuracy = 0.885

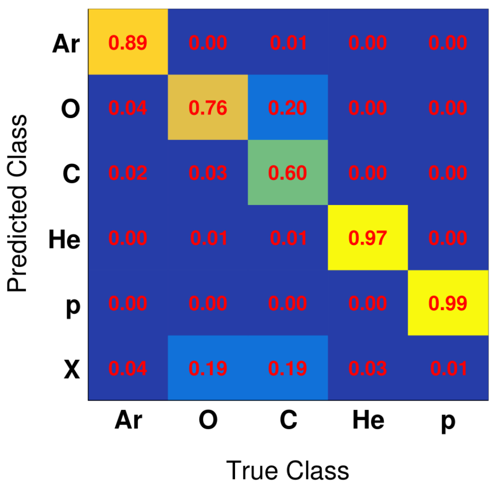


(d) 128 uniform, accuracy = 0.887

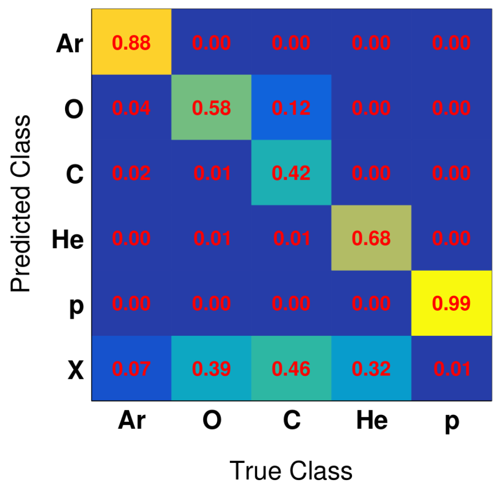
Figure 5.6: Selected confusion matrices of the k-NN classifier trained for various intervals in the sampling interval determination experiment.



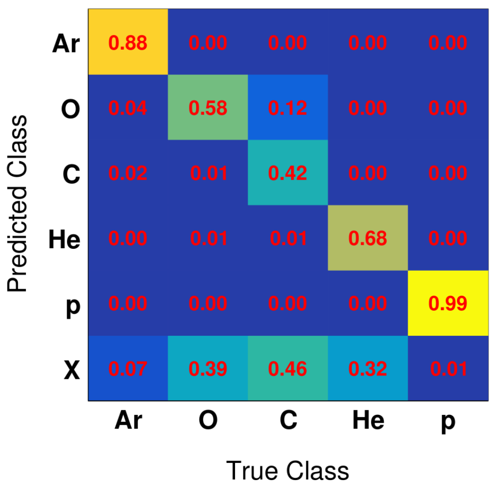
(a) $c = 0.5$, accuracy $= 0.895$, RR $= 0.006$



(b) $c = 0.75$, accuracy $= 0.922$, RR $= 0.111$



(c) $c = 0.9$, accuracy $= 0.935$, RR $= 0.249$



(d) $c = 0.95$, accuracy $= 0.935$, RR $= 0.249$

Figure 5.7: Confusion matrices of the k-NN classifier trained for various confidence thresholds in the rejection option experiment. In the rows of the confusion matrix, X marks samples rejected by the classifier.

Confidence Threshold c	$\langle t_{\text{train}} \rangle$ [ms]	$\langle t_{\text{test}} \rangle$ [ms]	$\langle$accuracy$\rangle$	$\langle$RR$\rangle$
0.50	1.208	2.481	0.895	0.006
0.75	0.731	2.325	0.922	0.111
0.90	0.620	1.612	0.935	0.249
0.95	0.721	2.398	0.935	0.249

Table 5.12: Results of the rejection option experiment.

5.3.4 Conclusion

The performed experiments have shown that the selected feature model is viable for particle species classification under the definition used in the scope of this work. In the experiments, various configurations of the feature extraction method and the k-NN classifier have been evaluated both in terms of accuracy and performance.

Overall, all tested configurations have yielded satisfactory results. Even though observed differences in accuracy were insignificant, the most accurate results were produced by 7-NN classifier using features obtained by sampling $\frac{dE}{dx}$ in 16 uniform intervals. This particular configuration gave the best performance using ball tree model storage layout.

To improve classification accuracy, a confidence measure has been introduced in order to identify samples likely to be misclassified. Testing with various confidence thresholds has shown visible improvements in classification accuracy, while maintaining relatively low rejection rates.

5.4 Analysis of MoEDAL Data

The objective of the last experiment is to show that the presented methods can be applied with success in practical analysis of measurements conducted with TPX detectors. For this purpose, data recently taken at the MoEDAL Experiment are examined.

5.4.1 About MoEDAL

Short for *the Monopole and Exotics Detector at the LHC*, MoEDAL is located at the Large Hadron Collider in Geneva, Switzerland. The experiment is operated by the European Organization for Nuclear Research[13] and its primary objective is to directly search for the Magnetic Monopole and other highly ionizing Stable (or pseudo-stable) Massive Particles (SMPs) at the LHC [Ach+14].

Among others, the experiment utilizes TPX detectors as a part of its radiation monitoring system.[14] For this purpose, 5 detector arrays have been positioned in and around the VELO machine, along the trajectory of the highly energetic LHC beam [Pin+09]. The detectors (shown in Figure 5.8) are configured for continuous data acquisition, which

[13]The organization is also known as *Organisation européenne pour la recherche nucléaire* or under the acronym CERN. `http://cern.ch`

[14]TPX detectors are the only active detectors in the MoEDAL experiment.

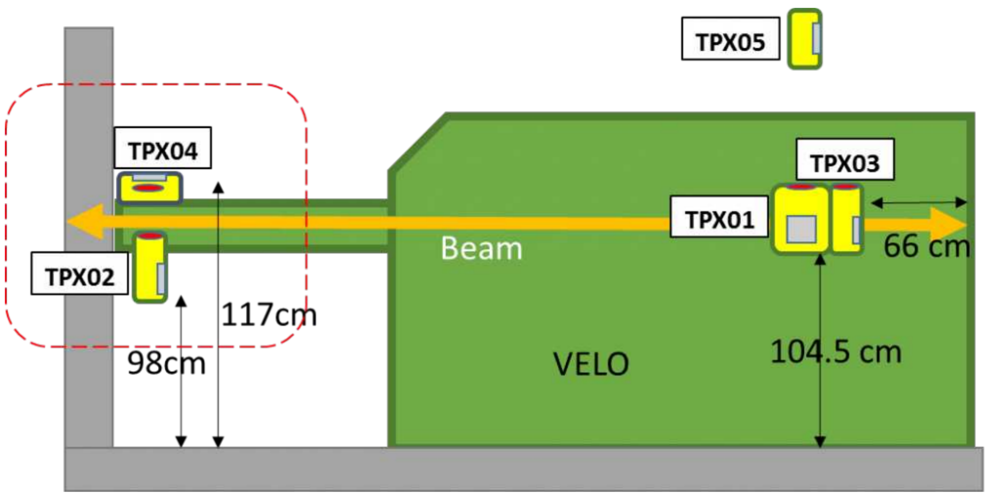


Figure 5.8: Position of the TPX detectors in the MoEDAL experiment [Pin+09].

is ensured by a software system originally developed for controlling a similar detector network at the ATLAS Experiment [Ber+16; Man+17]. Since all detectors operate in the ToT mode, their measurements are compatible with the methods proposed in this work.

5.4.2 Experimental Data & Processing

For the analysis, a series of LHCb runs occuring during a 3-hour time period have been selected. The analyzed frames come from the night of July 29-30, 2017.[15] During this time period, all 5 detectors were online and operational. The values of their configuration parameters at the time of data taking are noted in Table 5.13.

Detector	Sensor Material	Sensor Thickness [μm]	Acq. Time [ms]	Bias Voltage [V]
TPX01	$^{28}_{14}$Si	300	1.00	approx. 100
TPX02	$^{28}_{14}$Si	300	10.00	approx. 100
TPX03	$^{28}_{14}$Si	1,000	5.00	approx. 400
TPX04	$^{28}_{14}$Si	1,000	0.02	approx. 400
TPX05	$^{28}_{14}$Si	1,000	5.00	approx. 400

Table 5.13: Values of hardware and software configuration parameters of TPX detectors at the MoEDAL experiment during the examined runs.

Before processing, frames were filtered to ensure sufficient data quality. To achieve small overlap rates, only frames with occupancy 15% or lower were accepted. To maxim-

[15]In particular, the selected frames were taken during LHCb runs no. 195950, 195952, 195953, 195956 and 195958 (LHC fill no. 6024).

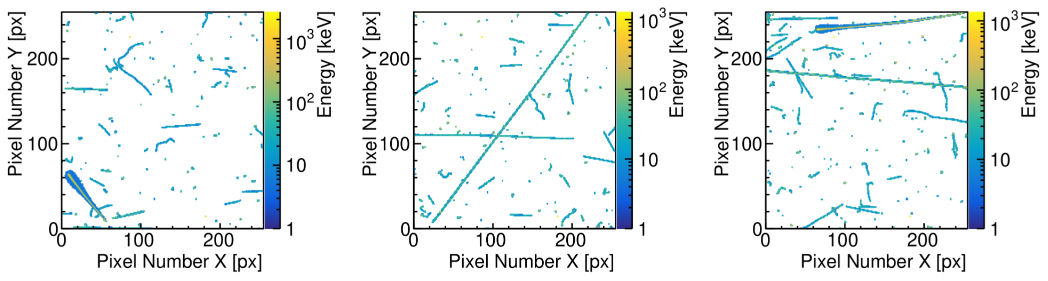


Figure 5.9: Examples of filtered frames used for the analysis. The displayed frames were taken by the detector labeled as TPX03.

ize the angular resolution, clusters labeled with morphological classes other than straight or heavy tracks were excluded from the analysis.

After filtering, TPX frames (shown in Figure 5.9) from each detector were processed by the presented track segmentation and trajectory fitting algorithms. In particular, track segmentation was performed by Flood Fill and refined by the Hough Transformation (for configuration see Table 5.14). Trajectory fitting was performed by a randomized algorithm similar to SA-RANSAC from the previous experiments. Its configuration parameters are listed in Table 5.15.

Segmentation Method	Parameter	Value
Flood Fill	Connectivity Threshold e_0	10^{-3} keV
Flood Fill	Propagation Neighborhood $\mathcal{N}$	8-neighborhood
Hough Transformation	Inlier Distance Threshold	5 pixels
Hough Transformation	Accumulator Threshold $A_{\min}$	5000
Hough Transformation	Maximum Iterations $s_{\max}$	20

Table 5.14: Parameters of the track segmentation algorithms.

Parameter	Value
Proposition Strategy Ψ	informed
Energy Kernel Φ	ReLU
Distance Weight w	Gaussian
Sample Count	1,000
Local Optimizer	sim. annealing
L. O. Iteration Count	1,000
L. O. Window Δ	$\{0, \pm 0.2\}$
Annealing Schedule	linear

Table 5.15: Parameters of the trajectory fitting algorithm.

The executed algorithms have produced hundreds of thousands of detections (their exact counts are listed in Table 5.16), which are subject to the following analysis.

Detector	Trajectory Count
TPX01	932,416
TPX02	587,421
TPX03	447,720
TPX04	869,311
TPX05	630,322

Table 5.16: Number of fitted trajectories per detector.

5.4.3 Angular Parameter Analysis

The purpose of the analysis of angular parameters is to identify prevalent directions among the reconstructed particle trajectories. Since the strongest source of ionizing radiation within the experiment is known to be constituted by the LHC beam, the majority of the reconstructed particle trajectories may be expected to originate at its location. It may therefore be possible to draw comparisons between the observed prevalent particle direction in every detector and the relative direction from the detector to the beam.

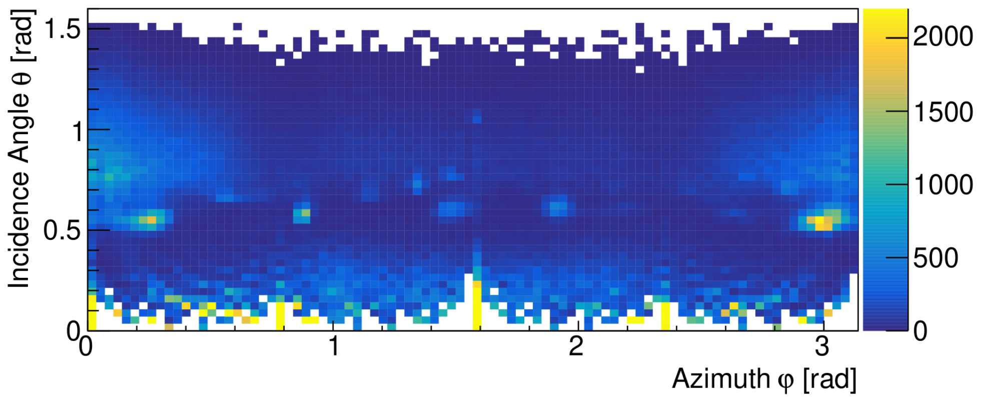
In order to identify the prevalent direction of observed particle trajectories, a simple method was employed. For each detector, fitted trajectories were sorted into bins by the values of two parameters:

1. the azimuth φ,
2. the incidence angle θ

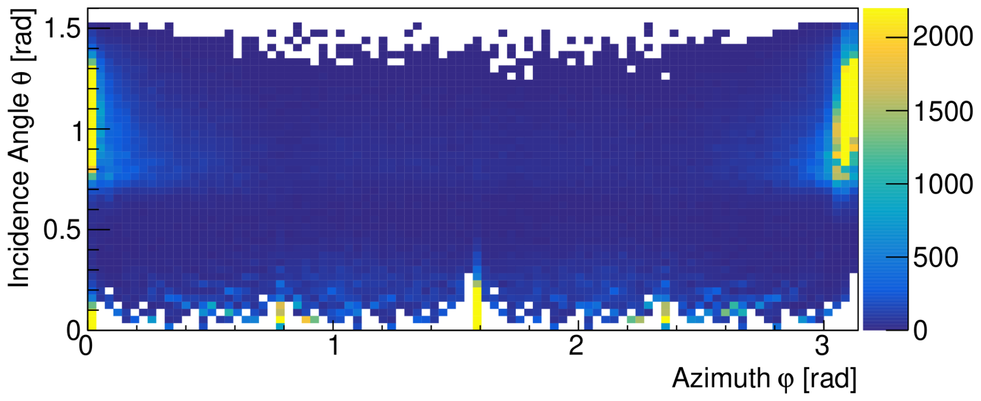
After binning, two-dimensional histograms were plotted in Figure 5.10. Note that due to the cyclical nature of the azimuth, the left and right border of the plots can be viewed as continuously connected (e.g. as if displayed on a cylindrical surface).

For detector TPX01, multiple peaks may be identified at $\theta = \pi/6$ and $\theta = 0$ with various values of azimuth φ. This suggests that observed particle trajectories are predominantly orthogonal, or possibly diagonal, and no particular azimuthal direction is strictly prevalent. Since the detector is positioned to frontally face the LHC beam, this observation is consistent with the expectations.

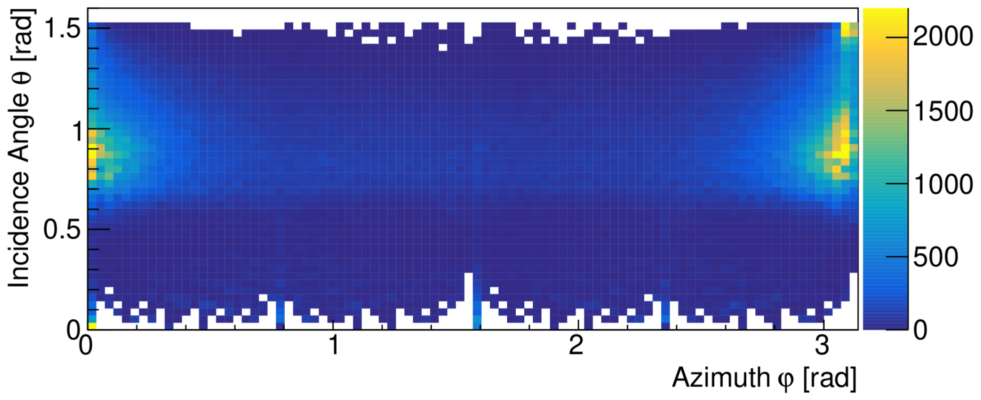
For detector TPX02, a significant peak can be observed at $\varphi = 0$ and $\theta \in [\pi/4, \pi/2]$. This indicates that a group of particles enters the detector diagonally or in near-parallel trajectories. This again confirms the expectation that such particles originate at the LHC beam, which is located to the side of the detector in the direction of azimuth zero. In addition, a pattern of scattered peaks at $\theta = 0$ similar to that observed in detector TPX01 can be identified. These peaks may correspond with particles deviating from the



(a) TPX01

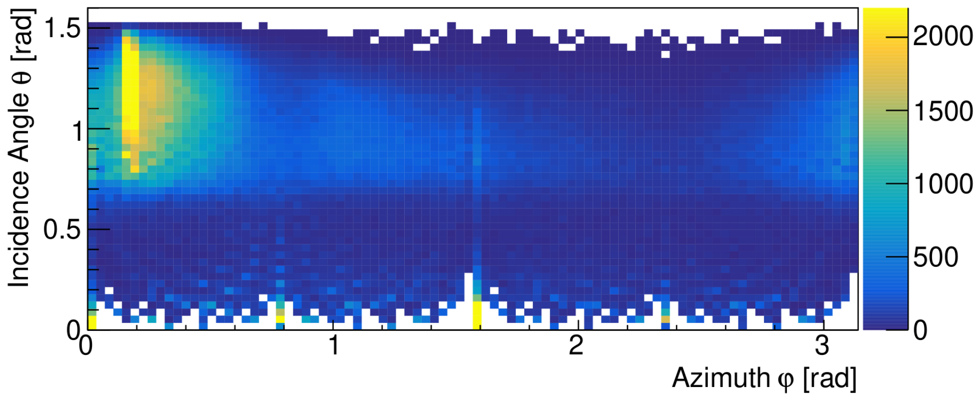


(b) TPX02

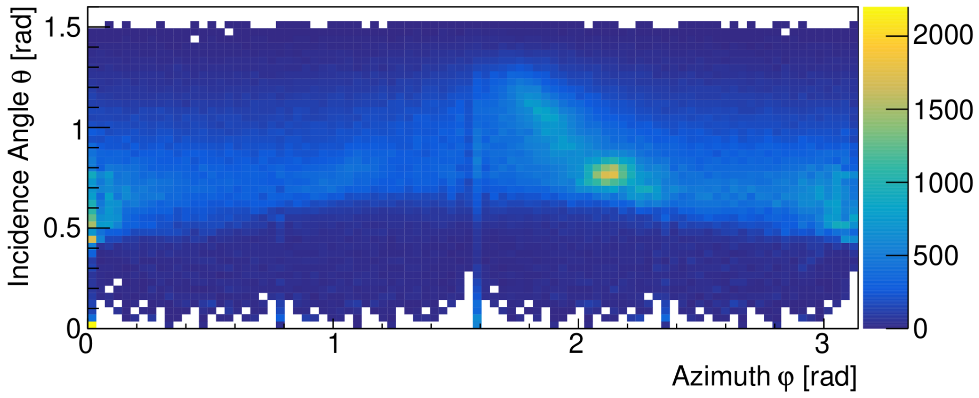


(c) TPX03

Figure 5.10: Histograms of angular parameters of the fitted trajectories grouped by detectors. The horizontal axis shows the azimuth $\varphi \in [0, \pi]$ and the vertical axis shows the incidence angle $\theta \in [0, \pi/2]$. The color axis represents the number of trajectories observed.



(d) TPX04



(e) TPX05

Figure 5.10: (continued) Histograms of angular parameters of the fitted trajectories grouped by detectors. The horizontal axis shows the azimuth $\varphi \in [0, \pi]$ and the vertical axis shows the incidence angle $\theta \in [0, \pi/2]$. The color axis represents the number of trajectories observed.

beam at earlier times (possibly within the VELO machine), consequently entering the detector at strictly orthogonal trajectories.

For detector TPX03, it is possible to discern a significant peak around $\varphi = 0$ and $\theta = \pi/4$. This direction is consistent with expected trajectories of particles emitted by the LHC beam. Since the collimated beam is located at the same height as the detector, its particles enter the TPX sensor layer along trajectories contained in planes parallel to the X-Z plane of the sensor chip, resulting in zero azimuth. Furthermore, since the detector is not directly penetrated by the beam but instead faces the beam from a side, the particles of the beam enter the sensor layer in neither normal nor parallel, but in diagonal direction.

For detector TPX04, a peak can be observed at $\varphi = \pi/6$ and $\theta \in [\pi/3, \pi/2]$. This is also consistent with prior expectations. Note the depiction of the detector in Figure 5.8 may be slightly misleading since the detector is drawn directly above the beam. While the detector assembly is in fact placed at a greater height than the beam, the diagram fails to indicate that the device is shifted approximately 18 cm in the direction of the projection towards the reader. For that reason, the prevalent azimuth was not expected to be exactly zero, but instead in the range $[0, \pi/4]$. In addition, the X-Y plane of the detector is parallel to the beam, leading to the expectation that the particles emitted by the beam will have nearly parallel trajectories.

For detector TPX05, a peak can be seen at $\varphi = 3\pi/4 - \varepsilon$ and $\theta = \pi/4$. This again confirms previous expectations. Similarly to detector TPX04, TPX05 is also not placed directly above the beam but shifted to the side in the direction of projection of Figure 5.8. Due to this, trajectories particles emitted by the LHC beam are expected to have azimuth $\varphi \in [\pi/2, 3\pi/4]$. Secondly, the incidence angle confirms a predominantly diagonal direction.

5.4.4 Energy Loss Analysis

The purpose of the energy loss analysis is to provide information about the radiation fields within the MoEDAL experiment by investigating energies deposited in the TPX detectors. For this, the feature extraction algorithm originally proposed for particle species classification in the earlier experiments has been reused.

Fitted particle trajectories have been uniformly divided into 128 intervals of the same size. In each interval, the energy loss has been sampled from the corresponding TPX frame by the neighborhood-aware feature extraction algorithm (see Expression 4.8). Assuming the Bethe-Bloch case, wherein observed particles were not completely absorbed by the TPX sensor layer material, the mean energy loss $\left\langle \frac{dE}{dx} \right\rangle$ is given by Expression 4.2. The observed values of the mean energy loss have been aggregated separately for each detector and plotted in histograms shown in Figure 5.11.

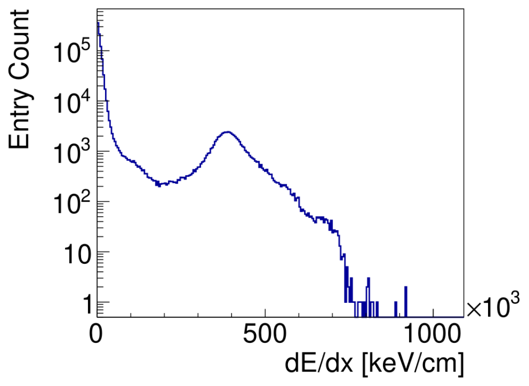
In the histograms, an overall decreasing trend can be observed. This indicates that the majority of the examined particles have transferred low energies to the detectors, whereas high energy transfers have been observed less frequently. Some histograms (e.g. TPX01) show local peaks, suggesting that particles of specific species and momenta have been observed at relatively higher rates. Due to the complexity of the examined field and lack of additional training information, no attempts at explicit inference of particle parameters have been made.

5.4.5 Conclusion

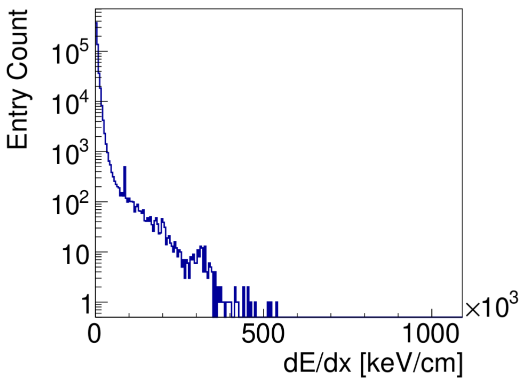
In this section, an analysis of data recently taken at the MoEDAL Experiment has been conducted. The presented track segmentation and trajectory fitting algorithms have been applied in order to determine predominant directions of particle trajectories observed by 5 TPX detector assemblies positioned around the experiment. Expecting that the majority of analyzed particles have been emitted by the LHC beam, the results of the analysis were compared to the known geometry of the experiment. In all 5 instances, the findings seem to support the expectations.

The energy loss analysis has investigated the energies transferred in the detected interactions. The sampling algorithm proposed for feature extraction has been utilized to produce aggregated energy loss histograms of each TPX detector assembly. Even though the findings have not been conclusive in terms of particle species determination, the presented method has identified local peaks, which might suggest relative abundance of particles of specific species and momenta.

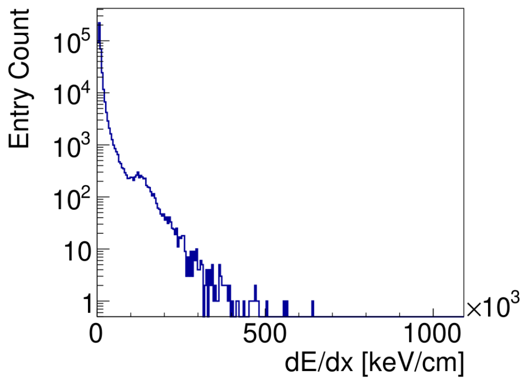
Overall, the performed analysis demonstrates that the methods developed in this work can be applied for practical use in research applications.



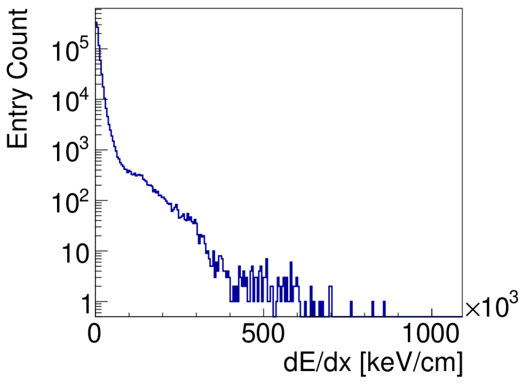
(a) TPX01



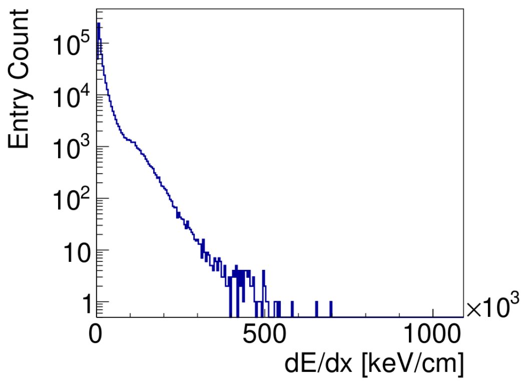
(b) TPX02



(c) TPX03



(d) TPX04



(e) TPX05

Figure 5.11: Histograms of $\left\langle \frac{dE}{dx} \right\rangle$ sampled from detected tracks grouped by the detector.

CHAPTER 6

Conclusion

The goals of this thesis were to update and extend existing methods for robust reconstruction of particle trajectories, and to propose and evaluate a machine learning algorithm for particle species classification. These goals have been successfully accomplished.

The solution to the track segmentation problem has been extended and improved by means of Hough Transformation. The proposed algorithm is capable of effectively handling situations, where a group of pixels is ionized by a multitude of incident particles during the course of a single acquisition period. To the best of author's knowledge, this approach has not been applied on TPX data before. The improvement in segmentation accuracy is particularly desirable in applications, where track counting plays a critical role (e.g. dosimetry or luminosity calculation).

For reconstruction of particle trajectories, a set of novel randomized algorithms has been proposed. Inspired by methods commonly used in estimating analytical parameters from image data, the presented fitting algorithms have been designed to be robust in suboptimal settings, often encountered in practical circumstances. By means of local optimization, the proposed algorithms can achieve subpixel precision through utilization of precisely calibrated energy measurements available in the ToT operation mode.

In order to achieve particle species classification, multiple sampling algorithms were introduced to extract discriminative features from fitted trajectories. The classification task has been formulated and appropriate machine learning algorithm has been selected and implemented. The result is a classifier operating in supervised learning scheme, capable of providing a basic confidence metric for its predictions.

To evaluate their accuracy, all presented algorithms have been tested on real data. In the experiments, track segmentation and trajectory fitting methods have been shown to be overall successful in their goals. In accordance with prior expectations, the experiments have also indicated that the time complexity and the error of the proposed algorithms increases along with the number n of tracks observed in the frame. Nevertheless, the quality of the output has been shown to be adequate up to $n = 20$, presumably surpassing other recent works, which were not designed for robustness in any aspect.

Furthermore, the evaluation of the proposed particle species classifier suggests that the selected feature model and classification algorithm constitute a viable solution to the formulated task.

To demonstrate their usability in research applications, the proposed algorithms have been utilized in a simple analysis of data recently taken by the TPX detectors at the MoEDAL Experiment. The findings have validated the expectation that the majority of analyzed particles likely originate at the LHC beam.

6.1 Future Work

Naturally, the scope of possible future work is quite broad and covers the entire bulk of algorithms presented in this thesis. In general, the time complexities observed in the performed experiments warrant further investigation into their optimization and parallelization. Furthermore, since all proposed algorithms rely on non-intuitive choice of multiple configuration parameters, adaptive approaches may be considered to improve their applicability.

In track segmentation, further refinements can be devised in detection and division of overlaps. The proposed approach is by no means flawless and may provide suboptimal results for collimated or predominantly non-linear tracks. In addition, the quality of subsequent feature extraction may be improved by employing division of energy values from the track intersection area among overlapping tracks.

The results of the performed experiments suggest that all trajectory fitting algorithms are prone to relatively high incidence angle error. Due to manual annotation of the ground truth data, it is not conclusive to determine whether this is a consequence of human bias or a consistent fault of the algorithms. For that reason, further experimentation is required, preferably with ground truth information produced analytically or by means of sufficiently accurate computer simulation.

The feature extraction and classification model can be improved at multiple points. Due to trajectory fitting ambiguity, the current implementation requires a priori information in order to work correctly. A possible enhancement may thus be explored in e.g. adaptive determination of sampling directions. Furthermore, additional relevant information may be introduced into the feature model by efficiently estimating parameters of the Bethe-Bloch a Bragg formulas from the sampled data.

Bibliography

[Ach+14] Bobby Acharya et al. “The physics programme of the MoEDAL experiment at the LHC”. In: *International Journal of Modern Physics A* 29.23 (2014), p. 1430050.

[Ago+03] Sea Agostinelli et al. “GEANT4—a simulation toolkit”. In: *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303.

[Ant+09] Ilka Antcheva et al. “ROOT—A C++ framework for petabyte data storage, statistical analysis and visualization”. In: *Computer Physics Communications* 180.12 (2009), pp. 2499–2512.

[Bar12] David Barber. *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

[Beg16] Jakub Begera. “Calibration and control software for network of particle pixel detectors within the Atlas experiment at the LHC at CERN”. Bachelor’s Thesis. Prague: Faculty of Electrical Engineering, Czech Technical University in Prague, 2016.

[Ben75] Jon Louis Bentley. “Multidimensional binary search trees used for associative searching”. In: *Communications of the ACM* 18.9 (1975), pp. 509–517.

[Ber+16] B Bergmann et al. “ATLAS-TPX: a two-layer pixel detector setup for neutron detection and radiation field characterization”. In: *Journal of Instrumentation* 11.10 (2016), P10002.

[Ber+17] Benedikt Bergmann et al. “3D track reconstruction capability of a silicon hybrid active pixel detector”. In: *The European Physical Journal C* 77.6 (2017), p. 421.

[Bet30] Hans Bethe. “Zur theorie des durchgangs schneller korpuskularstrahlen durch materie”. In: *Annalen der Physik* 397.3 (1930), pp. 325–400.

[Blo33] Felix Bloch. “Zur bremsung rasch bewegter teilchen beim durchgang durch materie”. In: *Annalen der Physik* 408.3 (1933), pp. 285–320.

[Boh13] Niels Bohr. “II. On the theory of the decrease of velocity of moving electrified particles on passing through matter”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 25.145 (1913), pp. 10–31.

[Bor97] Thomas Bortfeld. “An analytical approximation of the Bragg curve for therapeutic proton beams”. In: *Medical physics* 24.12 (1997), pp. 2024–2033.

[CH67] Thomas Cover and Peter Hart. “Nearest neighbor pattern classification”. In: *IEEE transactions on information theory* 13.1 (1967), pp. 21–27.

[CMK03] Ondřej Chum, Jiří Matas and Josef Kittler. “Locally optimized RANSAC”. In: *Joint Pattern Recognition Symposium.* Springer. 2003, pp. 236–243.

[DH72] Richard O Duda and Peter E Hart. “Use of the Hough transformation to detect lines and curves in pictures”. In: *Communications of the ACM* 15.1 (1972), pp. 11–15.

[FB87] Martin A Fischler and Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Readings in computer vision.* Elsevier, 1987, pp. 726–740.

[Frö15] Erik Fröjdh. “Hybrid pixel detectors: Characterization and optimization”. PhD thesis. Mid Sweden University, 2015.

[Gra+11] Carlos Granja et al. “Response of the pixel detector Timepix to heavy ions”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 633 (2011), S198–S202.

[Hec90] Paul S. Heckbert. “Graphics Gems”. In: ed. by Andrew S. Glassner. San Diego, CA, USA: Academic Press Professional, Inc., 1990. Chap. A Seed Fill Algorithm, pp. 275–277. isbn: 0-12-286169-5. url: `http://dl.acm.org/citation.cfm?id=90767.90829`.

[Hoa+12] S Hoang et al. “LET estimation of heavy ion particles based on a timepix-based Si detector”. In: *Journal of Physics: Conference Series.* Vol. 396. 2. IOP Publishing. 2012, p. 022023.

[Hoa+14] S Hoang et al. “Data analysis of tracks of heavy ion particles in timepix detector”. In: *Journal of Physics: Conference Series.* Vol. 523. 1. IOP Publishing. 2014, p. 012026.

[Hoa13] Son M Hoang. “A pattern recognition approach to learning tracks of heavy-ion particles in timepix detectors”. PhD thesis. 2013.

[Hol+08] T Holy et al. “Pattern recognition of tracks induced by individual quanta of ionizing radiation in Medipix2 silicon detector”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 591.1 (2008), pp. 287–290.

[Hou62] Paul VC Hough. *Method and means for recognizing complex patterns.* US Patent 3,069,654. Dec. 1962.

[Jak+08] Jan Jakubek et al. “Pixel detectors for imaging with heavy charged particles”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 591.1 (2008), pp. 155–158.

[Jak11] Jan Jakubek. “Precise energy calibration of pixel detector working in time-over-threshold mode”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 633 (2011), S262–S266.

[KGV83] Scott Kirkpatrick, C Daniel Gelatt and Mario P Vecchi. “Optimization by simulated annealing”. In: *science* 220.4598 (1983), pp. 671–680.

[Kra+11] V Kraus et al. “FITPix—fast interface for Timepix pixel detectors”. In: *Journal of Instrumentation* 6.01 (2011), p. C01079.

[Kuc11] Sujeet Kuchibhotla. “Classification of Sources of Ionizing Radiation in Space Missions: A Machine Learning Approach”. PhD thesis. University of Houston, 2011.

[Lan44] Lev Landau. “On the energy loss of fast particles by ionization”. In: *J. Phys.(USSR)* 8 (1944), pp. 201–205.

[Llo+07] Xavier Llopart et al. “Timepix, a 65k programmable pixel readout chip for arrival time, energy and/or photon counting measurements”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 581.1-2 (2007), pp. 485–494.

[Man+17] Petr Manek et al. “Software System for Data Acquisition and Analysis Operating the ATLAS-TPX Network”. In: *2017 International Conference on Applied Electronics.* Sept. 2017, pp. 103–106.

[Met+53] Nicholas Metropolis et al. “Equation of state calculations by fast computing machines”. In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[OG+14] Keith A Olive, Particle Data Group et al. “Review of particle physics”. In: *Chinese Physics C* 38.9 (2014), p. 090001.

[Omo89] Stephen M Omohundro. *Five balltree construction algorithms.* International Computer Science Institute Berkeley, 1989.

[Ped+11] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Pin+09] James Pinfold et al. *Technical design report of the moedal experiment.* Tech. rep. 2009.

[SP18] Nicholas Stoffle and Lawrence Pinsky. “Identification of stopping ions in a silicon Timepix detector”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 880 (2018), pp. 35–39.

[Sto+12] N Stoffle et al. “Initial results on charge and velocity discrimination for heavy ions using silicon-Timepix detectors”. In: *Journal of Instrumentation* 7.12 (2012), p. C12009.

[Sto+15] Nicholas Stoffle et al. “Timepix-based radiation environment monitor measurements aboard the International Space Station”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 782 (2015), pp. 143–148.

[Tur+11a] D Turecek et al. “Pixelman: a multi-platform data acquisition and processing software package for Medipix2, Timepix and Medipix3 detectors”. In: *Journal of Instrumentation* 6.01 (2011), p. C01046.

[Tur+11b] D Turecek et al. “Small dosimeter based on Timepix device for International Space Station”. In: *Journal of Instrumentation* 6.12 (2011), p. C12037.

[Vil+11] Ricardo Vilalta et al. “Machine learning for identification of sources of ionizing radiation during space missions”. In: *International Joint Conference on Artificial Intelligence, Workshop on AI in Space: Intelligence Beyond Planet Earth.* 2011.

Appendix **A**

Acronyms

TPX	Timepix Detector
MPX	Medipix Detector/Collaboration
ToT	Time over Threshold (TPX operation mode)
ToA	Time of Arrival (TPX operation mode)
RANSAC	Random Sample Consensus [FB87]
MCMC	Markov Chain Monte Carlo
LO	Local Optimization
SA	Simulated Annealing
k**-NN**	k-Nearest Neighbors [Bar12]
ASIC	Application Specific Integrated Circuit
TA	"Towards ASIC" direction
FA	"From ASIC" direction
TP, TN, FP, FN	True/False Positive/Negative (referring to detection)
ROOT	Data Analysis Framework [Ant+09]
LET	Linear Energy Transfer
RMSE	Root Mean Square Error
GUI	Graphical User Interface
MoEDAL	Monopole and Exotics Detector at the LHC
SIMD	Single Instruction, Multiple Data

Appendix B

Contents of Enclosed DVD

```
│
├── README.md ........................ description of the enclosed DVD
├── data/ ................................ examples of input data files
├── outputs/ ....................... results of the performed experiments
├── work/ .................................... thesis implementation
│   ├── src/ ............ source files of the redistributable library component
│   └── examples/ ........... source files demonstrating usage of the library
└── text/ .................................................. thesis text
    ├── src/ ...................... structured thesis text in the LaTeX format
    ├── img/ ................................ figures used in the thesis text
    └── thesis.pdf ..................... compiled thesis text in PDF format
```

Appendix C

Overview of Software Tools

This appendix gives practical information on various software tools developed for the purposes of this work.

Running the Demonstration

This work includes a simple graphical demonstration of the presented detection and recognition methods. This section lists the dependencies of the project as well as other relevant technical details.

Dependencies

The provided software implementation is dependent on the following tools and libraries:

- GNU Make Build System 4 or newer,
- CMake Build System 3.10 or newer,
- GNU C++ Compiler (`g++`) 7.3 or newer,
- ROOT Framework 6.13 or newer with the support for Qt, Python and C++17,
- Python 3.6 or newer with Cython and the SciKit Learn package,
- Qt GUI Framework 5.10 or newer.

Optionally, the project documentation requires Doxygen 1.8 or newer. The parallel implementation for GPU requires NVIDIA CUDA 9.1 or newer.

Integration with the Docker Platform

To ease the compilation of the system, a platform-independent installation may be obtained by use the Docker platform. In particular, this work provides two docker images:

`pm_thesis_fel_env` (defined in `Dockerfile.env`) This image is based on the latest version of the Archlinux image includes all required dependencies listed in the previous section.

`pm_thesis_fel` (defined in `Dockerfile.work`) This image is based on the environment image and contains a working installation of all software tools described in this appendix.

To build and run the images, the reader is referred to the documentation of the Docker platform, available online at `https://docs.docker.com/`. For convenience, a shell script for building both Docker images has been provided in the `dockerize.sh` file.

Demonstrator

The demonstrator application can be found in the `work/examples/demonstrator` directory. Upon execution, it expects a calibrated ASCII file containing TPX frames. This can be provided in two ways:

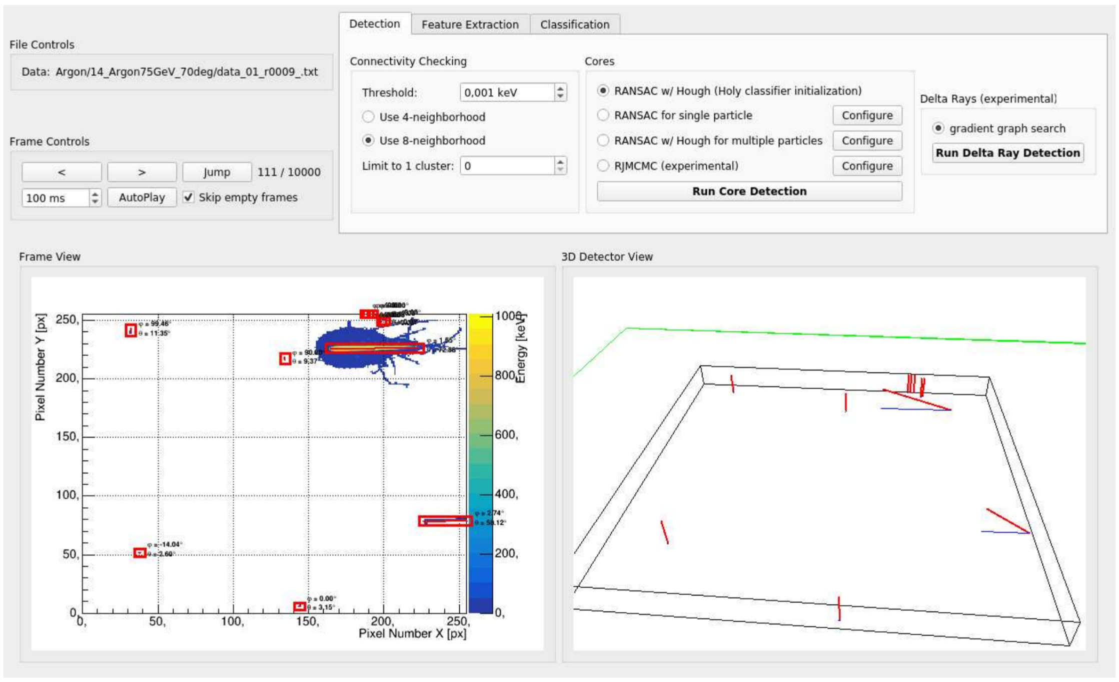
1. When executed without any arguments, the demonstrator first prompts the user with a GUI dialog to select an ASCII data file, then to choose a directory containing files with ToT energy calibration constants. The values of constants a, b, c and t for every pixel are expected to be stored in files located in the calibration directory, named `a.txt`, `b.txt`, `c.txt`, `t.txt`, respectively. In every file, constants are expected to be found in row-ordered sequence of ASCII-formatted numbers separated by white space characters.

2. For convenience, the demonstrator program will forego file system GUI prompts if the ASCII data file path and the calibration directory path are provided in the first two arguments, respectively.

If valid paths are provided, the demonstrator will present the user with a GUI window, sequentially displaying frames from the ASCII data file (shown in Figure C.1) in the order of acquisition. In the top left part of the screen, the user may change the displayed frame by going forward or backward in time, or by choosing to jump to a frame with a specific number in the sequence.

The bottom half of the window is dedicated to interactive visualization of the selected frame and the TPX detector at the time of acquisition. Both interfaces are managed by the ROOT Data Analysis Framework, and can thus respond to familiar plotting commands common to all ROOT user interfaces [Ant+09].

The algorithms implemented in this work can be accessed in the top right part of the window. For clarity, the presented methods have been semantically divided into three tabs.


File Controls

Data: Argon/14_Argon75GeV_70deg/data_01_r0009_.txt

Frame Controls

111 / 10000

100 ms

[✓] Skip empty frames

Connectivity Checking

Threshold: 0,001 keV

( ) Use 4-neighborhood

(●) Use 8-neighborhood

Limit to 1 cluster: 0

Cores

(●) RANSAC w/ Hough (Holy classifier initialization)

( ) RANSAC for single particle

( ) RANSAC w/ Hough for multiple particles

( ) RJMCMC (experimental)

Delta Rays (experimental)

(●) gradient graph search

Figure C.1: The demonstrator program showing fitted trajectories superimposed on a TPX frame in the left part of the user interface. An interactive three-dimensional view of the detector is rendered in the right.

Detection: In this tab, the user can explore, configure and execute various track detection algorithms described in Chapter 3. The interface allows user to modify parameters of the track segmentation method as well as the trajectory fitting algorithms (these can be accessed by pressing the *Configure* button next to the corresponding algorithm).

Upon pressing the *Run Core Detection* button, the selected algorithms are executed on the displayed frame. After completion, the results are superimposed on the frame in red color, showing the intersection of the trajectory with the TPX sensor layer. In the bottom right part of the screen, the trajectory is visualized in a geometrically accurate three-dimensional environment.

Feature Extraction: In this tab, the user can experiment with feature extraction methods described in Section 4.2. Again, the user interface allows to select an algorithm and configure its parameters.

To visualize the effects of parameter choice on the extracted features, the program allows to test the selected feature extraction method on a single instance of the core model produced by the previous stage of the algorithm. For this, the program requires the identification number of the core, which is displayed next to its bounding box in the frame. The feature extraction algorithm is executed by pressing the

Run Sampling button.

Classification: This tab controls the execution of the particle species classification methods, as described in Section 4.3. Since the classifier has to be previously trained on a set of known samples, for convenience, the provided interface allows user to load a trained classifier model from a binary file stored in the *Pickle* format. Such files may be produced by means of the Classifier program, which is described later in this appendix.

Similarly to the previous two tabs, the interface provides means of changing parameters of the classification method. To see effects of the changes, the program can produce labels for the displayed tracks detected by an earlier stage of the algorithm. The classifier is executed by clicking the *Run Classification* button. Results are displayed in the bottom part of the window, next to track bounding boxes. To better visualize different classes, the bounding box colors are set according to the track class.

CAL: A Cluster Analysis Library

All algorithms implemented for the purposes of this work are available in redistributable form of a C++ shared library. The library is named `libcal` and its source codes are available in the `work/include` and `work/src` directories.

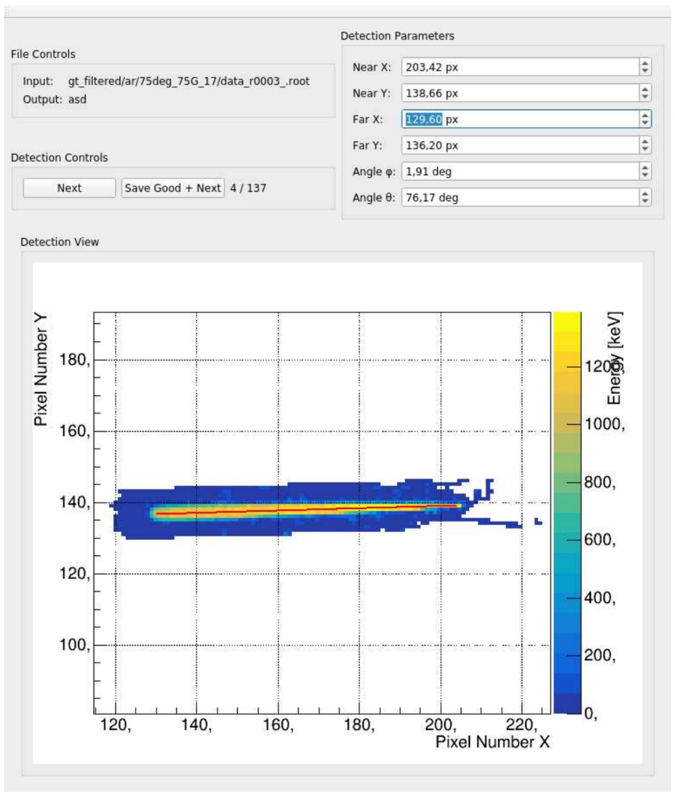
The dependencies and the build system required to compile the library into executable form are identical to those of the demonstrator program. For convenience, the library is also included in the provided Docker image `pm_thesis_fel`.

The programming interface of the library uses object-oriented design to separate implemented algorithms into semantic components (e.g. distance functions, energy kernels, trajectory fitters). By means of compile-time and runtime polymorphism, these can be composed in flexible manner in order to solve problems while keeping computational overhead minimal.

The library is provided along with a documentation package, which can be assembled automatically using the *Doxygen* software tool. An appropriate configuration file is provided in the `work` directory to facilitate the process.

Evaluation Tools

In order to conduct experiments described in Chapter 5, a number of programs were created. Apart from serving their purpose in the experiments, these programs can also be viewed as simple code examples of usage of the presented methods in programmatic environment.


File Controls

Input: gt_filtered/ar/75deg_75G_17/data_r0003_.root
Output: asd

Detection Controls

Detection Parameters

Near X: 203,42 px
Near Y: 138,66 px
Far X: 129,60 px
Far Y: 136,20 px
Angle φ: 1,91 deg
Angle θ: 76,17 deg

Figure C.2: Screenshot of the Track Annotator tool window. In the bottom part of the GUI, the annotated track is displayed along with superimposed model parametrization (drawn in red color). In the top right part, individual model parameters can be inspected and configured. In the top left part, the displayed track can be either accepted or rejected for inclusion in the output file.

Track Annotator

Track Annotator (shown in Figure C.2) is a simple visual tool for producing manual annotations of TPX frames for the purposes of evaluation. It operates on data files stored in the ROOT binary format.

The program source codes can be found in the `work/examples/annotator` directory. Upon start, it expects an input file and an output file path in the arguments. Sequentially, the graphical interface displays all tracks from the input file and lets the user decide whether each track is written in the output file or not. This allows to visually inspect tracks as well as possibly refine model parameters before including them in the output file.

Frame Synthesizer

Frame Synthesizer is a console program capable of producing randomized synthetic frames from a set of annotated tracks stored in the ROOT binary format [Ant+09]. It is located in the `work/examples/synthesizer` directory.

In invocation, the program expects a single output file path followed by a multitude of input file paths. Depending on the configuration specified in the program options, the program generates a number of synthetic frames by means of randomized superimposition. The produced frames as well as the truthful information about track placement are stored in the binary output file.

In addition to other options, the synthesizer is capable of generating visual renderings of each frame stored in sequentially named PDF files. The files following the `frameX_annotated.pdf` naming pattern contain the frame as well as truthful annotations of the track placement. The files following the `frameX_clean.pdf` naming pattern contain the frame without any annotations.

Detection Benchmark

Detection Benchmark is a console program capable of evaluating various track detection methods in different circumstances. As such, it can be used to reproduce results of the detection evaluation experiments.

The program is located in the `work/examples/detection_benchmark` directory and expects a path to a binary input file containing multiple frames with truthful track placement information. Depending on the program options, every frame is processed by a multiple runs of a specific detection algorithm and compared with the truthful information. At the end of evaluation, a report is printed to the standard output containing all tracked metrics.

Batch Detection Utility

Batch Detection Utility is a console program capable of executing track detection and trajectory fitting algorithms on arbitrary sets of data, producing a multitude of detections in a format compatible with the Track Annotator and Classifier Tool.

The program can be found in the `work/examples/detector` directory. Upon execution, it expects a single output file path, followed by at least one (but not necessarily only one) input file path. In addition, a multitude of options specifying the used algorithms and their parameters may be provided.

During its run, the program sequentially reads the input files in the ASCII format and saves obtained detections to the output file in the ROOT format, printing messages about its progress to the standard output. For convenient parallel execution, the program utilizes exactly one CPU core.

Classifier Tool

The Classifier Tool is a Python script capable of learning and executing k-NN classification models on data files containing fitted trajectories. The script can be found in the `work/examples/classifier` directory and may be executed either directly in interpreted mode or as a compiled binary using Cython. While the former approach is easier to use, the latter offers better performance thanks to compile-time optimization.

The script operates on database files – hierarchical structures in the YAML format referencing ROOT files with detections in groups by their labels. Depending on the provided options, the script can produce a learned classification model and save it for later use in the Pickle format,[16], or perform K-fold cross-validation of the provided data set, printing results to the standard output.

High Performance Processing

In practical application, runtime of the presented algorithms is a critical factor. Therefore, a set of tools optimized for high performance is provided to enhance their overall usability.

Trajectory Fitting in CUDA

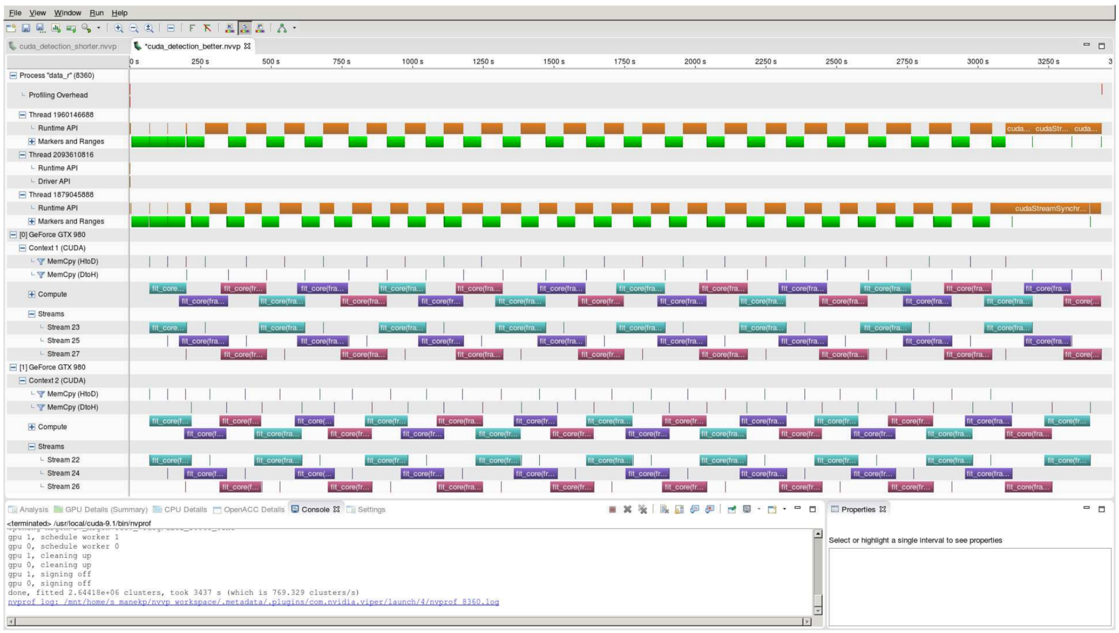
RANSAC, LO-RANSAC and SA-RANSAC algorithms have been implemented as CUDA kernels for parallel execution on NVIDIA graphics cards. The parallelized implementation can be found in the `work/examples/cuda` directory and is disabled by default. To enable compilation and execution, user needs to obtain proprietary NVIDIA hardware[17] and NVIDIA CUDA development libraries of version 9.1 or newer. The compilation can be enabled if the value of the `HAVE_CUDA` option in CMake is set to `ON`.

The program expects at least three arguments: the path to the output file, the calibration directory and the input file. For convenience, the last argument can be followed by an unlimited number of input file paths to process. Unlike other presented tools, all files are expected to be formatted in plain text ASCII format. For more specifications, the reader is referred to the documentation of the program.

Upon execution, the program attempts to identify all NVIDIA devices connected to the host computer. For each device, a host thread and three GPU streams are configured to enable continuous processing according to a well-known parallel pattern. In it, streams perform various tasks depending on the role they are assigned at the moment (shown in Figure C.3). Roles are unique and are periodically exchanged over time in a cycle. This way, each stream is guaranteed to be assigned all roles during the course of a single cycle. The roles are defined as follows:

[16]Saved classification models can be loaded by the Demonstrator or another Python script.

[17]The provided implementation has been tested with two GeForce GTX 980 devices. In general, the program is compatible with any NVIDIA GPU of compute capability 5 or newer. Note that in different GPU models, modification of parameters related to memory allocation may be required or desirable.



```
<terminated> /usr/local/cuda-9.1/bin/nvprof
gpu 1, schedule worker 1
gpu 0, schedule worker 0
gpu 1, cleaning up
gpu 0, cleaning up
gpu 1, signing off
gpu 0, signing off
done, fitted 2.64418e+06 clusters, took 3437 s (which is 769.329 clusters/s)
nvprof log: /mnt/home/s_manekp/nvvp_workspace/.metadata/.plugins/com.nvidia.viper/launch/4/nvprof_8360.log
```

Figure C.3: Screenshot of NVIDIA Visual Profiler depicting the operation of the parallel CUDA implementation with two GeForce GTX 980 devices. While the horizontal axis corresponds with time, the vertical axis shows the utilization of various processing units. Green and orange blocks indicate the utilization of the host CPU, turquoise, purple and dark pink blocks indicate the utilization of the GPU in respective streams. In correspondence with the provided description, the role-exchanging pattern between streams can be observed. In this particular run, over 700 tracks were processed per second.

Loader: In this role, the stream utilizes host CPU to read one of the input files into host memory. After loading a sufficient number of frames, the stream executes segmentation methods to obtain independent pixel sets and copies frame and partitioning information to GPU memory.

Worker: In this role, the stream schedules CUDA kernels on the GPU, invoking parallelized execution of trajectory fitting methods.

Writer: In this role, the stream retrieves the results of the processing from the GPU memory and prints them to the output file.

The output file contains fitted model parameters for every segmented track.

Track Segmentation on Intel® MIC

Flood Fill morphological connectivity checking algorithm for track segmentation has been implemented for execution on devices compatible with the Intel® Many Integrated Core Architecture. The implementation can be found in the `work/examples/separator` directory and is intentionally not integrated into the project build system. To utilize the provided program, user needs to obtain proprietary Intel hardware[18] and compiler. Instructions on building and operating the software are included in the program directory.

The program expects a list of file system paths to ASCII files containing TPX frames. The list can be provided in two ways: either in the standard input or in an additional file passed as the only argument. Upon execution, the program sequentially reads a specified number of frames from the input files and attempts to identify clusters in every frame.

Depending on the set options, computations of the program may be offloaded to compatible MIC devices connected to the host system. If enabled, the system architecture is first explored and all devices available for offloading are identified. Based on this, the appropriate amount of frames is retrieved from the input files. The application uses explicit offload scheme, wherein data are first transferred from the host memory space to the space of the coprocessor, the computation is executed on the coprocessor and upon its termination, the results are transferred back to the host memory.

Regardless of offloading, the provided implementation of Flood Fill uses data parallelism to process multiple TPX frames at the same time. The set of input frames is divided into subsets of similar size and each subset is assigned to a thread running on a single computational core. Within the thread, the algorithm sequentially processes all TPX frames from the given subset.

In addition to data parallelism, SIMD (single instruction, multiple data) instructions are used in frequently executed computational cycles. This allows so-called unrolling of the cycles, enabling the utilization of wide vector registers of the processor's ALU. Consequently, every iteration of the unrolled cycles is then equivalent to multiple iterations before unrolling. The speedup factor depends on the width of the available vector registers and the available instruction sets of the processor architecture.

[18]The provided implementation has been tested with Xeon Phi 5120 and 7120 coprocessors.