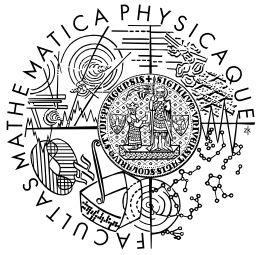


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Petr Mánek

A system for 3D localization of gamma sources using Timepix3-based Compton cameras

Department of Software Engineering

Supervisor of the master thesis: RNDr. Filip Zavoral, Ph.D.
Study programme: Software Systems
Study branch: System Programming

Prague 2018

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

I would like to express my deep sense of gratitude to the staff of the Institute of Experimental and Applied Physics, CTU. In particular, I would like to thank Dr. Martin Pichotka for proposing this project and Ing. Petr Burian, Ph.D. for providing measuring equipment and technical documentation necessary for the completion of this work.

I would also like to thank my supervisor RNDr. Filip Zavoral, Ph.D. for his valuable insight. Lastly, I would like to appreciate the support of my friends and family, for without them, this work would not have been possible.

Title: A system for 3D localization of gamma sources using Timepix3-based Compton cameras

Author: Petr Mánek

Department: Department of Software Engineering

Supervisor: RNDr. Filip Zavoral, Ph.D., Department of Software Engineering

Abstract: Compton cameras localize γ-ray sources in 3D space by observing evidence of Compton scattering with detectors sensitive to ionizing radiation. This thesis proposes a software system for operating a novel Compton camera device comprised of Timepix3 detectors and Katherine readouts.

To communicate with readouts using UDP-based protocol, a dedicated hardware library was developed. The presented software can successfully control the acquisition of multiple Timepix3 detectors and simultaneously process their measurements in a real-time setting. To recognize instances of Compton scattering among observed interactions, a chain of algorithms is applied with explicit consideration for a possibly high volume of measured information. Unlike alternate approaches, the presented work uses a recently published charge drift time model to improve its spatial resolution. In order to achieve localization of γ-ray sources, the software performs conical back projection into a discretized cuboid volume.

Results of randomized evaluation with simulated data indicate that the presented implementation is correct and constitutes a viable method of γ-ray source localization in 3D space. Experimental verification with a prototype model is in progress.

Keywords: Medipix, Timepix3, Katherine readout, SPECT, Compton camera, Back projection, 3D localization

Contents

1. Introduction

Active pixel detectors measure interactions with ionizing particles. Similarly to CCD chips found in commercial photographic cameras, their operation relies on a thin[1] layer of solid semiconductive material, which is evenly partitioned into orthogonal grid of discrete pixels. Each pixel of the grid is physically attached to a circuit, which measures its interactions. Ionizing particles interact with the semiconductive material, inducing currents in pixel circuits.

1.1 Timepix3 Detector

Timepix3 [1] is an active pixel detector, conceived within the Medipix collaboration at CERN, replacing the older Timepix design. The detector has 256×256 pixels with a 55 μm pitch. Each pixel has its own CMOS-based readout circuit with multiple integral counter registers, which can be operated independently of other pixels.

Much like CCD chips of digital photographic cameras, Timepix3 detectors are operated by opening and closing a shutter – a binary signal controlling the state of the detector. Opening the shutter initiates a data acquisition period, during which the detector pixels are responsive to interactions with incident charged particles. However, opposite to standard CCDs, Timepix3 provides a mode for so-called *event-based readout*. Here, upon observed interaction, the pixel is immediately blocked and read out, providing a data stream of observed events.

The information recorded by pixels of a Timepix3 detector mainly depends on *the operation mode* of the detector. The choice of this parameter usually depends on the application and determines the interpretation of values stored in integral counter registers. It has been shown that in the *Time-over-Threshold* operation mode, information recorded by pixels corresponds with the amount of energy deposited in the detector by incident particles [2].

1.2 SPECT & γ-ray Imaging

Single-photon emission computed tomography is a method of functional nuclear imaging, which relies on the measurement of γ-rays. The method requires a so-called γ-*camera* – a scanning device which is sensitive to γ-rays. After a sufficiently long scanning period, information measured by the γ-camera is combined by a reconstruction algorithm in order to obtain a three-dimensional image representation of the scanned environment in the γ spectrum. Since the reconstructed image reveals the location of the γ-ray emission source (or sources), SPECT may be also viewed as a three-dimensional localization method.

In medical applications, SPECT is considered to be an important tool in differential diagnosis of neurological, psychiatric and oncological diseases [3]. SPECT acquisition protocols usually require an injection of γ-emitting radioisotope into the bloodstream of the patient prior to scanning. The radioisotope is diffused

[1] CCD sensor chips are usually substantially thinner than chips of Timepix3 detectors.

and carried along with blood vessels, resulting in an image depicting areas of the scanned tissue, which are accessible to the blood flow.

SPECT also has wide applications beyond the field of medicine. In nuclear energy, the imaging technique is used to determine and verify radioisotope distributions in nuclear fuels [4]. SPECT γ-cameras are utilized for localization of radioactive sources during decommissioning operations or in homeland security applications [5, 6]. Furthermore, SPECT systems have also found applications in chemical engineering, particularly for monitoring of chemical processes [7].

1.3 Thesis Outline

The aim of this thesis is to develop a viable system for data acquisition control, processing and three-dimensional γ-ray source localization based on Timepix3 detectors. The goal of the work is to utilize modern technology in development of a novel γ-camera, which may be integrated with existing SPECT imaging solutions.

The rest of the text is divided as follows:

- Chapter 2 describes the architecture and detection principles used by Compton cameras. It also includes relevant information on the hardware composition and operation of Timepix3 detectors and Katherine readouts.
- Chapter 3 presents an implementation of a novel C library package, capable of controlling Timepix3 detectors by means of Katherine readouts. The text includes architecture design notes and simple usage examples.
- Chapter 4 describes a proprietary hardware and software setup for detection of Compton interactions. It also derives a series of algorithms, sequentially applied during data processing.
- Chapter 5 gives details on the process of volume reconstruction, which combines detected interactions in order to localize stationary point source in three-dimensional space.
- Chapter 6 describes various experiments designed to demonstrate the behavior of selected parts of the system. It also shows experimental results, and discusses their implications.
- Chapter 7 includes discussion of presented work and proposals for various applications. It also emphasizes several possibilities for research in the future.

2. Background

The usage of Compton cameras for γ-ray source detection has been investigated in previous works. The purpose of this chapter is to provide a concise introduction into the physics of Compton scattering, the detection principle based on this interaction and the devices utilized for the implementation.

2.1 Compton Camera

Compton camera [8, 9] is a detection device capable of determining the location of γ-ray sources based on Compton scattering. While Compton cameras are usually configured in stationary setups, their output can be viewed as a three-dimensional image in the γ-ray spectrum.

2.1.1 Related Work

Compton cameras (and more generally γ-cameras) have been known and perfected for several decades. Naturally, this presents a wide range of other related works in published literature, which may be viable for drawing comparisons with the presented system.

The first known γ-camera was proposed by Anger [10] in 1957 and relied on the scintillation property of inorganic crystals rather than Compton scattering. The device referred to as *Anger camera* has eventually become a significant imaging tool with its main application in nuclear medicine. Over 60 years, its angular and energy resolution have been subject to ongoing study and improvement.

The use of Compton scattering for γ-ray source localization was proposed by Todd *et al.* [8] and Everett *et al.* [9] over a decade after the introduction of Anger camera. Since then, various architectures of Compton cameras have been investigated by Singh and Doria [11], Singh and Brechner [12], Gormley *et al.* [13], LeBlanc *et al.* [14] and Sauve *et al.* [15]. In comparison with Anger cameras, Compton cameras are considered to be more sensitive. The properties of both devices have been extensively studied by Gormley *et al.* [16] and Han *et al.* [17].

As previously mentioned, the purpose of this work is to create a novel γ-camera device. Compared with the referenced camera devices, the presented system differs in two key aspects:

1. The system uses only Timepix3 semiconductor detectors as its sensing components.

2. The system estimates the point of interaction with γ-rays in three dimensions rather than only two, effectively reconstructing the depth of the interaction point within Timepix3 detector. This is possible due to charge drift time calibration [18] and allows to considerably suppress the angular error.

At the time of writing, no other γ-camera complying with these characteristics was known to the author. The presented system may thus be viewed as proposed enhancement of a general Compton camera.

2.1.2 Motivation

Since Compton camera is not the only known implementation of a γ-camera, it is appropriate to provide a justification and motivation for its choice as the imaging device used in this work.

Opposite to other alternatives, Compton cameras do not necessitate the presence of mechanical collimators between the scanned object and the sensitive components. Since collimators significantly contribute to the occlusion of the object, their removal creates a larger field of view, allowing for faster and more efficient scanning procedure.

In addition, since collimators are usually manufactured from heavy γ-opaque materials such as lead or tungsten, Compton cameras have overall smaller and more lightweight construction. This makes their design ideal for portable, handheld, or even drone-mounted devices. Thanks to their smaller size, medical Compton cameras may access remote or possibly otherwise unreachable tissues, improving the quality of diagnostic information obtained during radiological procedures.

Compared to collimated cameras, Compton cameras are inherently capable of capturing three-dimensional information. This eliminates any need for specialized gantries or complex rotational systems. Theoretically, the camera resolution is limited only by the angular error and parallax, implying that the usage of multiple viewpoints brings better in-depth resolution in comparison to single devices.

Lastly, Compton cameras are known to have a small fingerprint. In practice, this means that a relatively low number of observations is required in order to fully localize the emission source in a three-dimensional space.

2.1.3 Compton Scattering

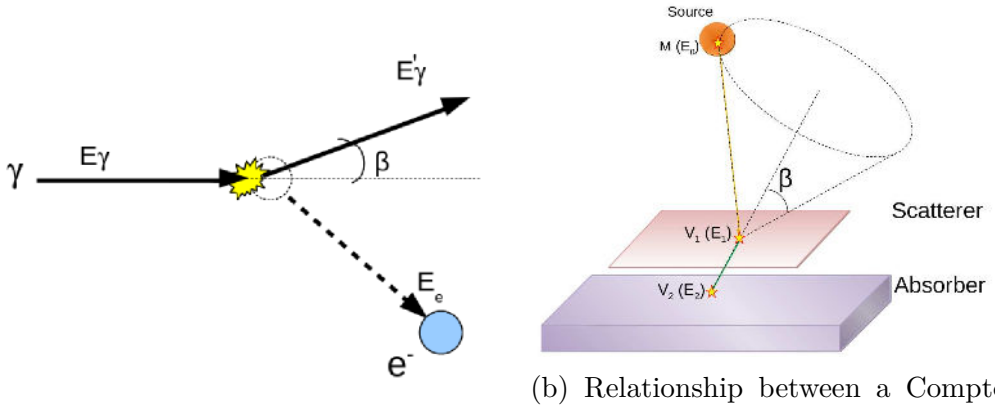
When a γ photon traverses a solid medium, one of the following three cases occurs:

1. The photon is scattered, depositing a fraction of its energy in the medium.
2. The photon is absorbed, depositing its entire energy in the medium.
3. The photon does not interact with the medium at all.

The first two of these interactions are referred to as Compton scattering and photoelectric absorption, respectively. Depending on the medium and the initial energy of the photon, some interactions may be more probable than others. For instance in silicon, Compton scattering is dominant at energies ranging from around tens of keV to tens of MeV [19].

During Compton scattering (illustrated in Figure 2.1a), an incident γ photon of initial energy E_γ interacts with an electron, causing it to recoil. Energy E_e is transferred from the photon to the electron, and the photon is scattered through the Compton angle β with a remaining energy E'_γ. The angle β is related to the initial energy E_γ and final energy E'_γ of the photon by the Compton equation: [20, 21]

$$\cos\beta = 1 - m_e c^2 \left(\frac{1}{E'_\gamma} - \frac{1}{E_\gamma} \right) \tag{2.1}$$



(a) Compton scattering of a γ photon.

(b) Relationship between a Compton cone and emission source M.

Figure 2.1: Compton scattering and its use in source localization [19].

Here, m_e is the electron mass at rest and c is the speed of light. Note that this description assumes that the scattered electron was completely at rest prior to the interaction, which is often not the case in reality. This is the cause of the so-called *Doppler broadening*, which constrains the accuracy when measuring the angle β in practical applications.

2.1.4 Architecture

As mentioned earlier, Compton camera was originally introduced by Todd *et al.*in 1974 [8, 9]. Its design relies on facilitation and measurement of Compton scattering. In general, the device is comprised of two types of components:

Scatterer This component, usually a semiconductor, facilitates Compton scattering with incident γ photons.

Absorber This component, a semiconductor or a scintillation detector, facilitates photoelectric absorption of scattered electrons.

In order to maximize the probability of Compton scattering, some camera designs utilize multiple scatterers arranged in a telescopic configuration. In this work, both the scatterer and absorber components are Timepix3 detectors.

2.1.5 Stationary Source Detection

The detection principle used in Compton cameras expects two observed points V_1 and V_2, one occurring in a scatterer and the other in the absorber component. Along with their locations in space, each point is associated with a corresponding measured energy E_1 and E_2. The aim is to localize a stationary γ-ray source at some unknown location M (depicted in Figure 2.1b).

Assuming that a γ photon was emitted at M in a direction towards the camera, underwent Compton scattering at V_1 and was subsequently absorbed at V_2, the vector $\overrightarrow{V_1V_2}$ represents the direction of the scattered particle. Under this assumption, the energy E_1 can be viewed as energy E_e transferred from the photon to the electron. Similarly, the energy E_2 is the final energy E'_γ of the photon after

the interaction. Assuming that the photon energy loss between scattering and absorption is negligible, its energy before the interaction is given by $E_\gamma = E_e + E'_\gamma$.

Knowing both energies prior to and after the interaction, the magnitude of the scattering angle β may be resolved from the Compton equation 2.1. Under ideal conditions, the source of emission M lies at angle β from vector $-\overrightarrow{V_1V_2}$ placed at V_1. In other words, M is located on the mantle of a cone with apex V_1, half-angle β and axis $\overrightarrow{V_2V_1}$. This gives definition of the so-called *Compton cone*, denoted as $\mathcal{C}(V_1, V_2, \beta)$.

2.1.6 Computed 3D Reconstruction

Observing a single Compton interaction with a camera yields a Compton cone – an infinite quadric surface in three-dimensional space. By the derivation of the method, the unknown emission source M is located somewhere on this cone. With no additional information, this is still insufficient in order to localize M exactly. It may, however, be used to constrain its domain and remove impossible solution candidates.

In bulk, cones produced by interactions corresponding to a single source may limit the solution space enough to localize M within the requested margin of error. This motivates various analytical and iterative approaches.

As their name suggests, analytical algorithms attempt to find an exact analytical solution (or operator) based on the observations. While this solution is exact in the continuous model, for practical application it is usually discretized. The assumes that the camera can be modeled analytically and that the solution can be derived in tractable manner, which may not always be the case.

Unlike analytical algorithms, iterative algorithms operate in a discrete solution space. Their output is produced by repeated evaluations, usually increasing the precision of the localized source by every cycle. Their disadvantage lies in obvious loss of information due to discretization and possibly large number of iterations required in order to achieve sufficient solution quality. On the other hand, iterative approaches allow to incorporate prior knowledge by probabilistic means, thereby allowing to remove intractable noise and artifacts in subsequent volumetric reconstruction.

2.2 Timepix3 Detector

Timepix3 detectors [1] are active pixel detectors comprised of a square matrix of 256×256 pixels with 55 μm pitch. While all pixels in a single assembly are usually operated in orchestrated manner, each pixel may be viewed as a separate detector, capable of performing individual measurements.

2.2.1 Motivation

Since there exist multiple alternative detector designs other than Timepix3, it is important to explain why specifically has Timepix3 been selected for this work.

Thanks to its high time resolution, Timepix3 allows accurate recognition of coincident events across multiple synchronized detectors. In comparison with other detector models, this enables its use in environments with considerably

increased particle flux, where frame-based detectors might observe ambiguous patterns such as cluster overlaps.

Furthermore, the precise time measurements allow to reconstruct interaction depth within the detector by means of drift time calibration [18]. This effectively eliminates angular error, which is otherwise inherently present in the outputs of other detector models.

Since Timepix3 supports data-driven readout mode, the measured information may be processed immediately after its occurrence rather than the end of the measurement period. In comparison with frame-based detectors, this removes the trade-off between efficiency and the response time of the device.

Finally, since Timepix3 offers better energy resolution than previous generations, Compton cameras based on Timepix3 consequently have the potential to achieve higher angular accuracy.

2.2.2 Detector Design

Timepix3 uses so-called *hybrid* design, wherein the detector is physically divided into two components:

Detection Medium A thin[1] layer of solid semiconductive material. Frequently used media include silicon, gallium arsenide or cadmium telluride.

Readout Electronics An ASIC[2] consisting of configurable CMOS-based circuits corresponding to individual pixels of the detector.

Due to the clear division between the two components, a single readout circuit design may be used in combination with various semiconductive materials, giving the detector different properties depending on the aims of particular applications.

2.2.3 Working Principles

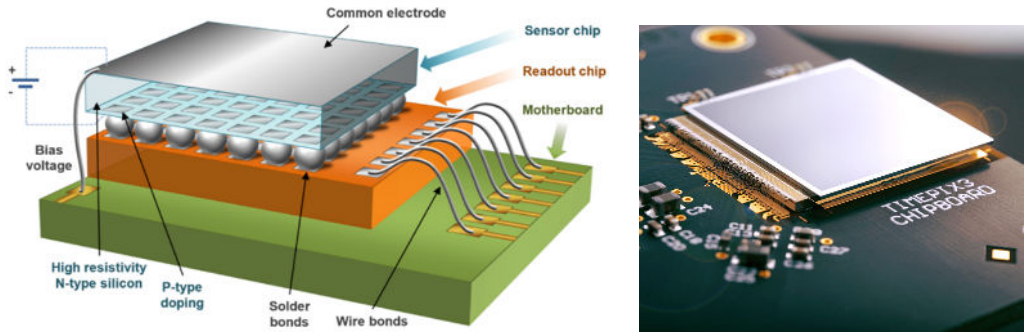
To observe ionizing radiation with Timepix3, potential difference is artificially generated within the detection medium. For that purpose, the material is fitted with a multitude of contacts. On one side, *back-side* electrode is common to all pixels, whereas on the other side *pixel site* electrodes are separately bump-bonded to individual pixel circuits within the ASIC. The potential between the electrodes is called *bias voltage* and is a configurable parameter of the detector. This setup is shown in Figure 2.2.

When an ionizing particle passes through the detector, fraction of its energy is deposited in the detection medium due to single elastic collision with electrons of the semiconductive material. This produces free charge carriers, which drift through the medium due to the applied potential difference. Eventually, charge carriers are collected at pixel site electrodes, generating currents which contribute to analog voltage pulse. The components of the ASIC are designed to efficiently measure, amplify and analyze this signal.

Rather than storing samples of the observed voltage pulse, pixel circuits keep three integral counter registers with 14-bit, 10-bit and 4-bit depth. Depending

[1] Sensor layer thickness usually ranges from 100 μm to 1 mm.

[2] Application Specific Integrated Circuit



(a) Schema with annotated components. (b) Photo of Timepix3 [22].

Figure 2.2: Timepix3 hybrid detector assembly.

on the *operation mode* of the detector, the information from the voltage pulse is encoded into the contents of these three registers.

2.2.4 Operation Modes

The analysis of the voltage pulse is performed using analog discriminator, which periodically compares the measured value to an adjustable *threshold* parameter. Once a pulse exceeds the set threshold, counter registers become active and record information about the pulse. On the falling edge, the counters stop and their values are read out. This process usually takes up to 475 ns and is sometimes referred to as the *dead time*, since the pixel is not responsive to any changes in the voltage pulse while the reading operation takes place. After its completion, the pixel circuit is reset and the analysis begins anew.

The meaning of the values stored in the counters is determined by the operation mode, which is usually the same for all pixels of the detector. In Timepix3, three modes are available:

ToA Mode In this mode, the detector keeps a global clock signal with pulsing frequency up to 40 MHz. Once the analog pulse exceeds the set threshold, the first rising edge of the clock signal latches the current timestamp into the 14-bit register. The register thus contains the *time of arrival* (abbr. ToA) accurate up to $(40\ \text{MHz})^{-1} = 25$ ns.

ToA & ToT Mode This mode is an extension of the ToA mode, wherein the circuit also tracks the number of clock cycles pulse spends above the set threshold. The cycle count is referred to as the *time over threshold* (abbr. ToT) and is stored in the 10-bit register.

Event Counting Mode In this mode, the pixel circuit does not reset after the second crossing of the set threshold, eliminating the undesirable dead time. Instead, the circuit tracks the number of such events in the 10-bit register. In addition, the 14-bit register holds the *integral time over threshold* (abbr. iToT), which can be viewed as a sum of ToT values across all observed events.

To increase time resolution in ToA operation modes, pixel circuits may be configured to utilize a second, faster clock signal provided by voltage-controlled

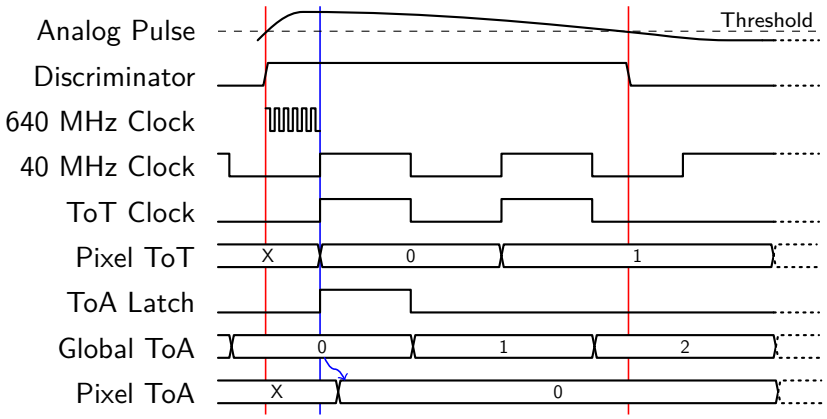


Figure 2.3: Schema of Timepix3 signals during a single event [1]. The red vertical lines mark the upward and downward crossing of the threshold. The first upward edge of the primary clock signal after event start is marked by a blue line.

oscillators (abbr. VCOs). The pulsing frequency of this signal can reach 640 MHz, yielding time resolution up to $(640\ \text{MHz})^{-1} = 1.5625$ ns. If enabled, the 4-bit register stores the number of fast clock cycles from the moment the set threshold was exceeded until the first rising edge of the slower primary clock signal.

During ToA measurements with fast VCO enabled, the 14-bit ToA value t_{slow} and the 4-bit value t_{fast} (abbr. fToA) are usually combined into so-called *pixel timestamp* as follows.

$$T(t_{slow}, t_{fast}) = 25 \cdot t_{slow} - 1.5625 \cdot t_{fast}\ [\text{ns}] \tag{2.2}$$

2.2.5 Readout Modes

As the name suggests, the *readout mode* determines the process used to retrieve measured information from a Timepix3 detector. The hardware design supports two modes:

Sequential Mode In this mode, pixel circuits keep information recorded in counter registers until explicitly queried by an external command. This mode resembles polling and may result in lower power consumption.

Data-Driven Mode In this mode, pixel circuits are read and reset as soon as possible after recording new information. The detector proactively notifies the user about newly available data. This mode resembles pushing and attempts to minimize pixel dead time.

2.3 Detector Calibration

Before practical operation, Timepix3 detectors undergo various calibration, equalization and correction procedures. Their goal is to correlate detector measurements with known response, and minimize known systematic errors and biases due to hardware design.

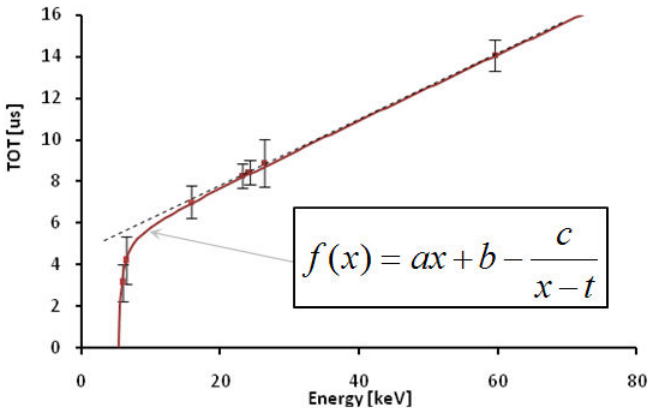


Figure 2.4: The dependence of ToT counter value (vertical axis) on the deposited energy in a pixel (horizontal axis) [2].

2.3.1 Threshold Equalization

Due to technological inaccuracies induced by the manufacturing process, individual pixels of a detector may produce varied response under the same configuration. The aim of threshold equalization is to eliminate this behavior by measuring the differences and altering the configuration in order to minimize their magnitude.

At the beginning of the process, the threshold level is set to the same value for all pixels. After a measurement is performed, pixel response is examined and additional corrective bias voltage is introduced into the amplifier channel of each pixel. Upon completion, noise levels of individual pixels are equalized.

2.3.2 ToT Energy Calibration

Recent works have shown that ToT counter values are related to the amount of energy deposited in pixels by incident particles [2]. The aim of the energy calibration process is to estimate this relationship (illustrated in Figure 2.4) by performing measurements with characteristic X-ray fluorescence lines.

The result of the process are four fitted constants $a, b, c, t \in \mathbb{R}$ for every pixel [23]. For a pixel of ToT value x, the deposited energy is then approximated by function

$$E(x \mid a, b, c, t) = ax + b - \frac{c}{x - t} \text{ [keV]}. \tag{2.3}$$

2.3.3 Column Time Offset Correction

Due to hardware malfunctions, pixel ToA values may be consistently offset by additive constants. Curiously, this additive offset has been observed to be equal for pixels of a single column.

To correct against this, ToA offsets are measured in all columns. In subsequent measurements, inverse values of these offsets are added in order to obtain corrected ToA values.

2.3.4 Time-walk Correction

Time-walk is an undesirable effect caused by different slope of signals with different amplitudes. If left uncompensated, energy-dependent error is induced in pixel timestamps. For instance, neighboring pixels ionized by the same particle may observe timestamps offset depending on the distribution of the collected charge between them.

Since simultaneous measurement of both the amplitude and the timestamp is supported by Timepix3 detectors, time-walk may be corrected by prior calibration [24]. The result of the process are two fitted constants $c, d \in \mathbb{R}$. For a pixel with threshold energy E_0 observing calibrated energy E (both quantities given in keV), the time-walk offset is then estimated by function

$$\Delta T(E \mid E_0, c, d) = \frac{c}{(E - E_0)^d} \text{ [ns]}. \tag{2.4}$$

To correct against time-walk, ΔT is simply subtracted from the original pixel timestamp.

2.3.5 Drift Time Calibration

The aim of drift time calibration is to accurately reconstruct depth information from pixel timestamps, effectively estimating three-dimensional location of the incident particle within the detector at the time of interaction. This procedure has been described and evaluated in previous works [18].

To reconstruct the depth, the time of charge drift within the detector is first measured in radiation environment, where particles enter and exit the detection medium at opposite sides. This way, the speed of charge drift v_{drift} is estimated. At a later time, clusters of pixels ionized by the same particle may reconstruct the depth using function

$$Z(T \mid T_0, v_{drift}) = (T - T_0) \cdot v_{drift} \text{ [}\mu\text{m]}. \tag{2.5}$$

Here, T is the corrected timestamp of the reconstructed pixel, T_0 is the lowest corrected timestamp within the pixel cluster (both given in ns). Lastly, v_{drift} is the drift speed fitted from prior calibration (given in μm/ns).

2.4 Katherine Readout

To facilitate communication between a computer and a Timepix3 detector, readout devices are utilized. In this work, focus is put on the recently developed *Katherine* readout.

2.4.1 Hardware Parameters

Katherine readout [25] (shown in Figure 2.5) is an autonomous device capable of operating a single Timepix3 detector. Unlike other readouts such as FITPix [26]

(a) Katherine with Timepix3. (b) Annotated ports of the readout.

Figure 2.5: Katherine readout [25].

or SPIDR [27], the readout can be viewed as an embedded computer rather than a direct connection interface.

The readout connects to the detector with a 68-pin VHDCI[3] connector and automatically provides the detector with an appropriate bias voltage up to ±300 V. In addition, the device provides GPIO[4] interface for up to 4 signals, which may be used for triggering or synchronization in advanced measurement setups.

Designed to be remotely operated in possibly harsh radiation environments, the readout supports a long distance Gigabit Ethernet connection up to 100 m. For the event of connection failure, the device is equipped with 1 GB of DDR3 memory, capable of temporarily buffering measured data.

2.4.2 Data Acquisition Control

For operation, Katherine can be controlled by other network agents using a proprietary UDP-based[5] protocol. Its command set features 36 instructions [28], capable of configuration, status inquiry and control of data acquisition.

Apart from direct control, the readout may also be configured to perform autonomous acquisition and periodically deposit measurement data in a known storage facility via SFTP[6].

For performance considerations, the readout implements certain correction procedures directly in hardware. For instance, pixel column time offsets are automatically determined during the readout power-up sequence and subtracted from pixel ToA values during measurements.

[3]Very High Density Cable Interconnect
[4]General Purpose Input/Output
[5]User Datagram Protocol
[6]Secure File Transfer Protocol

3. Acquisition Control

The localization process described in the following two chapters relies on a source of pixel data, which may be viewed as a continuous sequence of records with various attributes. Such a sequence is produced by the process of data acquisition, which requires Timepix3 hardware and may take place over an extended period of time. With this motivation, the focus of this chapter is on the specifics of communication and control of a Timepix3 detector by means of a Katherine readout.

Since data acquisition is a general task required in many practical applications, software components which interact with the readout hardware are deliberately isolated from the rest of the system. The implementation of these components, which was originally created only for the purposes of this work, is concentrated in a stand-alone C library package with the motivation of possible reusability in other projects. The C programming language was selected since it offers desirable performance capabilities, and its flexible interoperability allows easy integration into other applications.

The processing logic described in the remaining chapters of this work can be viewed as a documented usage example of this library. Consequently, this chapter may be considered to be its user documentation.

3.1 Operational Overview

Similarly to various hardware controllers in computers, Katherine readout acts as a controller of Timepix3. Upon receiving commands from a user application, it autonomously engages in an exchange of low-level instructions with the detector, leaving the application free to perform other tasks. At a later time, the application is asynchronously notified of the success or failure of the issued command.

Katherine may thus be viewed as an intermediary component, which facilitates the communication between the computer and the Timepix3 detector. However, it may be also viewed as an autonomous computing unit, to which low-level detector management tasks are offloaded by the computer in order to minimize CPU processing strain. In a typical scenario, Timepix3 detector is mounted in the vicinity of sources of ionizing radiation. Since this type of radiation is harmful to computing equipment, Katherine is usually placed in a shielded container, or preferably outside the contaminated area. Both the readout and the computer are connected to the same network infrastructure, which relays commands and measured data. This setup is illustrated in Figure 3.1.

In the computer which communicates with the Katherine readout, an application uses the operating system network interface to exchange messages with the device. It is at this point, where the presented hardware library would be utilized to abstract readout communication interface and protocol.

3.1.1 Typical Operation

Conventionally, Katherine may be operated as follows:

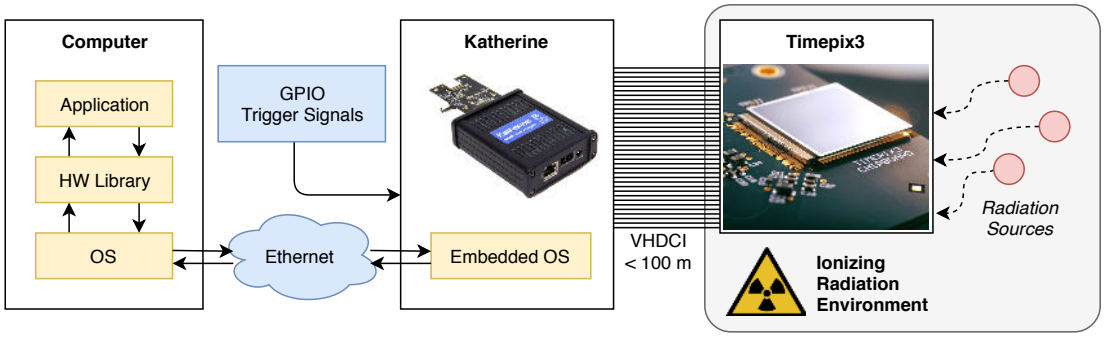


Figure 3.1: Expected hardware setup used in measuring ionizing radiation with Timepix3 detectors and Katherine readout.

1. The application queries the readout state and checks all necessary preconditions (e.g. sensor presence, communication lines).

2. The application configures the readout with acquisition parameters (e.g. DAC registers, bias voltage, acquisition duration).

3. The application initiates data acquisition. From this point on, the readout starts producing measurement data. The application processes this data for its purposes.

4. During acquisition, the application periodically monitors important indicators (e.g. temperature, voltage, communication performance).

5. Eventually, the acquisition is stopped and the measurement data stream is terminated. The application is asynchronously notified of this event.

A typical exchange of messages between the application and the readout is depicted in Figure 3.2. In all interactions, the application sends commands and receives acknowledgments upon their completion. The commands may cause exchange of multiple low-level instructions (case A), a single instruction (case B) or no instruction (case C) between the readout and the Timepix3 detector.

During data acquisition, the readout autonomously communicates with Timepix3 and periodically transmits batches of pixel measurement data to the application (cases D, E). Occasionally, the readout also transmits frame measurement data (case F), which may not require prior detector communication.

3.1.2 Communication Channels

In order to efficiently exchange commands, configuration and measurement data with user applications, Katherine is connected to a computer network via the Ethernet bus. On this bus, the device communicates using a proprietary UDP-based protocol [28].

The communication is divided into two parallel channels:

Control Channel This channel is utilized for sparse communication, such as issuing commands and receiving their acknowledgments. Both the readout and the application are permitted to initiate communication on this channel.

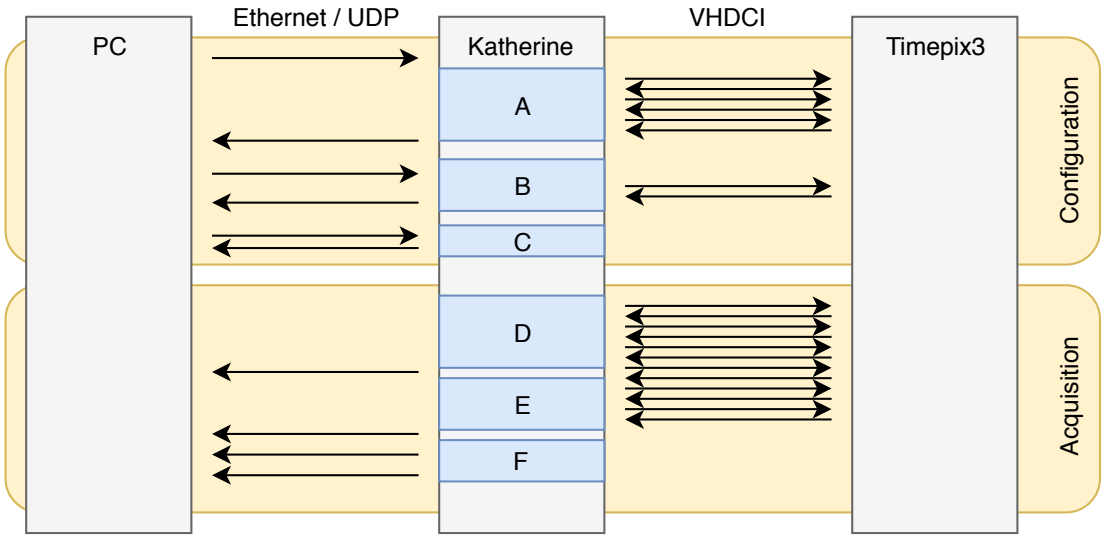


Figure 3.2: Communication diagram showing typical operation of Katherine readout.

Data Channel This channel serves for dense communication, such as unacknowledged measurement data transmissions. Only the readout can initiate communication on this channel.

In practical implementation, Katherine is identified on the bus by a single static IP address, whereas individual communication channels correspond to particular ports.

3.1.3 Issuing Commands

Commands are issued to the readout over the control channel in the form of individual UDP datagrams. To initiate a command, an application transmits 8 byte datagram containing a known header and optionally a multitude of parameters. Upon receiving, the readout processes the command and responds with a similar 8 byte datagram containing command acknowledgment along with optional values, depending on the command type.

The available commands may be semantically divided into three categories:

Status Commands These commands inquire about various observable parameters of the readout or the detector (e.g. temperature, voltage, performance or configuration).

Configuration Commands These commands alter the internal state of the readout or the detector, usually for purposes of subsequent data acquisition.

Acquisition Control Commands These commands initiate or abort an ongoing data acquisition.

While status and configuration commands may be expected to always produce a particular outcome, results of acquisition control commands may vary depending on whether the readout is performing data acquisition at the time of execution.

3.1.4 Receiving Measurement Data

During data acquisition, the readout generates a sequence of records referred to as *measurement data*. This data is transmitted asynchronously over the data channel in the form of 6 byte long UDP datagrams. Unlike command datagrams, measurement data are not acknowledged by the receiving side, opening the possibility of data loss during unreliable network transmissions.

Measurement data may be divided into two categories:

Pixel Data This measurement data contains information about a single active pixel in the Timepix3 sensor. The contents and structure of pixel data are determined by the operation mode requested at acquisition start.

Frame Data This measurement data may contain various information about the ongoing data acquisition (e.g. timestamps, number of lost pixels).

3.2 Library Architecture

The purpose of the presented library is to abstract all communication with Katherine readout from a user application, thereby allowing its developer to focus on the task of data processing rather than implementation of network protocols.

The library has been divided into the following components (illustrated in Figure 3.3):

Network Communication This component is responsible for abstracting network interface for UDP communication. Such interface may be for instance BSD sockets on the Linux platform.

Internal Command Interface This component encodes commands into datagrams and transmits them using the network communication layer. It is also responsible for properly formatting command parameters and decoding their acknowledgments, if applicable.

Public Command Interface This component implements high-level command interface for the user application, often performing multiple calls to the internal interface to complete a single command.

Measurement Data Listener This component efficiently processes incoming datagrams containing measurement data. The user application is notified by means of high-level events of interest.

Configuration Parser This component is responsible for parsing proprietary formats of configuration files into internal memory structures.

3.2.1 Network Communication

The purpose of the network communication layer is to form a close abstraction of the system network communication interface, thereby making the library easily portable to a multitude of platforms.

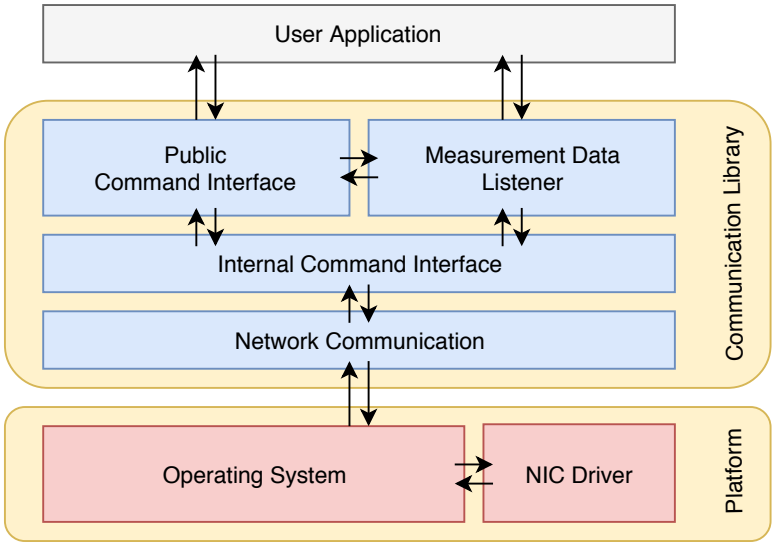


Figure 3.3: Architecture of library components and their communication relationships (indicated by arrows).

Since the library has been originally developed on a Linux system, the interface of the network communication layer essentially mimics that of BSD sockets. The interface defines the following six operations:

Open & Close Session This initializes (or terminates) a new communication session with a remote host (identified by an IP address and a port number). Optionally, a timeout for blocking transmission operations may be specified.

Send & Receive Datagram This transmits (or receives) a UDP datagram of arbitrary size to the remote host of an open session. Note that unlike conventional socket operations, [1] successful completion of these operations guarantees complete transmission (or reception) of the *entire* datagram.

Lock & Unlock Mutex This locks (or unlocks) a mutual exclusion synchronization primitive bound to an open session. While its state is not checked or assumed at any point during the execution of other operations, it is utilized by other components striving to offer thread-safe implementation.

3.2.2 Internal Command Interface

The internal command interface implements features required for transmission of commands using the network communication layer. As its name suggests, the interface is designed for exclusive usage by other library components and is not intended to be called by user applications. Instead, their developers are encouraged to utilize the public interface described in the following section.

The features implemented by the internal interface include operations common to transmission of most commands. For instance, the interface implements generic functions for transmission of a command with up to two parameters, functions capable of waiting for acknowledgments and functions to decode return values from acknowledgment datagrams. In addition, the internal interface also includes

[1] See the entries `sendto(3)` and `recvfrom(3)` in the POSIX Programmer's Manual [29].

definitions of command codes in accordance with the Katherine command set specification [28].

Similarly to the network communication layer, the internal interface makes no guarantees about thread safety. For that reason, synchronization is performed by other components prior to calling the interface.

3.2.3 Public Command Interface

The public command interface is the main expected point of interaction between the library and user applications. It implements thread-safe and reliable transmission of high-level commands to the Katherine readout.

The interface implements over 20 high-level commands for state inquiry, configuration and acquisition control. Command execution has synchronous semantics, waiting for positive acknowledgments from the readout before reporting successful completion to the user application. As a consequence, this also allows detection of transmission reliability in otherwise unreliable UDP communication environment.

In order to avoid possible misinterpretation of acknowledgments corresponding to simultaneously issued commands, the public command interface utilizes mutual exclusion capabilities implemented by the network communication layer. Before execution, each command acquires a lock of the open control channel. At that point, its thread may be suspended until another partially executed command completes. With the lock acquired, the command sends a sequence of command datagrams via the internal command interface and waits for their corresponding acknowledgments, relying on the nonexistence of outside interference guaranteed by the acquired lock. After completion, the lock is released, possibly waking up other waiting threads.

In code, the interface defines the `katherine_device_t` structure, which represents a single Katherine readout. Internally, this structure holds the control and the data channel information and tracks their state. A device is initialized with the readout IP address, and upon successful initialization opens communication sessions to both channels. Commands provided by the public interface operate with these sessions. For an example of such usage, see Listing 1.

```c
int result;
katherine_device_t device;

result = katherine_device_init(&device, "192.168.1.126");
if (result != 0) perror("katherine_device_init");

float temp;
result = katherine_get_readout_temperature(&device, &temp);
if (result != 0) perror("katherine_get_readout_temperature");

printf("The temperature is %.3f degrees.\n", temp);
katherine_device_fini(&device);
```

Listing 1: A simple C program, which establishes connection with the readout and prints the temperature measured by its internal sensors.

3.2.4 Measurement Data Listener

The measurement data listener is responsible for receiving and efficient decoding of measurement data during active acquisitions. Similarly to the public command interface, the listener provides a thread-safe synchronous call interface to user applications. Apart from processing the received measurement data, this component is also responsible for controlling the entire data acquisition process. It is therefore assumed that at the time of initialization, the operated readout is in idle state. The provided interface allows user applications to configure the readout, initiate the data acquisition procedure and retrieve its results.

Before acquisition may be initiated, user application is required to provide necessary configuration and event handler callbacks, which are notified during various events of interest. After a successful acquisition start, the readout begins to asynchronously send measurement data to the computer through the data channel. For that reason, the application is required to relinquish control of one processing thread to the measurement listener in order to process and decode such messages. The internal logic of the listener then executes the provided callbacks at appropriate times, and provides the application with measurement data in well-defined format.

Data acquisition with Katherine is split into units called *frames*. During a frame, the detector performs time-limited acquisition based on the provided configuration parameters, most importantly the operation and readout mode. At that time, the readout also transmits pixel measurement data, which describe the activity of individual pixel circuits during the frame acquisition period. At the end of each frame, the readout transmits frame measurement data, which describe overall state of the readout (e.g. the number of lost pixels).

To avoid overhead due to frequent callbacks, the measurement data listener includes a storage buffer for pixel measurement data of user-defined size. In the course of message decoding, this buffer is filled with inbound information. The provided interface defines callbacks for the following events of interest:

Frame Started This event occurs after of a new frame has been initiated.

Frame Ended This event occurs after the acquisition of the current frame has ended. The provided data include timestamps corresponding to the acquisition period, the number of sent and received pixel measurement data, and information, whether the frame has been successfully completed.

Pixel Measurement Data Ready This event occurs after a group of pixel measurement data has been prepared for reading by the application. Note that the amount of ready items does not necessarily have to utilize full size of the buffer.

In code, the measurement data listener component is represented by the `katherine_acquisition_t` structure. Its example usage along with appropriate event callbacks is illustrated in Listing 2.

```c
typedef katherine_px_f_toa_tot_t pixel_t;
void my_handler(const void *pixels, size_t count) {
    const pixel_t *px = (const pixel_t *) pixels;
    for (size_t i = 0; i < count; ++i) {
        /* do something application-specific with px[i] */
    }
}

/* ... */

int result;
katherine_device_t device;
katherine_config_t config;
/* ... initialize device, set config ... */

katherine_acquisition_t acq;
result = katherine_acquisition_init(&acq, &device, mdbuf_size,
    ↪ pxbuf_size);
if (result != 0) perror("katherine_acquisition_init");

acq.handlers.pixels_received = my_handler;
result = katherine_acquisition_begin(&acq, &config,
    ↪ READOUT_DATA_DRIVEN);
if (result != 0) perror("katherine_acquisition_begin");

/* data incoming now, relinquish thread to the listener */
result = katherine_acquisition_read(&acq);
if (result != 0) perror("katherine_acquisition_read");

katherine_acquisition_fini(&acq);
```

Listing 2: An example C program, showing the usage of the measurement data listener component in order to begin acquisition, receive and process data.

3.2.5 Configuration Parser

The configuration parser is the last of the three components provided to applications by the presented library. Its purpose is mostly auxiliary – to ease reading of proprietary file formats used by other measurement tools for storing acquisition configuration.

The parser operates on a valid memory range loaded with the contents of the file. Its usage is shown in Listing 3.

```
1  void *buffer;
2  /* ... load file contents into the buffer ... */
3
4  int result;
5  katherine_bmc_t bmc;
6
7  result = katherine_bmc_init(&bmc);
8  if (result != 0) perror("katherine_bmc_init");
9
10 result = katherine_bmc_load(&bmc, buffer);
11 if (result != 0) perror("katherine_bmc_load");
12
13 katherine_bmc_fini(&bmc);
```

Listing 3: An example C program, showing the usage of the configuration parser on one of the supported formats.

4. Compton Event Detection

In order to localize a γ-ray source in space by a Compton camera, a sufficient number of Compton cones must first be generated. As shown in previous sections, such cones may be obtained from pairs of photon interactions with semiconductor detectors. For the purposes of this chapter, these pairs of interactions are referred to as *Compton events*.

Since the algorithm described in Section 2.1.5 is sufficiently direct to be viewed as a mapping from the set of Compton events to the set of Compton cones, the primary challenge of this task lies in event detection rather than parameter fitting. With this motivation, the focus of this chapter is on the approaches used to ensure correct and efficient detection of Compton events.

4.1 Hardware Setup

The presented approach requires both proprietary hardware and software. Besides the communication infrastructure, the components of the hardware setup (illustrated in Figure 4.1) can be divided into three categories as follows:

Detection Unit This component is responsible for facilitating and observing γ photon interactions. It thus needs to be positioned in the vicinity of a suspected radiation source and use radiation-hard materials. It consists of a multitude of Timepix3 detectors.

Readout Unit This component maintains connection between the detection and the processing unit. It consists of multiple Katherine readouts.

Control Unit This component controls the readout unit, receives measured data and generates the aggregated output of the Compton camera. It consists of a computer with sufficient processing capabilities.

4.1.1 Detection Unit

To detect Compton events, conventional Compton camera architecture is used. The detection unit is thus comprised of a multitude of scatterers and a single absorber positioned in telescopic configuration. To implement both types of components, Timepix3 detectors are used.

Since the aim of scatterer detectors is to facilitate Compton scattering – the first of the two interactions of a Compton event, their detection medium is silicon. This choice is motivated by the desirable properties of the material, such as wide energy range, where Compton scattering dominates other types of γ photon interactions.

In order to facilitate photoelectric absorption, the absorber detector uses cadmium telluride as its detection medium. Furthermore, in order to maximize the probability of interaction, the semiconductive sensor layer is significantly thicker than in scatterer detectors.

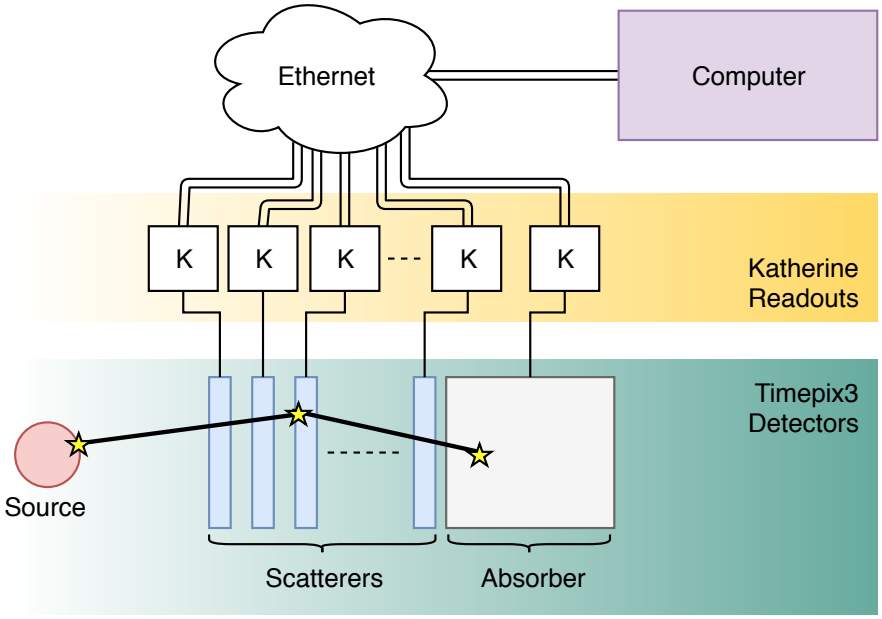


Figure 4.1: Diagram of the proposed Compton camera setup.

Both absorber and scatterer detectors are configured to perform continuous data-driven acquisition in the ToA & ToT operation mode. Moreover, equalization, calibration and correction procedures described in Section 2.3 are applied to normalize detector outputs prior to data acquisition.

4.1.2 Readout Unit

As the name suggests, the readout unit is comprised of a multitude of Katherine readouts, each dedicated to a single Timepix3 detector from the detection unit. For easy manipulation, readout devices are not connected to detector boards directly, but rather indirectly using a VHDCI extension cable.

By the hardware specification [25], Katherine readouts are capable of controlling detectors up to 100 m of distance. This implies that while the detection unit must be manufactured with considerations for the possibly harsh radiation environment, the readout unit may be placed outside or in a shielded container.

The readouts individually communicate with the control unit over the Gigabit Ethernet bus. While the bus communication infrastructure is not explicitly defined, it may be easily implemented by an Ethernet switch with a sufficient port count and performance capabilities.

4.1.3 Control Unit

The purpose of the control unit is to issue commands to Katherine readouts, receive and process the measured data. It is thus implemented by a computer with networking capabilities and sufficient processing power.

While the computer is generally not required to be positioned anywhere close to the remaining components, the amount of network infrastructure between units is usually minimized in order to decrease network communication latency and packet loss probability.

The architecture of the computer software assumes a multi-threaded computing environment. Under this assumption, the software creates multiple threads with assigned roles:

Readout Thread This thread is responsible for communication with a single Katherine readout. One thread therefore spawned for every readout. The purpose of the thread is to process pixel measurement data independently on the outputs received from other readouts.

Compton Event Thread This thread is responsible for aggregation of outputs of all readout threads. For this reason, only one such thread is created. The result of the aggregation process is a sequence of Compton events.

Monitoring Thread This thread is serves auxiliary purposes, such as monitoring of temperature, communication performance and state of individual readout devices.

Reconstruction Thread The purpose of this thread is to consume multiple Compton events and reconstruct three-dimensional probability distribution by the algorithm described in Chapter 5.

User Interface Thread This thread maintains the user interface, displays the state of the device and receives commands from the human operator.

4.1.4 Time Synchronization

Since each Timepix3 detector assembly has its own set of internal clock signals, pixel timestamps from different detectors do not necessarily represent the same points in time. For this reason, readouts are synchronized prior to analysis.

The synchronization process is performed at the beginning of device operation and relies on inducing predictable responses to a simultaneous phenomenon in all detectors. By comparing timestamps of such responses from different detectors, the difference between clock frequencies and offsets is identified. After examining a sufficient amount of data, the procedure is completed.

Rather than alteration of detector clock signals, the result of the time synchronization process is a set of fitted linear functions, which map detector-generated timestamps to so-called *global timestamps*. If performed correctly, it holds that simultaneous events receive the same global timestamps.

4.1.5 Coordinate System

For simplicity, the coordinate system used is based on that of Timepix3 detectors [1]. Since the detectors are oriented in the same direction and arranged in telescopic configuration, their detection planes are parallel to each other. This way, directions of the X and Y axes are uniquely defined and consistent across all detectors. The Z axis is then defined as the line intersecting pixels with coordinate $(0, 0)$ of each detector.

Assuming that detectors are well-ordered and numbered $D_1, D_2, \ldots, D_n$ for $n \geq 2$ such that detectors $D_1, D_2, \ldots, D_{n-1}$ are scatterers and D_n is the absorber,

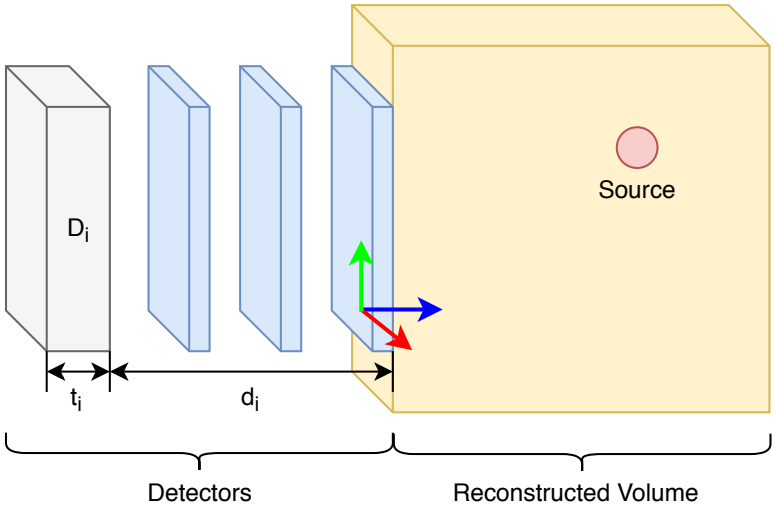


Figure 4.2: Detector depth d_i and thickness t_i.

the origin is placed in the corner of D_1 at the point of intersection with the line of Z-axis. This is illustrated in Figure 4.2.

The direction from D_1 towards the rest of the detectors is defined as the negative Z-axis direction. For $i \leq n$, detector D_i has thus two spatial parameters:

Thickness t_i This parameter measures the thickness of the detection medium in the direction of the Z-axis.

Depth d_i This parameter is the Z-component of the point of D_i closest to the origin.

This completes the definition of the camera coordinate system. Note that by the definition of the origin, $d_1 = 0$. Since it holds that all $d_i \leq 0$, $t_i > 0$, and $d_{i+1} \leq d_i - t_i$ for $i < n$ by the assumption that detectors are well-ordered, $\{d_i\}_{i=1}^{n}$ is a strictly decreasing sequence.

Furthermore, the reconstruction process described in the next chapter relies on the definition of a so-called *volume* – a cuboid body, in which the unknown emission source is located. For spatial description of this body relative to the camera assembly, five additional parameters are defined:

Volume Depth This parameter determines the dimension of the volume in the direction of the Z-axis. As its name suggests, it is related to the depth parameter of the detectors.

Volume Offsets These four parameters (marked in Figure 4.3) determine the dimensions of the volume in the directions of the X- and Y-axis. They represent two vertical and two horizontal distances between the edge of the detector and the edge of the volume.

Since the X- and Y-dimensions of individual detectors are known, the offsets determine the dimensions of the volume as well as the alignment of the camera relative to the volume. For instance, zero offsets imply a volume of X- and Y-dimensions matching those of the detectors.

For the reasons of consistency, all spatial parameters use Timepix3 pixels as the unit of measurement. Note that 1 pixel corresponds to 55 μm.

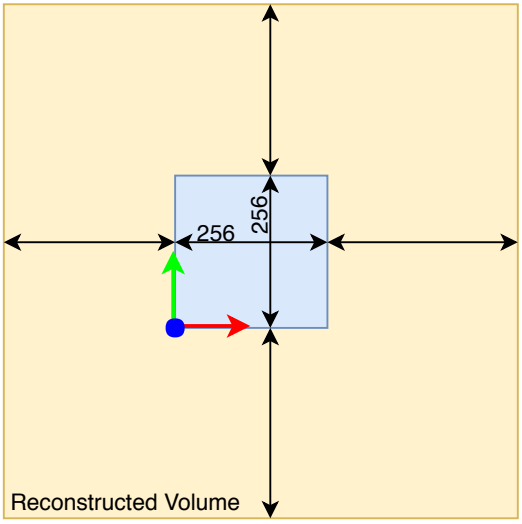


Figure 4.3: Volume offsets.

4.2 Pixel Stream Processing

The rest of this chapter describes the algorithms executed by the control computer in order to detect Compton events during continuous data acquisition.

For low-level communication with readouts, the computer uses the library package described in Chapter 3. Assuming successful startup and configuration of all components, the most dominant task performed by the software consists of processing received measurement data, which can be viewed as a continuous sequence of active pixels. For improved performance, pixels incoming from individual readouts are processed independently in parallel by readout threads. This section focuses on the actions of one such thread.

The architecture of a readout thread can be classified as a conventional producer/consumer design pattern, wherein the program is divided into a *producer* and a *consumer* component. During operation, the producer generates a sequence of items with different contents but identical data structure. The sequence is usually buffered in batches and eventually processed by the consumer component, which yields the requested results.

In the context of a readout thread, the remote readout device may be viewed as a producer of pixels. However, due to the nature of the Compton event detection problem, more than one instance of the pattern is required. For that reason, the algorithm is decomposed into a chain consisting of several stages, which transform and aggregate the received data into the output format. Strict ordering between stages is defined, making the consumer of each stage the producer of the next one (this is depicted in Figure 4.4). The output of the last stage is a sequence of *clusters*, which undergoes coincidence group matching in the Compton event thread.

4.2.1 Calibration Evaluation

The purpose of the first stage is to evaluate various correction and calibration functions described in Section 2.3. The input consists of so-called *raw pixels*,

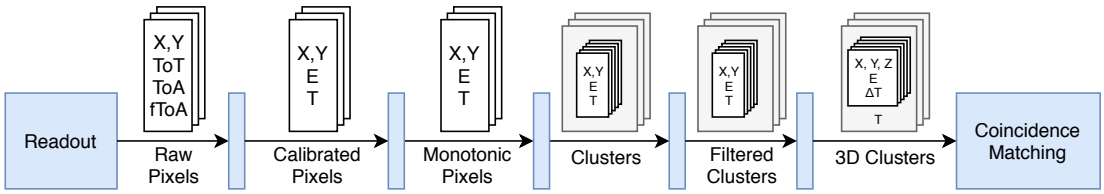


Figure 4.4: Processing chain corresponding to a readout thread.

which contain outputs of Timepix3 data acquisition in the ToA & ToT mode with fast VCO enabled. Each pixel thus contains information obtained directly from the readout:

Point of Interaction A point in the detector plane, denoted by X- and Y-coordinate.

ToT Value 10-bit value containing Time-over-Threshold information related to energy deposited in the detector.

ToA, fToA Values 14-bit and 4-bit values containing Time-of-Arrival information related to the pixel timestamp.

Before processing, the point of interaction is used to locate pixel-dependent calibration constants in an auxiliary lookup table. With the values loaded, the calibration function 2.3 is evaluated, transforming the ToT value into deposited energy E. Based on the X-coordinate, a constant column offset is subtracted from the ToA value, and the pixel timestamp T is calculated by combining ToA and fToA information in Expression 2.2. Lastly, depending on the energy E, the timestamp T is shifted one more time in order to compensate against the time-walk distortion effect.

The result of the process (summarized in Algorithm 1) are items in one-to-one correspondence with the input pixels. Each output item is comprised of the following fields:

Point of Interaction A two-dimensional point matching that of the input pixel.

Deposited Energy The amount of energy (in keV) deposited into the pixel by the incident particle.

Timestamp The time of the event start (in ns), expressed relative to the internal clock of the detector.

Algorithm 1 Calibration Evaluation

Parameters: calibration functions $E(x \mid X)$, $\Delta T(E \mid X)$

1: **for all** incoming pixels $(X, x_{ToT}, x_{ToA}, x_{fToA})$ **do**
2: $\quad E_p \leftarrow E(x_{ToT} \mid X)$
3: $\quad T_p \leftarrow 25 \cdot x_{ToA} - 1.5625 \cdot x_{fToA} - \Delta T(E_p \mid X)$
4: $\quad$ ReportPixel(X, E_p, T_p)

4.2.2 Ensuring Time Monotony

The second stage is responsible for imposing monotonic ordering on the sequence of incoming pixels with respect to their timestamps. This property is not necessarily guaranteed by the readout and is required by the subsequent spatial clustering stage.

Formally, the property of time monotony is defined for an ordered sequence of pixels as follows.

Definition 1. *Let T_i denote the timestamp (in ns) of the i-th pixel from an ordered pixel sequence $\mathcal{P}$. Then $\mathcal{P}$ is considered to be monotonic with respect to time if for all $i \leq j$ holds that $T_i \leq T_j$.*

Note that while the received pixel sequence is not implicitly guaranteed to be monotonic, due to the hardware design of the readout device, the property is always satisfied *to a certain extent*. This can be captured by the following, more general definition.

Definition 2. *Let $t \geq 0$ and T_i denote the timestamp of the i-th pixel from an ordered pixel sequence $\mathcal{P}$. Furthermore, let S_i denote the time of transmission[1] of the i-th pixel. The sequence $\mathcal{P}$ is t-monotonic if for all $i < j$ such that $S_i < S_j - t$ holds that $T_i \leq T_j$.*

Comparing the wording of the definitions, one can see that both require the sequence of timestamps T_i to be somewhat non-decreasing. However, unlike conventional monotony, t-monotony is designed to relax the scope of this requirement for a fixed local time interval of duration at most t. Consequently, for $s \geq t$, every t-monotonic sequence is also s-monotonic.

One desirable property of t-monotonic sequences is that they may easily be transformed into monotonic sequences in the conventional sense by means of *buffering*. As the name suggests, the algorithm relies on a dynamic ordered data structure, capable of holding an arbitrary number of pixels for a limited time duration. To transform a t-monotonic sequence into a monotonic sequence, its elements are sequentially inserted into the structure, possibly re-arranging its contents in the process, and removed from it at a later time. Upon removal, the elements are enqueued into the transformed sequence, which is guaranteed to satisfy the required property of monotony.

A simple way of understanding the motivation behind this approach involves considering pairs of elements of a t-monotonic sequence, which would violate the general monotonic property. If monotony is viewed as an ordering with respect to timestamps, these are examples of disorder within the sequence. The definition of t-monotony ensures that locally, elements of each such pair are not transmitted too far behind each other. In fact, t represents a constant upper bound on the maximum time duration between such transmissions.

When processing elements of the sequence online, it is not always clear whether the currently transmitted element is a member of a disordered pair or not. To instantly recognize such elements at all times would imply a certain degree of clairvoyance on the part of the algorithm. Instead, each element of the sequence

[1] In this abstraction, pixels are associated with two time points. While the original event is detected at time T_i, the information about the detected event is transmitted at a later time S_i.

is retained for a time at least t, allowing its successors with earlier timestamps to possibly surpass it within the buffer. Once an element with transmission time S_j is added to the buffer, the element of the buffer with the lowest timestamp T_i may be removed if $S_i < S_j - t$. If the element satisfied this condition and was a member of a disordered pair, contradiction of the t-monotony of the sequence could be easily derived.

Assuming a constant fraction of disordered pairs in the sequence and uniform transmission delay, the value of t influences the average number of elements in the buffer. For lower values, the buffer may be evacuated more frequently and will therefore hold less items. Since the number of items in the buffer may influence the time complexity of its insert and delete operations, finding the smallest possible t is critical for achieving good performance.

In practical implementation, the sequence of pixels consumed by the second stage of the program is t-monotonic for $t = 600$ ms. The buffering algorithm (summarized in Algorithm 2) is thus executed in order to obtain a monotonic sequence in the general sense. The structure and data of the output matches that of the input – the pixels are merely re-ordered. To improve performance, the buffer is implemented by a priority queue, using T_i as a sorting key.

Algorithm 2 Pixel Buffering (with heap data structure)

Parameters: retention duration t

```
1: B ← empty heap
2: for all incoming pixels P do
3:     HeapInsert(B, P)
4:     T_ready ←ReadyTime(B, t)
5:     while |B| > 0 and AccessMinKey(B)≤ T_ready do
6:         P' ←ExtractMin(B)
7:         ReportPixel(P')
8: while |B| > 0 do
9:     P' ←ExtractMin(B)
10:    ReportPixel(P')
```

4.2.3 Spatial & Temporal Clustering

During the operation of Timepix3 detectors, incident particles usually interact with more than one pixel. This motivates the third stage, which is responsible for aggregating pixels into so-called *clusters* – groups of adjacent pixels corresponding to incident particles.

The clustering process is based on two assumptions, which are inferred from the physics of a particle traveling through a solid material:

(A1) During its traversal of the detection medium, the particle interacts with pixels, which consecutively lie in local neighborhood of each other.

(A2) A constant upper bound exists on the difference of timestamps between pairs of consecutively activated pixels.

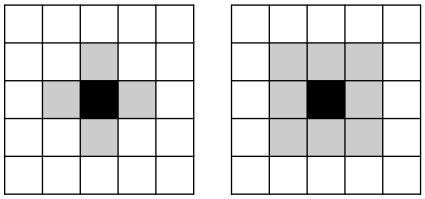


Figure 4.5: Examples of pixel 4-neighborhood (in the left) and 8-neighborhood (in the right). While the pixel is filled in black color, its respective neighborhood is filled with gray.

Both of these assumptions follow from the continuity of particle trajectory and are crucial to derivation of the presented clustering algorithm. The concept of pixel neighborhood referenced in A1 is conventionally defined as follows.

Definition 3. *Let $P \subseteq \mathbb{Z}^2$ be a set of pixels. 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:*

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

The definition of neighborhood describes a set of closely adjacent pixels in a discrete square grid such as the detector pixel matrix (illustrated in Figure 4.5). While the 8-neighborhood is usually preferred, the presented implementation allows to use 4-neighborhood instead, leading to possibly less accurate, but also computationally less intensive algorithm, which may be favorable in performance-critical applications.

To derive the clustering algorithm, a simpler problem may first be considered by replacing A1 with the following assumption:

(A1′) At all times, the detector always observes at most one interaction.

Since time monotony of the input pixel sequence is guaranteed by stage 2, the clustering algorithm for A1′ and A2 is very similar to the buffering algorithm described in the previous section. The algorithm keeps a data structure for temporary storage of pixels. This structure represents *an open cluster* – a group of pixels involved in the singular ongoing interaction with an incident particle.

By A2, a constant upper bound ΔT_{max} exists on timestamp differences between consecutive pixels of a cluster. The algorithm can thus sequentially consume pixels from the input monotonic sequence and distinguish three possible cases:

1. If the open cluster is empty, no interaction is in progress. The incoming pixel marks the beginning of a new interaction. It is therefore inserted into the cluster.

2. If the open cluster contains at least one pixel and the timestamp difference ΔT between the last inserted pixel and the received pixel is smaller than ΔT_{max}, the received pixel is considered part of the current interaction and thus inserted into the cluster.

3. If the open cluster contains at least one pixel and the timestamp difference ΔT is larger than ΔT_{max}, the received pixel marks the beginning of a new interaction, other than that represented by the open cluster. Since by A1′ at most one interaction occurs at all times, the open cluster is reported as final and reset to empty contents. The received pixel is then inserted into the open cluster.

The described algorithm (summarized in Algorithm 3) may be labeled as *location-unaware clustering*. While the algorithm is efficient and simple to implement, it heavily relies on the satisfaction of A1′, which may be quite hard to ensure in reality. Still, the algorithm serves to illustrate the basic principles of temporal clustering. Furthermore, as shown in the rest of this section, the algorithm may be modified to use A1 instead of A1′ at the expense of increased complexity.

Algorithm 3 Location-unaware Clustering

Parameters: temporal upper bound ΔT_{max}

```
1:  O ← {}, T_end ← −∞
2:  for all incoming pixels (X, E, T) do
3:      if |O| = 0 then                                  ▷ new interaction
4:          O ← {(X, E, T)}
5:          T_end ← T
6:      else
7:          ΔT ← T − T_end
8:          if ΔT ≤ ΔT_max then                          ▷ part of the current interaction
9:              O ← O ∪ {(X, E, T)}
10:             T_end ← T
11:         else                                         ▷ new interaction
12:             ReportCluster(O)
13:             O ← {(X, E, T)}
14:             T_end ← T
```

In the algorithm, the assumption A1′ serves primarily to limit the number of open clusters to one. If this assurance is not available, the algorithm has to track multiple open clusters simultaneously. In practice, this also complicates the decision logic executed for every incoming pixel. While the location-unaware clustering algorithm only decides whether the pixel is a part of the singular open cluster or not, the generalized algorithm must also decide to which open cluster the pixel belongs. This motivates the use of spatial adjacency in the formulation of A1.

At this point, the generalized clustering algorithm may be derived. The algorithm utilizes two types of data structures:

Open Cluster List This structure is a list of all open clusters, ordered ascending by the largest timestamp of a pixel within each cluster.

Pixel Pointer Lists These auxiliary structures serve to allow fast lookup of clusters based on spatial coordinates. For every pixel of the pixel matrix,

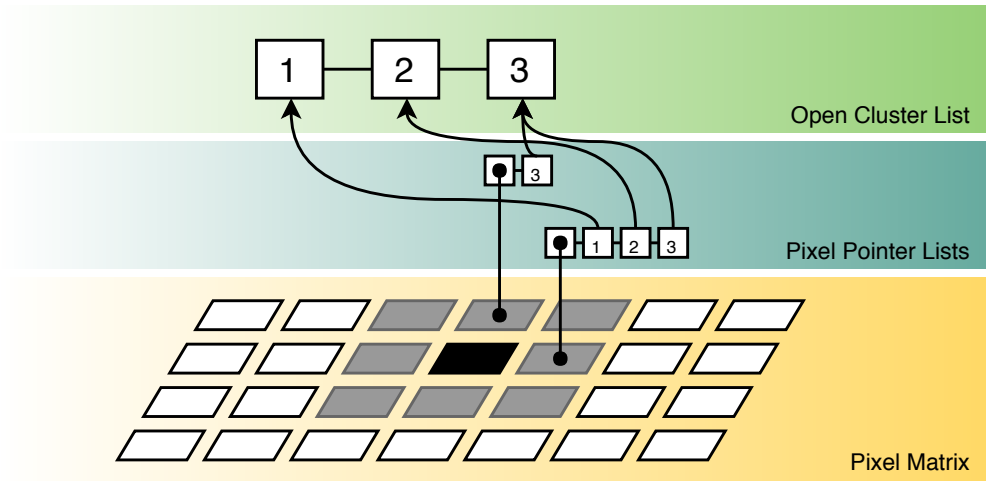


Figure 4.6: Data structures used by the generalized clustering algorithm.

the associated pixel pointer list holds pointers to open clusters, which are known to contain it.

Upon receiving a new active pixel with timestamp T_i and spatial coordinate (x, y), the generalized algorithm (also summarized in Algorithm 4) performs the following operations:

1. The open cluster list is partially scanned from one side, closing, removing[2] and reporting all clusters older than $T_i - \Delta T_{max}$.

2. The pixel pointer lists corresponding to pixels in $\mathcal{N}((x, y))$ are scanned and a list L of unique[3] cluster pointers is compiled. In order to be included in L, the largest timestamp of the cluster must be greater than $T_i - \Delta T_{max}$.

3. Depending on the size of L, one of the three following actions is performed:

   (a) If L is empty, a new open cluster is created and inserted into the open cluster list. The incoming pixel is added to this cluster.

   (b) If L contains exactly one cluster, the incoming pixel is added to it.

   (c) If L contains at least two clusters, their contents are merged. Without the loss of generality, the oldest of the cluster is selected to survive the operation. The incoming pixel as well as all pixels of the remaining clusters in L are then inserted into the selected cluster and cluster pointers in referencing pixel lists are corrected. Lastly, the remaining clusters are removed from the open cluster list.

4. A pointer to the cluster, which contains the incoming pixel, is inserted into the pixel pointer list corresponding to (x, y).

[2]By remembering backward links to referencing pixel pointer lists in the cluster record, all references to a removed cluster containing n pixels can be removed in $\mathcal{O}(n)$ time.

[3]Uniqueness can be achieved by extending cluster records with a zero bit. When compiling the list L, the bit is set to one upon the first encounter, and all records with nonzero bits are skipped. After the list is compiled, bits are reset in preparation for the next pass.

The output of the third stage is a sequence of closed clusters. Each such cluster contains the following attributes:

Pixel List Non-empty list of pixels included in the cluster. Each pixel record includes the attributes described as the output of the second stage.

Smallest/largest Timestamp The timestamps of the first and the last pixel included in the cluster (in ns).

Integral Energy Sum of energies of all pixels included in the cluster (in keV).

Algorithm 4 Clustering

Parameters: temporal upper bound ΔT_{max}, neighborhood function $\mathcal{N}$

1: $O \leftarrow$ empty open cluster list, $P \leftarrow$ empty pixel pointer lists
2: **for all** incoming pixels (X, E, T) **do**
3: ReportOldClusters(O, P, T)
4: $L \leftarrow \{\}$ ▷ look for neighbors
5: **for all** $X' \in \mathcal{N}(X)$ **do**
6: $M \leftarrow$ClustersNearTime$(T, \Delta T_{max}, O, P[X'])$
7: $L \leftarrow L \cup M$
8: **if** $|L| = 0$ **then** ▷ create new cluster
9: $A \leftarrow \{(X, E, T)\}$
10: InsertCluster(O, A)
11: **else if** $|L| = 1$ **then** ▷ add pixel to cluster
12: $A \leftarrow$FindOldestCluster(L)
13: $A \leftarrow A \cup \{(X, E, T)\}$
14: **else** ▷ merge clusters
15: $A \leftarrow$FindOldestCluster(L)
16: $A \leftarrow A \cup \{(X, E, T)\}$
17: **for all** $A' \in L \setminus A$ **do**
18: **for all** $(X', E', T') \in A'$ **do**
19: DeleteClusterPointer$(P[X'], A')$
20: InsertClusterPointer$(P[X'], A)$
21: InsertClusterPointer$(P[X], A)$

4.2.4 Filter Cascade

The goal of the fourth stage is to increase the performance of subsequent stages by excluding all clusters, which clearly do not correspond with expected interactions occurring during a Compton event.

In a cascade, the following predicates are verified:

1. The cluster contains at most 6 pixels.
2. The integral energy falls within the expected energy range characteristic for Compton interactions in the selected detection medium.

3. The cluster does not contain a pixel in close vicinity to a malfunctioning pixel or the edge of the pixel matrix.

The output of the fourth stage (summarized in Algorithm 5) has the same structure as that of the third stage. Clusters, which do not satisfy at least one of the listed predicates, are omitted.

Algorithm 5 Cluster Filter Cascade

Parameters: energy range $[E_{min}, E_{max}]$, set X_{bad} of excluded pixel coordinates

```
1: for all incoming clusters C do
2:     if |C| ≤ 6 then
3:         E_integral ← Σ_{(X,E,T)∈C} E
4:         if E_min ≤ E_integral ≤ E_max then
5:             R ← X_bad ∩ {X | (X, E, T) ∈ C}
6:             if |R| = 0 then
7:                 ReportCluster(C)
```

4.2.5 Drift Speed Evaluation

The aim of the fifth stage is to perform three-dimensional reconstruction of pixel coordinates based on prior drift speed calibration of the Timepix3 detector described in Section 2.3.5.

First, the smallest timestamp within the cluster is subtracted from every pixel, leaving pixels with so-called *relative timestamps*. These are then used to assign each pixel a Z-coordinate calculated from Expression 2.5. The calculated three-dimensional coordinates are then aggregated into *the volumetric centroid* as follows:

$$C = \frac{1}{E} \sum_{i=1}^{n} E_i C_i \tag{4.1}$$

Here, C is the volumetric centroid, $E = \sum_{i=1}^{n} E_i$ is the integral energy of the cluster, n is the number of pixels in the cluster, E_i and C_i denote the energy (in keV) and the reconstructed coordinates of the i-th pixel, respectively.

Lastly, all calculated coordinates C_i and C are transformed from the detector coordinate system to the camera coordinate system by adding the detector depth constant d_j. The output of the fifth stage (summarized in Algorithm 6) are cluster records, which are comprised of the following attributes:

Pixel List Non-empty list of pixels included in the cluster. Each pixel record includes the following attributes:

- **Point of Interaction** A reconstructed three-dimensional point in the camera coordinate system.
- **Deposited Energy** The amount of energy (in keV) deposited into the pixel by the incident particle.

Relative Timestamp The time of the event start (in ns), expressed relative to the lowest timestamp within the cluster.

Smallest/largest Absolute Timestamp The timestamps of the first and the last pixel included in the cluster (in ns), expressed relative to the internal clock of the detector.

Integral Energy Sum of energies of all pixels included in the cluster (in keV).

Volumetric Centroid A reconstructed three-dimensional point in the camera coordinate system.

Algorithm 6 Drift Speed Evaluation

Parameters: drift speed v_{drift}

1: **for all** incoming clusters C **do**
2: $T_{start} \leftarrow \min_{(X,E,T)\in C} T$
3: $C' \leftarrow \{\}$
4: **for all** $(X, E, T) \in C$ **do**
5: $(x, y) \leftarrow X$
6: $z \leftarrow Z(T \mid T_0, v_{drift})$
7: $X' \leftarrow (x, y, z)$
8: $C' \leftarrow C' \cup \{(X', E, T)\}$
9: ReportCluster(C')

4.3 Coincidence Group Matching

After three-dimensional cluster reconstructions are created, clusters undergo coincidence group matching. Just as the purpose of a clustering algorithm is to aggregate pixels in clusters, the aim of coincidence group matching is to link clusters together in so-called *coincidence groups*.

Coincidence group matching is the last step leading to Compton event detection. By their definition, Compton events can be viewed as coincidence groups comprised of two occurrences of interactions, one in a scatterer and the other in the absorber detector.

4.3.1 Thread Synchronization

Since coincidence group matching requires the inputs from all detectors, it is executed in the Compton event thread, independently of readout-related algorithms described in the previous section. This prompts the question of efficient synchronization between threads.

After the last stage of the readout thread chain, reconstructed clusters are saved in a shared queue (linked FIFO memory structure), which is owned by each readout thread. To avoid memory strain, readout threads periodically check the number of stored clusters. If a configurable threshold is exceeded, cluster

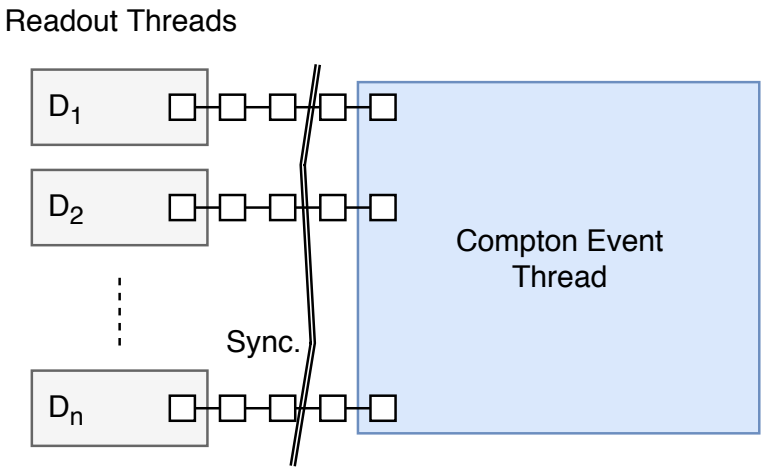


Figure 4.7: Data flow between readout threads and the Compton event thread.

processing is temporarily suspended or depending on the configuration, cluster processing continues and threads start skipping clusters.

The Compton event thread is responsible for consuming clusters from all detector queues (illustrated in Figure 4.7). Since due to parallel processing, a queue might be accessed both by a readout thread and the Compton event thread simultaneously, synchronization primitives are used to prevent undefined states. Rather than controlling access to the queue by mutual exclusion of threads, the implementation uses a lock-free approach relying on carefully placed memory barriers, which ensure that at all times the queue is accessed in a consistent state [30]. The main benefit of this method is the absence of waiting, which increases overall performance and eliminates the possibility of deadlocks.

4.3.2 Matching Algorithm

After a sufficient number of clusters is removed from individual detector queues, the Compton event thread executes the matching algorithm, which identifies correspondences between clusters from different detectors. The result is a sequence of coincidence groups, each comprised of multiple clusters. If no correspondences are found, clusters are simply reported as coincidence groups of trivial size.

In order to be mutually comparable, timestamps in all clusters are transformed prior to processing from internal detector time to synchronized camera time by linear functions described in Section 4.1.4. Due to the nature of the time synchronization functions, no additional reordering is necessary to maintain the property of time monotony.

The logic of the matching algorithm resembles that of the location-unaware clustering algorithm. The algorithm operates in cycles – at the end of each cycle, a single coincidence group is created. Since coincidence groups include at most one cluster from each detector, the matching algorithm requires that all input queues are non-empty. If that is not the case, the algorithm is temporarily suspended until more data is available.

In the course of a single cycle, synchronized cluster timestamps are compared among clusters at the end of each queue. The cluster with the lowest timestamp T_{min} is identified, and in a second pass, all clusters such that their timestamp T_i

satisfies $T_i - T_{min} < \Delta T'_{max}$ are greedily removed from their respective queues. The constant $\Delta T'_{max}$ is a configurable parameter of the matching algorithm, different from that used in the clustering stage of the readout chain. The removed clusters are inserted into a new coincidence group, which may be considered to be the result of the cycle.

Note that the definition of the algorithm (summarized in Algorithm 7) ensures that after each cycle, at least one cluster is removed. Consequently, an empty coincidence group is never produced. In practical implementation, the presented algorithm may encounter difficulties when matching multiple clusters observed by a single detector simultaneously. If such a scenario occurs, greedy approach may incorrectly aggregate irrelevant interactions depending on their ordering within the cluster queue. However, due to high[4] time resolution of Timepix3 with fast VCO enabled, such scenarios are considered sparse and thus excluded.

Algorithm 7 Coincidence Group Matching

Parameters: temporal upper bound $\Delta T'_{max}$

$Q_i \leftarrow$ cluster queues for $D_1, D_2, \ldots, D_n$
repeat
  for all $i \leq n$ **do**
    WaitForData(Q_i)
    $T_i \leftarrow$AccessMinKey(Q_i)
  $j \leftarrow \text{argmin}_i T_i$
  $T_{end} \leftarrow T_j$
  $G \leftarrow \{\}$
  $P \leftarrow \{1, 2, \ldots, n\}$
  for all $i \in P$ **do**
    if $T - T_{end} \leq \Delta T'_{max}$ **then**
      $C \leftarrow$ExtractMin(Q_i)
      $G \leftarrow G \cup \{C\}$
      $P \leftarrow P \setminus \{i\}$
      $T_{end} \leftarrow \max\{T_i, T_{end}\}$
  ReportGroup(G)
until stopped

4.3.3 Filter Cascade

To distinguish likely Compton events among coincidence groups, a filter cascade similar to that used for clusters is applied to the output of the matching algorithm. Sequentially, the cascade verifies a series of predicates for every coincidence group. If any predicate is not satisfied, the analyzed coincidence group is discarded and computational time is not spent verifying the remaining predicates.

The verified predicates are as follows:

1. The coincidence group consists of exactly 2 clusters.

[4]After applied calibrations and corrections, Timepix3 was shown to achieve time resolution up to 2 ns [18].

2. One of the clusters has been observed in the absorber detector.

3. One of the clusters has been observed in a scatterer detector.

4. The sum of cluster integral energies falls within the expected Compton event energy range for the used detection media.

5. The ratio of cluster integral energies falls within the expected range for forward scattering.

6. Cluster integral energies satisfy the necessary conditions for a valid Compton event: [19]

$$\begin{cases} \frac{m_e c^2 E_\gamma^2}{2E_\gamma + m_e c^2} \leq E'_\gamma \leq E_\gamma \\ 0 \leq E_e \leq \frac{2E_\gamma^2}{2E_\gamma + m_e c^2} \end{cases} \tag{4.2}$$

   Here, m_e is the electron mass at rest, c is the speed of light, E_e is the integral energy (in keV) observed by the scatterer detector, E'_γ is the integral energy (in keV) observed by the absorber detector, and $E_\gamma = E_e + E'_\gamma$.

The algorithm considers coincidence groups satisfying the listed predicates to be Compton events with high probability. The output of the filter cascade is therefore used for volume reconstruction described in the next chapter.

4.4 Handling Common Problems

During the operation of the camera, various malfunctions may occur in its components. The presented work includes methods to alleviate and compensate against some of the frequently encountered problems. This way, the resilience of the device for use in practical applications is increased.

4.4.1 Pixel Circuit Malfunctions

Due to its high exposure to possibly harsh radiation fields, a common source of malfunctions is presented by the detection unit, which is comprised of Timepix3 detectors. Radiation damage to their components may result in partial or complete severance of connection, or incorrect outputs of the data acquisition process. Of these, the latter is the only issue, which may not be fatal to the operation of the camera.

Upon malfunction, pixel circuits of a Timepix3 detector usually exhibit one of two behaviors:

1. The pixel becomes *dead* and does not report any changes in the analog pulse, even if the pixel actually receives energy from an incident particle.

2. The pixel becomes *noisy* and reports changes at random times (in a special case, at all times), even if the pixel actually receives no energy at all.

Due to their large number, occurence of dead pixels usually only limits the spatial sensitivity of the device and does not pose a critical threat to its operation. However, noisy pixels are responsible for producing a constant stream of false information, which is undesirable, especially during data-driven acquisitions.

In order to detect pixel malfunctions, the number of observed events (or *hits*) is counted for every pixel of each detector. The resulting counts form a matrix, which is aggregated over an extended period of time. After the set time expires, the matrix is analyzed for pixel malfunctions, reset and the aggregation process begins anew.

In the hit count matrix, pixel malfunctions may be easily identified by comparing the observed values with two thresholds h_{dead} and h_{noisy}, such that $h_{dead} < h_{noisy}$. If the hit count exceeds h_{noisy}, its corresponding pixel is labeled as noisy. If the hit count falls below h_{dead}, the pixel is labeled as dead. Otherwise, the pixel is considered to be operating properly.

Depending on the expected radiation environment, the used thresholds may be static or adaptive. While static thresholds do not require any additional computational time, they also require prior knowledge of the camera surroundings. If such information is not available, adaptive thresholds may offer a better alternative.

Assuming static unbiased radiation conditions in the vicinity of the detector, observed hit counts follow the normal distribution. By the law of large numbers, given a sufficiently large time window, the aggregated hit counts approximate the expected value of this distribution. Under the assumption that the majority of pixels is operating without malfunctions, one way to configure values of h_{dead} and h_{noisy} is by determining the median hit count $\widetilde{h}$ and setting:

$$h_{dead} = \widetilde{h} - \delta \qquad\qquad h_{noisy} = \widetilde{h} + \delta \tag{4.3}$$

Here, δ may be either a static configurable constant or adaptive outlier margin calculated from the standard deviation σ with respect to $\widetilde{h}$ as $\delta = \alpha\sigma$. In the latter case, the multiplicative constant $\alpha > 0$ determines the relative tolerance of the method in multiples of the standard deviation. Setting $\alpha > 1$ makes the method more tolerant, but also increases the probability of false negatives. Setting $\alpha < 1$ has a converse effect.

After the thresholding procedure is completed, pixel labels are used to create a *mask* – a binary matrix, where malfunctioning pixels are assigned non-zero values. The mask is used at early stages of the readout chain to filter incoming pixels in $\mathcal{O}(1)$ time. Furthermore, to prevent redundant communication, which might lead to congestion of communication channels, Katherine readouts allow to apply mask at the hardware level prior to acquisition start. For that reason, data acquisition is not performed continuously, but rather in batches, between which the hardware mask is reconfigured among other operations.

4.4.2 Temperature Monitoring

In order to prevent malfunctions due to overheating, the temperature of detectors and readouts is periodically monitored. Abrupt increases in temperature values

usually hint at undesirable events, such as malfunctions or infinite loops executed in the embedded hardware.

For monitoring detector and readout temperature, an independent monitoring thread is included in the application, periodically querying all readouts for temperature information.

Once retrieved, temperatures are compared with their previously recorded values. For safety of the equipment, the application allows to set configurable limits, which abort the operation of the device, if exceeded.

4.4.3 Readout Infrastructure Malfunctions

Since the camera consists of multiple components, which communicate over bus infrastructure, malfunctions might arise from severed or otherwise disrupted connections between detectors, readouts and the control computer.

At the level of the control computer, all communication with readouts is limited by a configurable timeout window. Depending on the activity performed during connection disruption, the expiration of the set timeout may have different effects. If the activity is not critical to device operation, the application attempts to mitigate any possible temporary disruptions by re-sending the communication. If the activity has at-most-once semantics or is critical in other sense, re-sending is not attempted. If the communication failure persists consistently, the affected readout is labeled as *unreachable* along with its corresponding detector, and both devices are excluded from device operation.

At the level of the Katherine readout, the presence of a Timepix3 detector is detected by the embedded computer. While the readout does not have the capability to proactively report communication loss to the computer, its command set offers a communication test instruction, which checks for detector presence and verifies the integrity of the communication lines upon request [28]. This command is periodically executed by the monitoring thread to detect communication disruptions. If at any point, communication is consistently disrupted, the detector is labeled as unreachable and excluded from device operation.

Depending on the hardware configuration, occurence of unreachable detectors may or may not be fatal to the operation of the camera. If at least one scatterer and the absorber detector are still reachable, the device remains operational.

5. Volume Reconstruction

Having obtained a sufficient quantity of Compton events, the process of volume reconstruction allows to produce a three-dimensional image of the immediate surroundings of the detection unit. The resulting image depicts the environment in the γ-ray spectrum, hinting at the relative location of the emission source. With this motivation, the topic of this chapter is the process of generating such an image.

5.1 Operational Principles

As shown earlier in Section 2.1.5, a correspondence exists between Compton events and spatial cones. Each Compton event can thus be assigned a cone, which is determined by the points of interaction and the amounts of deposited energy. By the derivation of the cone, the unknown emission source M is located in its mantle. To efficiently localize M, multiple cones are required.

5.1.1 Volume Restriction

While the domain of M may be considered to be $\mathbb{R}^3$, this definition is not actually used in practice for the reasons of tractability. Instead, the domain is artificially limited to be a compact set V called *the volume*, which can later be easily discretized. One example of such a set is *axis-parallel cuboid* – a convex polyhedron, which may be defined in three dimensions as a Cartesian product of intervals $[l_1, h_1] \times [l_2, h_2] \times [l_3, h_3]$, where $l_i < h_i$ are finite numbers for $i \leq 3$. Any $x \in V$ such that $x = (x_1, x_2, x_3)$ then satisfies the constraints: $l_i \leq x_i \leq h_i$ for all $i \leq 3$.

It is important to note that the process of domain restriction is not without its dangers. If the constraints are chosen to be excessively strict, the resulting volume V may not contain M at all, leading to ill-conditioned problem definition. Fortunately, this is not the case in practice, where measurements are burdened by noise and detection range of localization hardware is limited by the laws of physics. For this reason, sufficiently large constant values of l_i and h_i always exist that allow $M \in V$. Usually, the values are chosen such that $h_1 - l_1 = h_2 - l_2 = h_3 - l_3$, making V an axis-parallel cube of side $d = h_1 - l_1$.

Restriction of the solution space presents an opportunity for the user to influence the behavior of the reconstruction process by inputting knowledge about the emission source known *a priori*. If used conservatively, this allows to improve performance or precision of the reconstruction without impacting correctness of the algorithm.

5.1.2 Volume Discretization

Before reconstruction, the volume V is discretized into a finite number of identical *cells*. The cells are arranged in an orthogonal, axis-parallel grid such that no cell pair overlaps, and the union of all cells is the original volume V. The set of all cells may thus be called *a discrete partitioning* of V.

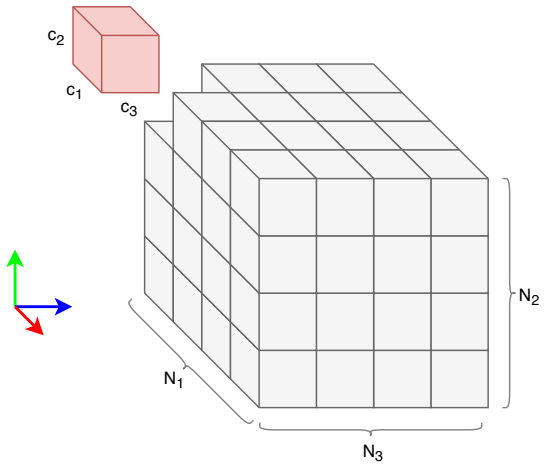


Figure 5.1: The cell grid, which is a result of the discretization process. Note that while c_i denotes cell dimensions, N_i labels cell count for $i \leq 3$.

In the presented work, the shape of axis-parallel cuboid has been selected for the solution space with explicit consideration for the complexity of the discretization process. Discretization of cuboids is simple to implement and efficient to address, as it can be obtained by cutting the cuboid by axis-orthogonal planes at regular intervals. Consequently, if V is an axis-parallel cuboid, the produced cells are also axis-parallel cuboids.

In order to regularly divide V into identical cells, three integer parameters N_1, N_2 and N_3 must be provided such that $1 \leq N_i$ for all $i \leq 3$. The values of these parameters represent requested cell counts in directions along the principal axes. The cell size c_i in the direction of the i-th axis is then given by inverse relationship $c_i = (h_i - l_i)/N_i$ for $i \leq 3$. If x_1, x_2 and x_3 are integer *coordinates* of a cell such that $0 \leq x_i \leq N_i - 1$ for $i \leq 3$, the dimensions of its corresponding cuboid are defined as follows.

$$\mathcal{U}(x_1, x_2, x_3) = [c_1 x_1, c_1(x_1 + 1)] \times [c_2 x_2, c_2(x_2 + 1)] \times [c_3 x_3, c_3(x_3 + 1)] \quad (5.1)$$

The relationship between the cell size c_i and the number of cells N_i gives the user means of leveraging precision with complexity, which is characteristic to discretized models. While the choice of large N_i will produce more cells, leading to longer processing times, the choice of small N_i will inevitably limit the spatial resolution of the localized emission source.

In the presented implementation, it holds that $N_1 = N_2 = N_3 = N$. Since also $h_i - l_i = d$ for all $i \leq 3$, this implies that $c_1 = c_2 = c_3 = c$. Consequently, all cells are cubes of side $c = d/N$.

5.1.3 Forward Projection

The first part of the volume reconstruction process is called *forward projection* [31, 32]. As the name suggests, the purpose of the procedure is to project a single Compton cone forward into an arbitrary axis-orthogonal plane within the discretized volume V.

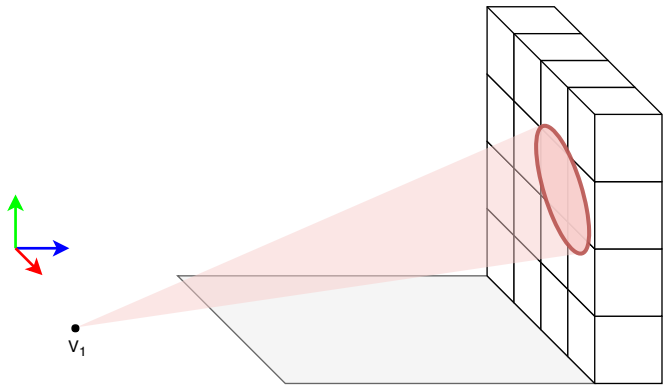


Figure 5.2: Forward projection of a Compton cone $\mathcal{C}(V_1, V_2, \beta)$.

In the input, the cone is parameterized by points V_1 and V_2, and half-angle β. The plane ρ is parallel to the XY-plane and thus parameterized by a single constant coordinate z of all its points. Usually, z is chosen to be the mean Z-coordinate of the cells determined by the set $\mathcal{S}_3(N_3 - 1)$, which is generally defined for arbitrary integer $i \leq 3$ as follows.

$$\mathcal{S}_i(y) = \{(x_1, x_2, x_3) \mid x_i = y \wedge (\forall j \leq 3 : 0 \leq x_j \leq N_j - 1)\} \tag{5.2}$$

Semantically, the set $\mathcal{S}_i(y)$ represents *a slice* in the cell grid coordinate system at the coordinate y of the i-th axis. Formally, the set function $\mathcal{S}_i$ is well-defined for any integer y such that $0 \leq y \leq N_i - 1$, and integer $i \leq 3$.

The output of forward projection is a mapping $f : \mathcal{S}_3(N_3 - 1) \to \mathbb{R}$, which assigns real values to the cells of the selected slice depending on their location relative to the cone mantle. The interpretation of these values is determined by a so-called *response function* $r : \mathbb{R} \to \mathbb{R}$, which is a configurable parameter of the method. Composed, the mapping $r \circ f$ yields likelihood that a given cell contains the source of γ-ray emission.

5.1.4 Back Projection

If viewed as a partial image function, the mapping $r \circ f$ produced by the forward projection method may be considered to be the first part of the reconstructed three-dimensional image I. To obtain its remaining parts, *the back projection* method is utilized [31, 32].

In a naïve view, back projection may be deemed unnecessary, as forward projection can simply be executed multiple times for various axis-orthogonal planes ρ at arbitrary depths z. In a given discretized volume V, the remaining parts of I would then be obtained by evaluating forward projection for the slices $\mathcal{S}_3(N_3 - 2), \mathcal{S}_3(N_3 - 3), \ldots, \mathcal{S}_3(1), \mathcal{S}_3(0)$. While this approach is technically correct, in practice it is considered excessively wasteful since the calculation of forward projection is a computationally intensive operation. This motivates the use of back projection, which attempts to minimize processing time by re-using previously obtained information. With decreased processing time, a finer discretization may be performed, producing outputs of better accuracy.

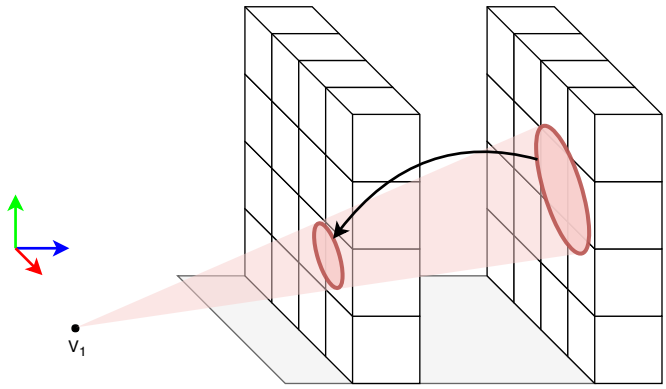


Figure 5.3: Back projection of a Compton cone $\mathcal{C}(V_1, V_2, \beta)$.

To hint at the basic principle used by back projection, a two-dimensional analogy may first be examined. When projecting a point along a set transversal direction onto multiple parallel lines, it is sufficient to calculate only one of its projections. Due to the mutual relationship between the parallel lines, the remaining projections may be extrapolated from the original point, the first projection and known distances between the lines. In general, back projection performs analogous extrapolation in three-dimensional space.

In the input, the algorithm receives a cone parameterized by points V_1 and V_2, and half-angle β. Moreover, the algorithm also requires coordinates z and z' of the axis-orthogonal planes used for forward projection and back projection, respectively. Lastly, the algorithm relies on the mapping f, which is the result of forward projection of $\mathcal{C}(V_1, V_2, \beta)$ onto the plane ρ at depth z.

The back projection algorithm uses the relationship between V_1, z and z' to construct a transformation T, which defines the projection f' of $\mathcal{C}(V_1, V_2, \beta)$ onto the plane ρ' at depth z' as $f'(x) = f(T(x))$. Since values of $T(x)$ are not guaranteed to be integral, an interpolation method is employed to estimate values of f in undefined points.

Similarly to forward projection, the output of back projection is a mapping f', which assigns cells at depth z' real values. Consequently, $r \circ f'$ represents another component of the reconstructed three-dimensional image I.

5.1.5 Aggregation

Thus far, the volume reconstruction process has been described only for the instance of a single Compton cone C_j. To summarize, the cone first undergoes forward projection, obtaining the projection mapping $f_{N-1}^{C_j}$ for the farthest slice. After that, multiple back projections are performed, where $f_{N-1}^{C_j}$ is used to produce mappings $f_{N-2}^{C_j}, f_{N-3}^{C_j}, \ldots, f_1^{C_j}, f_0^{C_j}$. Eventually, a three-dimensional image I^{C_j} of the cone is created by composition with response function r as $I^{C_j} = r \circ \bigcup_{i=0}^{N-1} f_i^{C_j}$

Given a set of cones $\{C_1, C_2, \ldots, C_n\}$, this process is repeated for each cone, obtaining a set of corresponding images $\{I^{C_1}, I^{C_2}, \ldots, I^{C_n}\}$. The images then are combined into a single image I as $I = \sum_{j=1}^{n} I^{C_j}$. Provided that a sufficient number of distinct cones is available and all cones correspond with the same emission source, I reflects the probability that a given cell contains the source.

5.2 Compton Event Processing

By the taxonomy used in the previous chapter, the process of volume reconstruction may be viewed as a chain of multiple producer/consumer components, wherein the first component is a consumer of Compton events, and the last component is a producer of reconstructed images.

5.2.1 Spectroscopic Source Discrimination

The purpose of the first stage of the chain is to discriminate incoming Compton events by their source of emission, ensuring that subsequent stages operate only on events originating from a single source. If the presence of at most one source is guaranteed by *a priori* knowledge, this stage may be omitted from processing in order to increase performance.

In the presented work, source discrimination is performed by comparing integral energies of both interactions with a baseline established by prior measurements. If the energies significantly deviate from the baseline, their corresponding event is rejected and excluded from processing. The remainder of events is passed on to the next stage. The baseline and tolerance interval width are static configurable parameters of the algorithm.

Algorithm 8 Spectroscopic Source Discrimination

Parameters: baseline energies $\widetilde{E}_s$, $\widetilde{E}_a$, tolerance ΔE_{max}

1: **for all** incoming events (T, C_s, C_a) **do**
2: $(T_s, P_s, E_s) \leftarrow C_s$
3: $(T_a, P_a, E_a) \leftarrow C_a$
4: **if** $|E_s - \widetilde{E}_s| \leq \Delta E_{max}$ and $|E_a - \widetilde{E}_a| \leq \Delta E_{max}$ **then**
5: ReportEvent(T, C_s, C_a) ▷ Event is consistent with the baseline

5.2.2 Batch Aggregation

Since multiple Compton events are required by the reconstruction process, the purpose of the second stage is to aggregate a sufficient number of events in so-called *batches*.

The aggregation process is sequential, grouping events in the order of appearance. Once a sufficient number of events is reached, the batch is passed on to the next stage. The size of a batch is a configurable parameter, which is constrained by the available memory of the control computer and the requested frequency of reconstruction. Usually, the batch size is chosen to be in the order of hundreds or thousands of events.

To avoid possible smearing due to source or camera manipulation over an extended period of time, the size of the interval determined by the lowest and the largest timestamp within the batch is tracked. If the interval size exceeds a set threshold, Compton events are not detected at a sufficient rate, and the entire batch is thus rejected. Analogous to the batch size, the maximum interval size is a configurable parameter of the algorithm, chosen with consideration for the source expected decay rate.

Algorithm 9 Batch Aggregation

Parameters: batch size N_b, maximum time interval size T_{max}

$B \leftarrow \{\}$, $T_{start} \leftarrow \infty$, $T_{end} \leftarrow -\infty$
for all incoming events (T, C_s, C_a) **do**
    $B \leftarrow B \cup \{(T, C_s, C_a)\}$
    $T_{start} \leftarrow \min\{T, T_{start}\}$
    $T_{end} \leftarrow \max\{T, T_{end}\}$
    if $|B| = N_b$ **then**
        ReportBatch(B) ▷ Pass batch on to the next stage
        $B \leftarrow \{\}$, $T_{start} \leftarrow \infty$, $T_{end} \leftarrow -\infty$
    else if $T_{end} - T_{start} > T_{max}$ **then**
        $B \leftarrow \{\}$, $T_{start} \leftarrow \infty$, $T_{end} \leftarrow -\infty$ ▷ Reject batch due to *timeout*

5.2.3 Response Function Caching

During volume reconstruction, the constructed image mapping f is composed with a response function r, yielding a three-dimensional image I. While the values of the mapping f are determined by the orientation of the projected cone, the arbitrary function r is considered to be a parameter of the algorithm, and can thus be modified depending on the expected application of the method. Since $r \circ f$ is evaluated for every cell of the discretized volume V and usually comprised of non-linear elements, which are computationally expensive to calculate, its evaluation time is thus subject to optimization.

By introducing the process of back projection, the number of cells in which $r \circ f$ is evaluated is significantly limited. Consequently, the function is only evaluated for the cells of the last slice $\mathcal{S}_3(N_3 - 1)$ of V. To further minimize the calculation time of a single cell, values of $r \circ f$ are cached in a dedicated data structure analogous to a look-up table.

In the implementation, values of $r \circ f$ are not calculated when needed but rather only once, in advance. Since the points of evaluation are not known prior to processing start, $r \circ f$ is evaluated in multiple discrete points obtained by uniform division of a multi-dimensional interval. The results of this evaluation are cached. Later during processing, when $(r \circ f)(X)$ would conventionally be evaluated, the cached points closest to X are identified and used in inference of $(r \circ f)(X)$ by an interpolation method. If the look-up and interpolation is performed faster than direct evaluation of $(r \circ f)(X)$, this approach presents a viable improvement in processing time, especially provided that $r \circ f$ is evaluated frequently.

It should be noted that usage of interpolation inherently introduces inaccuracies into further calculations. The magnitude of such inaccuracies however depends on the number of points sampled in the pre-calculation stage. With a greater number of samples, interpolation interval size decreases along with the interpolation error. This work therefore assumes that the available memory of the control computer allows to calculate a sufficient number of samples.

The description thus far is identical to that of conventional look-up tables. The presented approach however differs in its dimensionality and the use of geometric information in calculation of the sampled points. While look-up tables in the

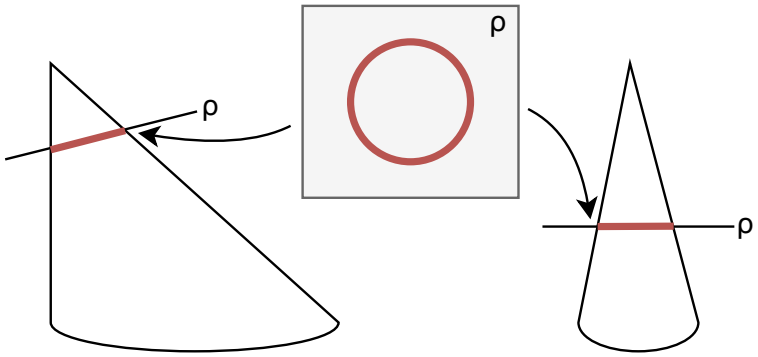


Figure 5.4: Different locations and orientations of ρ dependent on the half-angle of the intersected cone.

conventional sense are usually one-dimensional, the domain in which $r \circ f$ is calculated is rather a discretized two-dimensional plane ρ analogous to a slice of the reconstructed volume V.

By the definition of f and r, the values of $r \circ f$ cached within points of ρ may be viewed as indicators[1] of intersection between ρ and a Compton cone. In this context, the conic section may either be an ellipse or a circle depending on the location and orientation of ρ relative to the intersected cone. For the reasons of consistency, additional condition is thus imposed, ensuring that both bodies are positioned and oriented so that the section is a perfect unit circle. As shown below, this leads to unique determination of ρ by the parameters of the cone up to rotational symmetry. Consequently, the distance of ρ from the cone apex varies depending on the half-angle of the cone, as depicted in Figure 5.4.

Let $\mathcal{C}(V_1, V_2, \beta)$ be a general Compton cone. In order to achieve a section in the form of a perfect circle, the plane ρ must be orthogonal to the principal axis of the cone. The normal vector of ρ is therefore $\overrightarrow{V_2V_1}$. This fully defines the orientation of ρ, leaving only one remaining degree of freedom representing the distance d between ρ and the apex V_1 of the cone.

To calculate d, it is sufficient to enforce the unit property of the intersected circle. By a simple geometric observation (illustrated in Figure 5.5), the radius s of the circle is related to d by the expression:

$$\tan \beta = \frac{s}{d} \tag{5.3}$$

Since β is a constant parameter of the cone, and $s = 1$ by the unit property, d is derived as:

$$d = \frac{1}{\tan \beta} \tag{5.4}$$

In this expression, note that the non-triviality condition for the cone arises from $\tan \beta > 0$.

[1] Unlike indicators in the conventional sense, cached values of ρ are not binary and their interpretation depends on the choice of r.

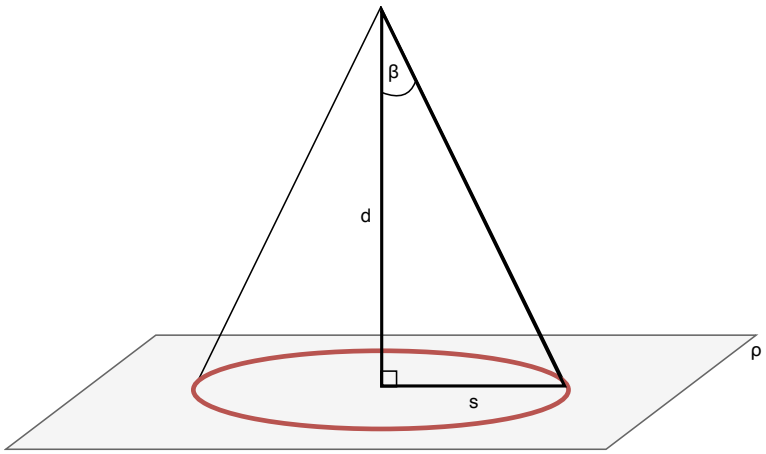


Figure 5.5: An orthogonal triangle within a Compton cone, which shows the relationship between β, s and d described by Expression 5.3.

This concludes the calculation of the parameters of ρ from the parameters of $\mathcal{C}(V_1, V_2, \beta)$. While the location and orientation of ρ changes depending on the cone, the values of $r \circ f$ contained in its cached points remain the same, if expressed relative to the unit circle within the plane.

Returning to the look-up table analogy, this derivation allows to evaluate $r \circ f$ without any prior knowledge of the cone or the plane, on which the cone is projected. Instead, $r \circ f$ is evaluated only once for the points of the generic plane ρ. Later during forward projection of non-trivial cone onto a general Z-orthogonal plane σ, the parameters of ρ are found by the presented calculation, and a projection is constructed between σ and ρ. The value of $(r \circ f)(X)$ for a point $X \in \sigma$ is then obtained by interpolation of the points in ρ close to the projection of X.

5.2.4 Batch Projection

To facilitate projection of a cone batch by the approach described in the previous section, a specialized three-stage algorithm is used. While the first stage is only performed once during initialization, the remaining two stages are repeated for each cone of the batch.

The purpose of the first stage is to allocate an empty volume V, to which all cones of the batch will eventually contribute. At that time, constant values of $r \circ f$ are also pre-calculated. The second and the third stage of the algorithm correspond to forward and back projection of a particular cone. In the second stage, the projection mapping between ρ and σ is constructed and the cells of the last slice $\mathcal{S}_3(N_3 - 1)$ of V are filled with interpolated values. In the third stage, the cells of the remaining slices $\mathcal{S}_3(N_3 - 2), \mathcal{S}_3(N_3 - 3), \ldots, \mathcal{S}_3(1), \mathcal{S}_3(0)$ are filled from $\mathcal{S}_3(N_3 - 1)$. This approach is summarized in Algorithm 10.

5.2.5 Forward Mapping Construction

In Algorithm 10, the second stage performs forward projection in three steps:

1. The parameters θ_ρ of the plane ρ are calculated from the projected cone.

Algorithm 10 Batch Projection

Parameters: cone batch B, response function r

1: $V \leftarrow$ empty volume with slices $\mathcal{S}_i(y)$ ▷ Stage 1
2: $\rho \leftarrow$ PrecalculateResponse(r)
3: **for all** $(V_1, V_2, \beta) \in B$ **do** ▷ Stage 2
4: $\quad\theta_\rho \leftarrow$ FindPlaneParameters(V_1, V_2, β)
5: $\quad P_\sigma \leftarrow$ ForwardMapping(θ_ρ, V_1, V_2)
6: $\quad$Interpolate$(V, \mathcal{S}_3(N_3 - 1), \rho, P_\sigma)$
7: $\quad$**for all** $0 \leq y \leq N_3 - 2$ **do** ▷ Stage 3
8: $\quad\quad P' \leftarrow$ BackMapping(y)
9: $\quad\quad$Interpolate$(V, \mathcal{S}_3(y), \mathcal{S}_3(N_3 - 1), P')$
10: ReportVolume(V)

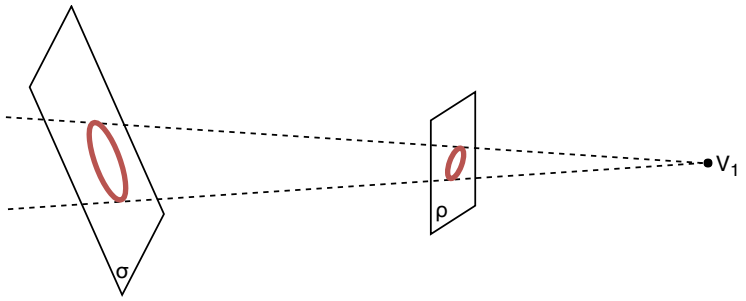


Figure 5.6: Perspective transformation used for forward projection.

2. The *forward mapping* P_σ is constructed between ρ and the target plane σ.

3. Points of ρ are projected onto σ by means of reverse interpolation using P_σ.

Given a Compton cone $\mathcal{C}(V_1, V_2, \beta)$, parameters θ_ρ are derived by the geometric calculation presented in Section 5.2.3. To simplify the calculations performed in the second step, an alternate *cone coordinate system* is introduced, wherein the origin is placed in V_1 and the principal axes are rotated so that $\overrightarrow{V_2V_1}$ defines the positive direction of the Z-axis. Transitioning between this coordinate system and the camera coordinate system thus only requires application of translation and rotation. Henceforth, for the clarity of distinction all bodies in the cone coordinate system are denoted using prime notation (e.g. X'), whereas bodies in the camera coordinate system use regular notation (e.g. X).

The purpose of the cone coordinate system is to allow the construction of a perspective projection (shown in Figure 5.6) defined by the viewpoint V_1' and the projection plane ρ'. Note that in this interpretation, the parameter d of θ_ρ denotes the focal length of the projection.

The unknown projection mapping may be viewed as a matrix P, which maps arbitrary points in the cone coordinate system onto ρ'. For practical purposes, it is advantageous that P is derived in so-called *homogeneous* coordinates, which introduce an additional fourth dimension into all coordinates. In the subsequent calculations, point coordinates are thus normalized so that this extraneous dimension is equal to one, removing ambiguity due to the fact that multiple ho-

mogeneous points may represent the same[2] point in space. If a normalized point in space $X' = (x', y', z', 1)$ is projected[3] onto ρ' as $X'_\rho = (x'_\rho, y'_\rho, z'_\rho, z'_\rho/d)$, the projection matrix P satisfies:

$$X'_\rho = P \cdot X' \tag{5.5}$$

$$\begin{bmatrix} x'_\rho \\ y'_\rho \\ z'_\rho \\ z'_\rho/d \end{bmatrix} = P \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \tag{5.6}$$

One matrix P which is compliant with this constraint is given by:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \tag{5.7}$$

Finally, if P_R and P_T represent standard rotation and translation transformation matrices required to transition from the camera coordinate system to the cone coordinate system, the projection of a point X expressed in homogeneous coordinates of the camera coordinate system may be calculated as:

$$X'_\rho = \underbrace{PP_RP_T}_{P_\rho} \cdot X \tag{5.8}$$

Here, the matrix P_ρ denotes the product of the three previously specified matrices, which corresponds with the sought mapping. Furthermore, note that the computation of P_ρ is not dependent on X, allowing the implementation to calculate the matrix only once prior to projecting all points of the given cone.

At this point, a curious reader might have noticed that while the aim of the forward projection process is to project points of ρ onto σ, the derivation described thus far allows the exact opposite – this is no coincidence. To minimize interpolation errors, the projection process is implemented *in reverse*. One way of understanding this optimization is to observe that in order to calculate a value of a single point in σ, the implementation can reverse-project its location into ρ to identify nearby points in ρ which contribute to the interpolation of its value.

5.2.6 Back Mapping Construction

In comparison with forward projection, the process of back projection is more simple and computationally less intensive. Its goal is to project the values of cells of the last slice $\mathcal{S}_3(N_3 - 1)$ onto an arbitrary slice $\mathcal{S}_3(y)$ of the volume V determined by $y \in \mathbb{N}_0$ such that $0 \leq y \leq N_3 - 2$. Consistent with the notation of the previous section, the cells of $\mathcal{S}_3(N_3-1)$ correspond with points in Z-orthogonal

[2] $(x, y, z, 1)$ is *the normalized representative* of equivalence class $\{(kx, ky, kz, k) \mid k \in \mathbb{R}\setminus\{0\}\}$.

[3] Note that X'_ρ normalizes to $(x'_\rho/z'_\rho, y'_\rho/z'_\rho, d, 1)$ and therefore lies in ρ'.

plane σ. Similarly, the cells of $\mathcal{S}_3(y)$ correspond with points in plane τ, which is parallel to σ.

In the implementation, back projection is repeatedly executed by the third stage of Algorithm 10. For each target slice, two steps are performed:

1. The *back mapping* P_σ is constructed between σ and the target plane τ.
2. Points of σ are projected onto τ by means of reverse interpolation using P_σ.

The process of calculation of P_σ mimics that of P_ρ described in the previous section. First, an alternate coordinate system is constructed (here denoted by double prime notation, e.g. X'') by placing the cone apex V_1 in the origin. For this purpose, the previously calculated matrix P_T is re-used. Since both planes τ and σ already are Z-orthogonal, no further rotation is required in order to begin the construction process.

By the Z-orthogonality of σ, the shortest distance d_σ of σ from the origin is obtained as the Z-coordinate of any of its points. In this context, the distance d_σ may be interpreted as the focal length of the constructed perspective. To derive P_σ, two points in homogeneous coordinates of the alternate system are considered.

Let $X'' = (x'', y'', z'', 1)$ be a normalized point of τ'', which is projected onto σ'' as $X''_\sigma = (x''_\sigma, y''_\sigma, z''_\sigma, z''_\sigma/d_\sigma)$. The projection matrix Q satisfies:

$$X''_\sigma = Q \cdot X'' \tag{5.9}$$

$$\begin{bmatrix} x''_\sigma \\ y''_\sigma \\ z''_\sigma \\ z''_\sigma/d_\sigma \end{bmatrix} = Q \cdot \begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix} \tag{5.10}$$

Matrix Q is then analogous to P. It is defined as follows:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d_\sigma & 0 \end{bmatrix} \tag{5.11}$$

The general projection matrix P_σ is obtained by transitioning from the camera coordinate system into the alternate system, projecting by Q and transitioning back. With this motivation, P_σ is given by:

$$X''_\sigma = QP_T \cdot X \tag{5.12}$$

$$X_\sigma = \underbrace{P_T^{-1} Q P_T}_{P_\sigma} \cdot X \tag{5.13}$$

Here, note that it is not necessary to calculate inverse matrix of P_T in order to obtain P_T^{-1}. Since P_T is a standard translation matrix, it corresponds to a vector $\vec{w}$. By the invertibility of translation, the matrix P_T^{-1} is also a translation matrix which corresponds to vector $-\vec{w}$.

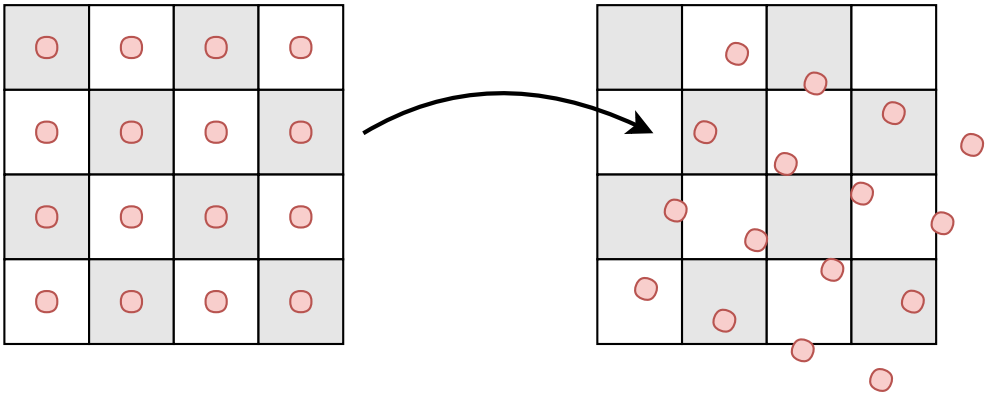


Figure 5.7: Coordinate transformation, which motivates the use of interpolation.

This concludes the derivation of the projection matrix P_σ. Similarly to forward projection, the value of each cell of τ is calculated by projecting its corresponding point X onto σ. The coordinates of X_σ are then used by the interpolation method to calculate the value of the cell.

5.3 Interpolation Methods

In each of the presented projection procedures, multiple coplanar points are projected onto a target plane. Much like in conventional two-dimensional images, the input points are arranged in a uniform two-dimensional lattice and addressed by their grid coordinates. In the target plane, the point projections are organized in the same structure (e.g. cells of a volume slice).

After a projection mapping is constructed, the projection procedure is reduced to the calculation of values corresponding to points of the target plane. For each such point, the projection mapping supplies coordinates in the source plane, which are projected onto its location. If these coordinates are integers, the specified point in the target plane can be directly matched with *a template point* in the source plane, which has the same value. Unfortunately, in reality that is often not the case.

If it is not possible to find a template point, the coordinates supplied by the projection mapping are either decimal or out of bounds (shown in Figure 5.7). In the latter case, a constant zero value is chosen for the target point by the implementation. In the former case, *an interpolation method* is used to calculate the value of the target point.

In the context of projection, the task of an interpolation is formalized as follows. Let $T \subseteq \mathbb{R}^2$ be a finite set of template points in the source plane and $\nu : T \to \mathbb{R}$ be a mapping which assigns values to template points. Let $X \in \mathbb{R}^2$ be a point in the source plane supplied by the projection mapping. Assuming non-trivial conditions (X is within bounds and has at least one decimal coordinate), the goal of interpolation method is to find the value assigned to X based on its location and known values assigned to points in T.

The presented work contains two interpolation methods further described in this section.

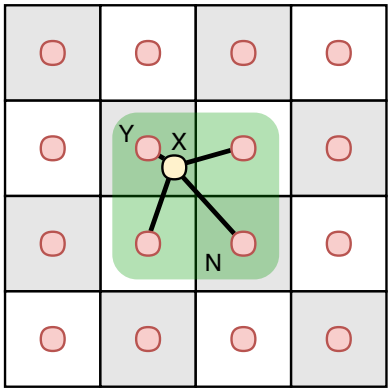


Figure 5.8: Nearest neighbor interpolation.

5.3.1 Nearest Neighbor Interpolation

Nearest neighbor interpolation is a simple method, which infers the value of X from a single point of $Y \in T$, as if the coordinates of X were integral. However, since Y is not fully determined by X, the method chooses a surrogate point instead based on the distance from X. As the name of the method suggests, the point Y is thus selected to be *the nearest neighbor* of X, which minimizes the distance between X and Y.

Since T is a two-dimensional lattice, the search for the nearest neighbor can be optimized to be performed in $\mathcal{O}(1)$ time. If $X = (x, y)$, T may be constrained to a set T_C of candidate points as follows:

$$T_C = T \cap N \quad \text{where} \tag{5.14}$$

$$N = \{(\lfloor x \rfloor, \lfloor y \rfloor), (\lfloor x \rfloor, \lceil y \rceil), (\lceil x \rceil, \lfloor y \rfloor), (\lceil x \rceil, \lceil y \rceil)\} \tag{5.15}$$

Since $|T_C| \leq 4$, at most 4 distance comparisons are required in order to identify the nearest neighbor $Y \in T_C$. This is illustrated in Figure 5.8.

5.3.2 Bilinear Interpolation

An example of a more complex interpolation method is presented by bilinear interpolation [33]. Unlike nearest neighbor interpolation, this method does not infer the value of X directly from a single point in T. Instead, the method combines values of nearby points by means of conventional linear interpolation.

The points used to estimate the value of $X = (x, y)$ are the points of the set N defined in the previous section. For the sake of clarity, these points may be labeled $\{Y_{TL}, Y_{BL}, Y_{TR}, Y_{BR}\}$, where the letters in subscript abbreviate *top*, *left*, *bottom* and *right*.

Linear interpolation is first performed in the direction of the X-axis. This results in two values, which may be attributed to imaginary points $Y_T = (x, \lfloor y \rfloor)$ and $Y_B = (x, \lceil y \rceil)$ (shown in Figure 5.9). Their values are given by:

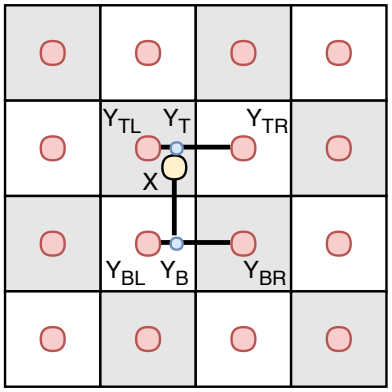


Figure 5.9: Bilinear interpolation.

$$\nu(Y_T) = \frac{\lceil x \rceil - x}{\lceil x \rceil - \lfloor x \rfloor} \cdot \nu(Y_{TL}) + \frac{x - \lfloor x \rfloor}{\lceil x \rceil - \lfloor x \rfloor} \cdot \nu(Y_{TR}) \tag{5.16}$$

$$\nu(Y_B) = \frac{\lceil x \rceil - x}{\lceil x \rceil - \lfloor x \rfloor} \cdot \nu(Y_{BL}) + \frac{x - \lfloor x \rfloor}{\lceil x \rceil - \lfloor x \rfloor} \cdot \nu(Y_{BR}) \tag{5.17}$$

Next, linear interpolation is performed in the direction of the Y-axis between points Y_T and Y_B. This yields the value for X as follows:

$$\nu(X) = \frac{\lceil y \rceil - y}{\lceil y \rceil - \lfloor y \rfloor} \cdot \nu(Y_B) + \frac{y - \lfloor y \rfloor}{\lceil y \rceil - \lfloor y \rfloor} \cdot \nu(Y_T) \tag{5.18}$$

$$= (\lceil x \rceil - \lfloor x \rfloor)^{-1} (\lceil y \rceil - \lfloor y \rfloor)^{-1} \tag{5.19}$$

$$\cdot \begin{bmatrix} \lceil x \rceil - x & x - \lfloor x \rfloor \end{bmatrix} \begin{bmatrix} \nu(Y_{BL}) & \nu(Y_{TL}) \\ \nu(Y_{BR}) & \nu(Y_{TR}) \end{bmatrix} \begin{bmatrix} \lceil y \rceil - y \\ y - \lfloor y \rfloor \end{bmatrix} \tag{5.20}$$

Here, the interpolated value of X is given explicitly and can thus be obtained in $\mathcal{O}(1)$ time, similarly to the nearest neighbor interpolation.

5.4 Implementation Notes

In order to improve performance and precision of the volume reconstruction process, the presented implementation includes various minor optimizations. This section describes the most notable changes.

5.4.1 Subsampling

The precision of the volume reconstruction process is in part determined by the density of discretization of the reconstructed volume V. While this density is specified as cell counts N_1, N_2 and N_3, the numbers of lattice points used in projection source planes ρ and σ are usually chosen differently.

To minimize interpolation error, both lattices are structured to contain a greater number of points than the dimension of the volume. This results in subsampling during the projection process, leading to more accurate reconstruction.

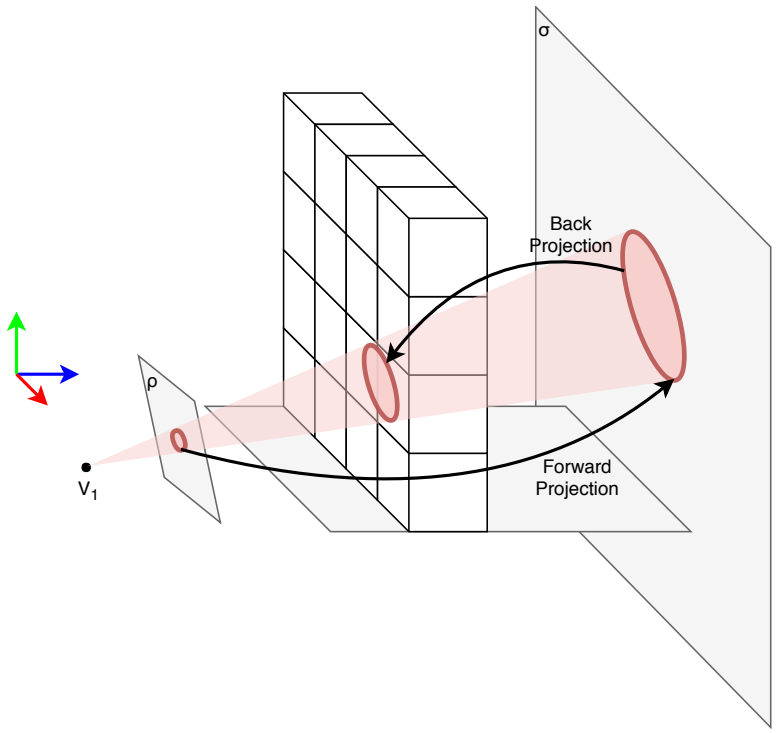


Figure 5.10: Reconstructed volume V and projection planes ρ and σ.

Consequently, due to the change in lattice point count, the plane σ is not considered to be a part of the volume. Instead, the plane is positioned just behind the last slice $\mathcal{S}_3(N_3 - 1)$ (as shown in Figure 5.10), which is calculated as a result of back projection along with the remaining slices. Among others, this change allows to extend the range of points in σ to a larger neighborhood, avoiding any possible problems due to cropped cone projections.

5.4.2 Parallelization

The presented series of algorithms is compatible with optimization by means of data parallelization. Since the presented implementation already uses multi-threaded computing architecture for simultaneous processing of detector outputs, this optimization is not integrated. However, for use in certain applications, its basic notion is described by this section.

To reconstruct a cone batch using parallelized computation, the same algorithm may be used. However, the volume V is allocated privately for each processing unit. Cones are divided uniformly among units, and each unit performs forward and back projections for its assigned cones sequentially. The only major difference is that the projected cones are saved into the local copy of V owned by the processing unit to avoid writing into shared memory. At the end of the calculation, local copies of V are aggregated into one volume by means of additive reduce operation.

If wide vector registers are available in the processing units, multiplicative speedup may be achieved by re-ordering stages 2 and 3 of Algorithm 10. If the vector register width is k and a sufficient amount of memory is available, the unit may then process k cones simultaneously.

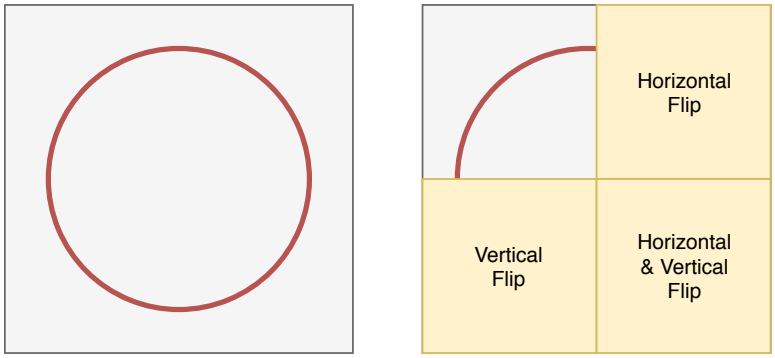


Figure 5.11: Point symmetry used to reduce memory requirements.

5.4.3 Symmetry Compression

In stage 1 of Algorithm 10, the values of the mapping $r \circ f$ are pre-calculated for a lattice of points in ρ. By the definition of ρ, the values of f exhibit point symmetry with respect to the intersection point of ρ and the cone axis $\overrightarrow{V_2V_1}$ placed in V_1. This relationship is utilized to decrease memory requirements of the interpolation look-up table.

The plane ρ is divided into four quadrants, split by the point of symmetry. Of these four quadrants, it is necessary to calculate and store values of points in only one quadrant. By the symmetry, the points of the remaining quadrants may be matched with those of the first quadrant by horizontal and vertical flips. This is indicated in Figure 5.11.

6. Evaluation

To show the viability of the implementation, selected parts of the work are presented in this chapter. Due to lack of a sufficient amount of ground truth data, it is not suggested that these results lead to qualitative comparison with other state-of-art technologies. Instead, the motivation is to simply demonstrate the operation of the system under simulated conditions.

6.1 Performance Experiments

In the first set of experiments, the performance of various parts of the system is evaluated. The purpose of this evaluation is to investigate the performance properties of the components responsible for sequential processing of Timepix3 detector data with consideration for possibly high volume of information produced.

6.1.1 Rate Estimation

In order to relate the results of this experiment with the parameters of the hardware components, it is first necessary to estimate the rate of information generated by the sensing equipment.

The rate of measured information produced by pixel detectors is usually expressed in hits (activated pixels) per second. Naturally, this quantity is only given as a theoretical upper bound determined by the limitations of readout electronics. The actual observed hit rate heavily depends on the radiation environment around the detector at the time of measurement. However, the upper bound guarantees that hit rate increases along with particle flux only up to some point, at which the detector may be considered fully saturated.

In Timepix3, the upper bound given by authors is $40 \cdot 10^6$ hits per second for a square centimeter of sensor material [1]. The area of a conventional 256×256 detector with 55 μm pitch is $(256 \cdot 55\ \mu\text{m})^2 \approx 1.98\ \text{cm}^2$. This amounts to approximately $79.3 \cdot 10^6$ hits per second.

When controlling a Timepix3 detector by a Katherine readout, the maximum hit rate is additionally limited by the throughput of the Ethernet bus, which is used to communicate the measured data, and the used data encoding [28]. According to its authors, the largest hit rate is thereby constrained to $16 \cdot 10^6$ hits per second [25]. For the purposes of this work, this rate may thus be viewed as the worst-case scenario.

However, as previously mentioned, the maximum hit rate merely describes the capability of the detector and the readout to convey the measured information to the operator. In practical applications, the actual observed hit rate may be considerably smaller depending on the particle flux of the measured environment. This motivates the definition of another quantity used to describe the rate of information – the number of events (observed clusters) per second. In this context, an event is regarded to be an interaction of a single incident charged particle with the sensor material. Trivially, events may thus consist of multiple hits, depending

on the particle species and trajectory, and the event rate is always bounded from the top by the hit rate.

In SPECT applications, the event rate varies based on the type of the used radioisotope. Assuming that only a fraction of the γ-rays emitted by the source reaches the Timepix3 detector, and only a limited fraction of those interact with its sensor material, the observed event rate may be disproportionately smaller in comparison to the rate of emission. In the prospective applications of the presented system, the event rate is therefore expected to fluctuate between 10^5 and 10^6 events per second.

6.1.2 Task Description

The goal of the performance evaluation task is to determine the information rate, at which the presented system is capable of processing measured data. If the system is able to process data in real time for the purposes of the specified applications, the measured rate is expected fall into the range estimated in the previous section.

The tested components are selected stages of the producer/consumer chain described in Section 4.2. Since stages are applied sequentially to incoming data and the execution of each stage requires the output of the previous one, their evaluation is conducted as follows.

A Timepix3 detector output is simulated by re-playing a previously recorded acquisition from an ASCII file. Unlike a real detector, the simulation passes pixels to the chain as fast as possible, allowing to measure the smallest time required by the chain to process the entire file. Since the number of hits and events in the file is known, the time measurement gives a sufficient amount of information in order to estimate the rate of processing.

To investigate the time complexity of the individual stages of the chain, the measurement is started with empty chain, and repeated multiple times, adding a single stage to the chain each time. For the purposes of this task, four stages are evaluated:

1. Raw Pixel Reading
2. Calibration Function Evaluation
3. Monotonic Pixel Buffering
4. Spatial & Temporal Clustering

While the first stage serves as a control measurement, capturing only the complexity of the simulator, the remaining three stages contain the main logic of the chain.

To avoid bias due to the non-deterministic nature of preemptive scheduling, the entire experiment is repeated 10 times, and the presented results are aggregated as mean values over all attempts.

6.1.3 Results

The evaluation was performed on a computer with 2.7 GHz Intel Core i5 processor and 8 GB DDR3 RAM. The aggregated results are displayed in Table 6.1.

Last Stage	Time	Hit Rate	Event Rate
Raw Pixel Reading	1.168 s	$16.98 \cdot 10^6$ px/s	$1.78 \cdot 10^6$ ev/s
Calibration Function Evaluation	1.161 s	$17.08 \cdot 10^6$ px/s	$1.79 \cdot 10^6$ ev/s
Monotonic Pixel Buffering	3.153 s	$6.31 \cdot 10^6$ px/s	$0.66 \cdot 10^6$ ev/s
Spatial & Temporal Clustering	13.126 s	$1.51 \cdot 10^6$ px/s	$0.16 \cdot 10^6$ ev/s

Table 6.1: Mean results of the performance experiments.

Consistent with the initial expectations, computation time seems to increase along with the length of the processing chain. While calibration function evaluation seems to consume a negligible amount of processing time in comparison with raw pixel processing, the remaining two stages of the chain appear to produce a multiplicative slowdown of factors two and four, respectively. This observation seems to match the relative increase in complexity of algorithms executed at each stage.

Overall, the observed hit rates fall into the expected range for SPECT applications, suggesting that the presented system is a viable choice for processing measurements in a real time setting. Nevertheless, in the established worst-case scenario, the system is still considered prone to data congestion problems. This warrants the use of data rate safeguards and further research into the optimization of the presented work.

6.2 Interpolation Experiments

In the second set of experiments, the accuracy and the performance of the implemented interpolation methods is tracked in an isolated setting. While interpolation is usually performed multiple times during volume reconstruction, a single simplified interpolation task was selected in order to provide a better insight into the behavior of tested methods.

6.2.1 Task Description

The interpolation task closely models the one solved during forward projection of a cone onto a target plane. In it, a known mapping $r \circ f$ is first evaluated in a uniform lattice of $N \times N$ points from domain D. The interpolation then attempts to use values of these points to estimate values $r \circ f$ for arbitrary points in D without having to explicitly evaluate $r \circ f$.

For this set of experiments, D was chosen to be the two-dimensional interval $[-1.5, 1.5]^2$, and the mapping $r \circ f$ was chosen to correspond with the definitions listed in Section 5.2.3. Since f describes a unit circle, it is defined as:

$$f(x, y) = x^2 + y^2 - 1 \tag{6.1}$$

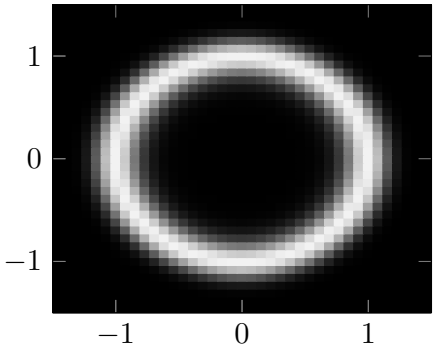


Figure 6.1: Values of the mapping $r \circ f$ prior to interpolation.

The response function r is an exponential defined for a real parameter $\sigma > 0$:

$$r(x) = \exp(-x^2/\sigma^2) \tag{6.2}$$

The resulting mapping $r \circ f$ (shown in Figure 6.1) returns values in $[0, 1]$ such that the points close to the circle receive large values, and points far from the circle receive small values. Furthermore, with increasing distance from the circle, the mapped values decrease exponentially.

6.2.2 Parameter Configurations

In the experiments, two previously introduced interpolation methods are evaluated:

1. nearest-neighbor interpolation,
2. bilinear interpolation

To illustrate the behavior of both methods under different circumstances, both methods are evaluated for input lattices of various densities N. This choice is motivated by the expectation that providing an interpolation method with more sample points should have a desirable effect on the output quality.

In all experiments, the real parameter σ of the response function r is set to $\sqrt{0.1}$.

6.2.3 Visual Demonstration

To produce visual demonstration of the interpolation process, images were produced by both tested interpolation methods. This corresponds with evaluation of the interpolation methods on a simple 50×50 point lattice, which represents individual pixels of the target image.

By decreasing values of the source lattice density N while maintaining the dimensions of the image, the information available as a basis for interpolation was incrementally reduced, possibly revealing various artifacts and undesirable patterns.

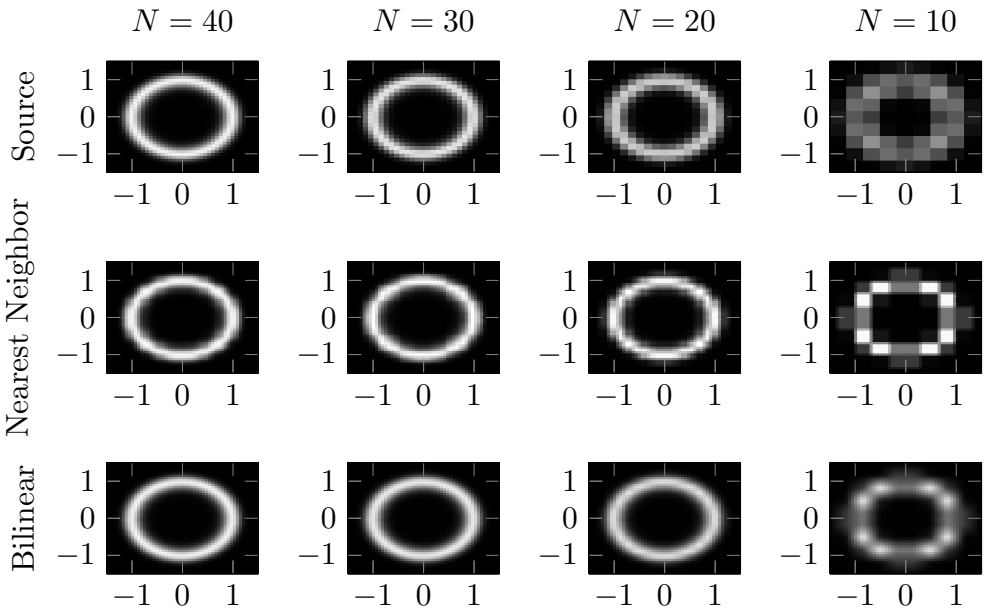


Figure 6.2: Example outputs of interpolation methods for various values of N.

The results (displayed in Figure 6.2) show that both methods seem to have successfully completed the interpolation task, creating images resembling the one shown in Figure 6.1.

In accordance with the initial expectation, the quality of the interpolated image seems to increase for larger values of N. For the particular circular shape created by $r \circ f$, bilinear interpolation appears to have distorted the image less than nearest-neighbor interpolation.

6.2.4 Evaluation by Random Sampling

Motivated by the results presented in the previous section, the second experiment was designed to further investigate the dependence of interpolation output quality on the source lattice density N.

In the experiment, interpolation methods were executed on sets of points, independently sampled from D at random. For each point, the value of $r \circ f$ was first evaluated directly from its explicit definition, and then estimated by the evaluated interpolation method. The interpolated value $\hat{y}$ was compared with the value y yielded by direct evaluation.

The quality of output is quantified by means of root-mean-square error, which is conventionally defined for values y_i and $\hat{y}_i$ as follows:

$$RMSE(y) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{6.3}$$

In addition to the interpolation error, the wall time of evaluation of $\hat{y}_i$ was tracked. To achieve sound measurements, the time of evaluation of the entire point set was measured, and the set size was chosen to be 10^6.

The experiments were performed for various densities $N \in \{20, 40, \ldots, 1000\}$. In order to achieve comparability of results, the same point set was evaluated

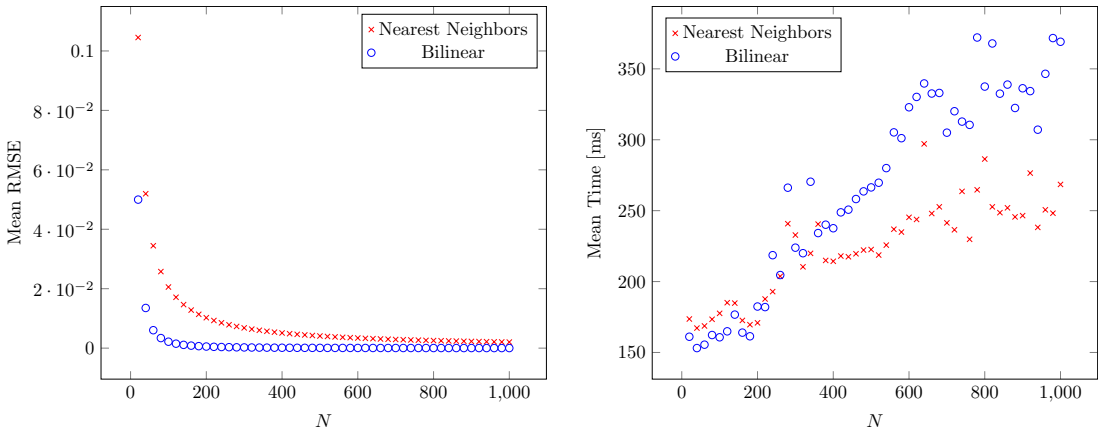


Figure 6.3: Results of the interpolation experiments.

in all experiments. To avoid bias due to random sampling, 10 independently sampled random point sets were used, and mean values were calculated across all sets.

After the experiments were performed, their outputs were successfully examined and aggregated. The results (shown in 6.3) confirm the prior expectation, indicating that in both cases, the interpolation error decreases as the number N of input lattice points increases. Furthermore, in accordance with the visual demonstration, bilinear interpolation seems to consistently outperform nearest-neighbor interpolation, achieving results of significantly lower error at all times.

Unlike interpolation error, the mean time of the computation was burdened by noise, presumably due to the non-deterministic nature of preemptive scheduling. Still, a discernible pattern may be spotted in the plots. The mean calculation time seems to consistently increase along with N. Moreover, the calculation time of nearest neighbors seems to increase at a slower rate than that of bilinear interpolation.

6.2.5 Discussion

Since the definition of both interpolation methods does not include any explicit dependence of their time complexity on N, it may be speculated that the observed relationship is due to the underlying implementation of the memory architecture, which may not always offer $\mathcal{O}(1)$ read access. This theory assumes a conventional multi-tier cache hierarchy, where read latencies are small for tiers which are physically closer to the processing unit, and increase in subsequent tiers.

Since the points of evaluation are chosen independently at random, the memory transfers during testing may be viewed as random reads. For small N, a significant fraction of the read values fits in the cache. This leads to more cache hits during interpolation, overall yielding low memory latency. Since the probability of a cache miss increases along with N, the expected read latency is also related to the value of N in the same way. In aggregation, this may be considered to be a plausible explanation for the observed increasing trend.

Curiously, this theory may also be used to explain why nearest-neighbor interpolation seems to outperform bilinear interpolation in evaluation time. While both methods are analogous in the sense that they consider four surrounding

points, their actual memory read counts differ. Whereas nearest-neighbor interpolation requires only one value corresponding to the selected nearest neighbor, bilinear interpolation always performs exactly four reads, and then combines the retrieved values with various weights. Since the bilinear method performs four times more reads than the nearest-neighbor method, any influence of N on the read latency is amplified, leading to a more prominent *slope*.

In the observed results, points of discontinuity may be consistently identified in evaluation times of both methods. In the presented theory, these points may correspond with transitioning points between individual tiers of the cache hierarchy.

6.3 Point Source Experiments

The third set of experiments demonstrates the operation of the volume reconstruction process on a simple point source. In a practical experimental setting, this would correspond to emissions originating from a source body (e.g. a radioactive element), observed through a collimator.

6.3.1 Task Description

In order to evaluate the volume reconstruction component independently of the readout component, the Compton events are simulated artificially at random. First, the volume V is defined as a product of intervals $[x_0, x_1] \times [y_0, y_1] \times [z_0, z_1]$. The volume is then discretized into N cells in each dimension, yielding N^3 cells in total. In the specified domain, the location M of the emission source is determined. This location serves as a basis for subsequent randomized generation of Compton events.

To produce a random Compton event related to the specified emission source M, parameters of the corresponding Compton cone $\mathcal{C}(V_1, V_2, \beta)$ are generated from specified prior distributions as follows. First, an arbitrary location representing the cone apex V_1 is selected. After that, an arbitrary vector representing the cone axis $-\overrightarrow{V_1V_2}$ is chosen, yielding the location of the point V_2. Finally, the angle β is calculated as the angle between vectors $\overrightarrow{V_1V_2}$ and $\overrightarrow{V_1M}$:

$$\cos\beta = \frac{\overrightarrow{V_1V_2} \cdot \overrightarrow{V_1M}}{\|\overrightarrow{V_1V_2}\| \, \|\overrightarrow{V_1M}\|} \tag{6.4}$$

By the derivation of the described method it holds that the selected source location M lies in the mantle of the constructed cone. The set of multiple such cones may then be viewed as simulated Compton events corresponding to the specified emission source.

The experiments are motivated by a simple notion of verification. If the presented volume reconstruction method is implemented correctly, the cells in the vicinity of M are expected to be assigned large values by the mapping $r \circ f$. Furthermore, in a volume aggregated for a sufficient number of non-trivial events, the global maximum is expected to determine the cell containing the point M. The values of cells adjacent to M may also provide an insight into the spatial resolution of the volume reconstruction method.

6.3.2 Parameter Configurations

For the purposes of the performed experiments, the volume V is defined as a cube of side equal to 20 pixels. The values of the interval bounds are determined by the following parameters:

$$\begin{array}{lll} x_0 = -10, & y_0 = -10, & z_0 = -5 \\ x_1 = 10, & y_1 = 10, & z_1 = 0 \end{array} \tag{6.5}$$

The number of cells in each dimension of the volume is set to $N = 300$, yielding $N^3 = 27 \cdot 10^6$ cells in total. Using conventional double precision floating-point number representation, this quantity corresponds to approximately 216 MB of memory. Note that since the volume is a cube and each its dimension contains the same number of cells N, each cell is consequently also a cube.

The mapping f and response function r are defined consistently with interpolation experiments by Expressions 6.1 and 6.2, respectively. The value of the smearing parameter is set to $\sigma = \sqrt{0.005}$.

Since it has proven to achieve more accurate results in the previous set of experiments, bilinear interpolation is used for all projections. The plane ρ is discretized into a uniform lattice of 100×100 points. Furthermore, the plane σ is positioned at $z = 20$. In order to avoid propagation of errors caused by cropped forward projections of cones with more extreme directions, the point lattice of σ is extended by factor of ten in both dimensions. To compensate for possibly increased interpolation errors, the plane is subsampled so that the uniform distance between points matches that of the rest of the volume.

The simulated emission source is placed at $M = [4, -2, 8]$. In order to mimic a stack of Timepix3 detectors arranged in telescopic configuration, the prior distribution for the apex V_1 of the simulated cones is restricted to $[-2, 2] \times [-2, 2] \times [-5, 0]$.

6.3.3 Single Event Projection

The goal of the reconstruction of the first event set is to present the results of the implemented forward and back projection algorithms. For the reasons of clarity, only one simulated event was included in the set. The results are expected to contain a singular cone surface projected into the volume.

The produced volume is displayed in Figure 6.4. For better visualization, multiple views of the volume are included, and in addition to the to the volume, three cuboid wireframes are plotted.

Volume Wireframe This wireframe is the largest of the three bodies. It describes the boundaries of the reconstructed volume V.

Emission Source Wireframe This wireframe is completely contained by the volume wireframe. Its center marks the point M of emission within V.

Apex Prior Distribution Wireframe This wireframe describes the boundaries, which constrain the prior distribution, from which the cone apex V_1 is sampled. It is located outside, yet adjacent to the volume wireframe. It may be viewed as a plausible location of the detection unit.

The observations of the volume seem to confirm all prior expectations. The plots show apparently continuous curved surface, consistent with a cone mantle. In an orthogonal cut, the surface resembles the two-dimensional circular pattern produced by the mapping $r \circ f$ shown in Figure 6.1.

The surface is divided into two separate components, which would presumably connect at the randomly chosen apex V_1 located outside the volume. Finally, the simulated emission source M seems to be directly intersected by one of the components of the described surface. Apart from the surface itself, no undesirable artifacts appear to be present within the volume.

In summary, the projection of a single Compton cone has fulfilled all expectations, demonstrating the correctness of both derived projection methods.

6.3.4 Localization Demonstration

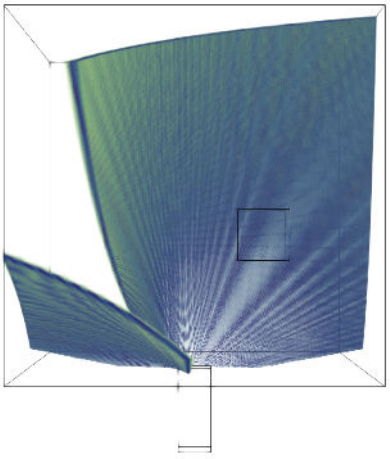
Motivated by the results of the previous experiment, the reconstruction of the second simulated event set aims to fully localize the emission source M based on the intersection of projected conic surfaces. For that reason, $5 \cdot 10^3$ events corresponding to the specified point source were simulated. After repeating the same projection procedure for every cone of the set and aggregating cell values by additive reduce operation, the results are expected to contain a single cluster located in a close vicinity of the point M.

The results of the experiment are plotted in Figure 6.5. For comparison with the previous experiment, the same camera locations and orientations are used. However, note that in order to ensure that the majority of the plotted volume is transparent, the range of the color map has been shifted, excluding cells with small values.

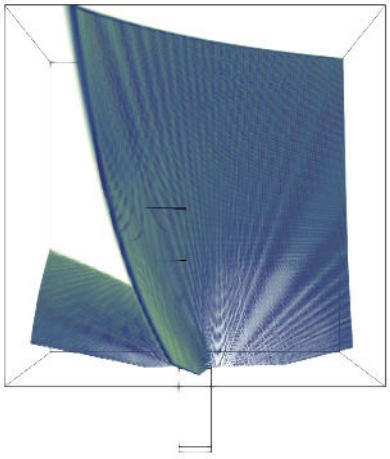
Similarly to the previous experiment, the results are in close agreement with the stated expectations. In the volume, the cells maximizing aggregated values form a singular cluster around the emission source M. In addition, cell values seem to increase exponentially when approaching M in axis-orthogonal cuts.

The described cluster does not appear to have a discrete boundary. Instead, cell values seem to rather continuously increase towards its center, where the maximum is located. Curiously, the derivative of the cell value appears to be significantly anisotropic. This may be observed for instance when comparing various axis-orthogonal views, thresholded at the same value. In particular, the smearing in the direction of the X- and Y-axis seems to be less prevalent than smearing in the direction of the Z-axis. This suggests a possible relative difference in localization accuracy between individual dimensions.

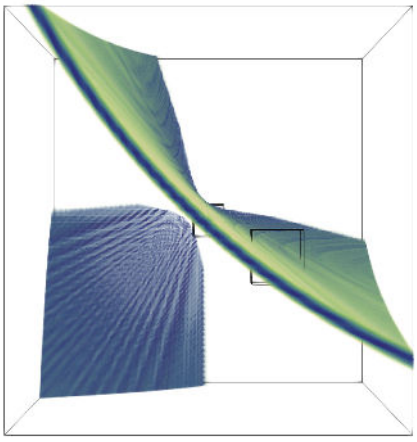
Overall, the experiment has successfully demonstrated the viability of the presented volume reconstruction method as means of emission source localization under ideal simulated conditions.



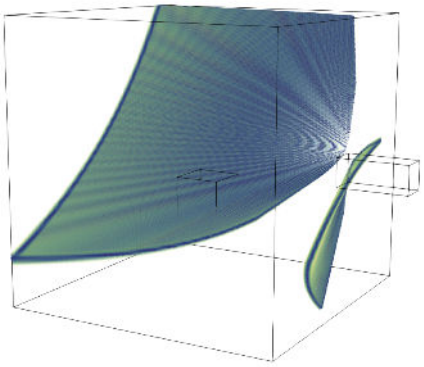
(a) Superior view.



(b) Lateral view.

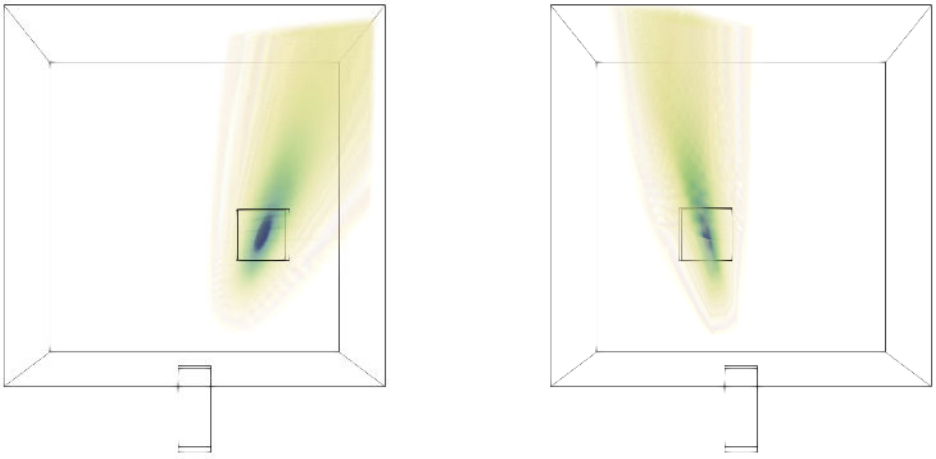


(c) Anterior view.



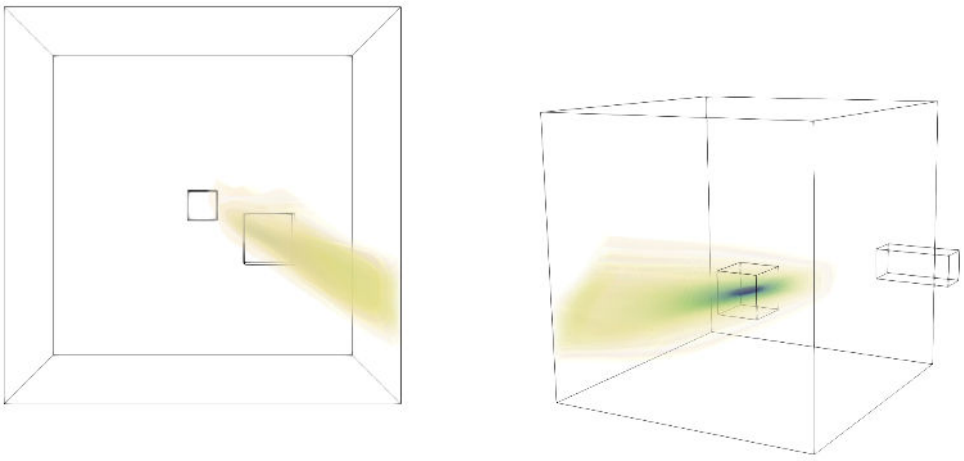
(d) Perspective view.

Figure 6.4: Reconstructed volume corresponding to a single event (various views).



(a) Superior view.

(b) Lateral view.



(c) Anterior view.

(d) Perspective view.

Figure 6.5: Reconstructed volume corresponding to $5 \cdot 10^3$ events (various views).

7. Conclusion

The goal of the thesis was to design and implement a software system capable of three-dimensional localization of γ-ray sources based on Timepix3 detectors. This goal has been successfully accomplished.

For detection of γ-rays, multiple Timepix3 detectors were arranged in a telescopic configuration, creating a Compton camera. In order to efficiently communicate with detectors, it was decided that a semi-autonomous readout device will relay application commands and aggregate detector outputs. For this purpose, the recently developed Katherine readout was selected.

To remotely control the Katherine readout by a proprietary UDP-based protocol, a novel C hardware library was developed and tested. The library is capable of issuing commands to the readout, inquiring its state and processing aggregated detector outputs in a real-time setting. Furthermore, the library is designed to operate independently of the rest of the system, and thus be redistributable for various other applications.

To recognize instances of Compton scattering in detector outputs, a producer/consumer chain of data processing algorithms was implemented with explicit consideration for possibly high volume of retrieved information. The chain consists of filter cascades, various calibration functions and morphological aggregation of spatially adjacent events. To optimize its performance with multiple detectors, the chain is divided into two components, where the first component is replicated for each detector, while the second component is common to all detectors. This allows a considerable speedup of operation in multi-threaded processing environment, where the first components of the chain may be executed simultaneously in parallel, independently of each other.

To perform three-dimensional γ-ray source localization based on the outputs of the processing chain, a conventional discretized approach to tomographic volume reconstruction was chosen. In the implemented algorithm, each observed Compton event is assigned a three-dimensional conic surface containing the unknown location of the emission source. Having aggregated a sufficiently large number of events, the most frequently intersected cells of the volume are identified. To calculate cell values related to intersection counts, a variant of the well-known back projection algorithm was implemented. To improve its scalability with respect to volume size, the algorithm was divided into more computationally intensive forward projection, which is performed only once for each event, and a less complex back projection, which is executed multiple times. For each of these projections, a corresponding perspective mapping was algebraically derived. In order to avoid evaluation of non-linear response function in each cell of the volume, two common interpolation methods were implemented.

Due to manufacturing delays, it was not possible to evaluate the presented system as a whole with a real hardware assembly at the time of writing. For that reason, its individual components were evaluated instead. The hardware library was successfully tested with a singular Katherine readout device. The behavior of the implemented interpolation functions was examined for various counts of sampled points. Lastly, the correctness of the derived perspective projections was demonstrated on artificially generated conic surfaces. The results indicate that

under ideal simulated conditions, the presented implementation of back projection algorithm is a viable localization method of γ-ray point sources.

7.1 Future Research

The presented work offers many opportunities for further investigation and improvement. Since Timepix3 assemblies are theoretically capable of producing up to 80 million detected events per second, a natural avenue of exploration presents itself in the field of performance optimization of all presented algorithms. With greater processing speeds, stronger γ-ray sources may be observed with the device, reducing overall scan time.

On the hardware level, utilization of wide vector registers or modern many-core architectures such as GPUs may yield a significant speedup of data processing. In addition, a considerable performance improvement may be obtained by employing iterative tomographic techniques, which repeatedly trim the volume of empty cells, refine the reconstruction, and refocus on the remaining regions.

Further work is also required in practical integration of the presented technology with industrial SPECT systems. The experimental nature of the prototype assembly warrants implementation of fault tolerant behaviors and user safeguards protecting against adversarial configurations, which may lead to undesirable states. In addition, a qualitative comparison study of the γ-ray source localization accuracy is necessary before integrating the system with any commercial solutions.

In addition to improvements in the software solution, optimizations of the hardware configuration may also be explored. For instance, to improve localization accuracy, the detection unit may be positioned at various angles with respect to the scanned object, or mounted on a ring frame equipped with actuators. Consequently, more sophisticated tomographic algorithms may be applied to efficiently correlate multiple projections of the same body in the volume reconstruction phase.

Finally, the visualization of the reconstructed volumes also warrants further research. In commercial volume visualization tools, the user is conventionally presented with the results in the form of parallel slices or an interactive three-dimensional model. Recent technological advances have however opened possibilities of efficiently rendering volume representations in mobile devices, such as tablets with augmented reality capabilities or virtual reality headsets like Oculus Rift[1]. Thanks to their flexibility and portability, such devices may have the potential for creating a more comprehensible user experience.

[1] `https://www.oculus.com/rift/`

Bibliography

[1] T Poikela, J Plosila, T Westerlund, M Campbell, M De Gaspari, X Llopart, V Gromov, R Kluit, M van Beuzekom, F Zappon, et al. Timepix3: a 65K channel hybrid pixel readout chip with simultaneous ToA/ToT and sparse readout. *Journal of instrumentation*, 9(05):C05013, 2014.

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

[3] AB Newberg and A Alavi. Single photon emission computed tomography (SPECT): Technique. *New Encyclopedia of Neuroscience*, 2008.

[4] Staffan Jacobsson Svärd. *A tomographic measurement technique for irradiated nuclear fuel assemblies*. PhD thesis, Acta Universitatis Upsaliensis, 2004.

[5] O Gal, C Izac, F Jean, F Lainé, C Lévêque, and A Nguyen. Cartogam–a portable gamma camera for remote localisation of radioactive sources in nuclear facilities. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 460(1):138–145, 2001.

[6] Frédérick Carrel, Roger Abou Khalil, Sébastien Colas, Daniel de Toro, Gilles Ferrand, Emmanuelle Gaillard-Lecanu, Mehdi Gmar, Daniel Hameau, Sylvie Jahan, Frédéric Lainé, et al. Gampix: A new gamma imaging system for radiological safety and homeland security purposes. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2011 IEEE*, pages 4739–4744. IEEE, 2011.

[7] Apostolos Kantzas, Kelly Hamilton, Taghi Zarabi, Amit Bhargava, Ian Wright, Glen Brook, and Jinwen Chen. Application of gamma camera imaging and SPECT systems in chemical processes. *Chemical Engineering Journal*, 77(1-2):19–25, 2000.

[8] RW Todd, JM Nightingale, and DB Everett. A proposed γ camera. *Nature*, 251(5471):132, 1974.

[9] DB Everett, JS Fleming, RW Todd, and JM Nightingale. Gamma-radiation imaging system based on the compton effect. In *Proceedings of the Institution of Electrical Engineers*, volume 124, pages 995–1000. IET, 1977.

[10] Hal O Anger. A new instrument for mapping gamma-ray emitters. *Biology and Medicine Quarterly Report UCRL*, 3653:38, 1957.

[11] Manbir Singh and David Doria. Single photon imaging with electronic collimation. *IEEE Transactions on Nuclear Science*, 32(1):843–847, 1985.

[12] Manbir Singh and R Ricardo Brechner. Experimental test-object study of electronically collimated SPECT. *Journal of Nuclear Medicine*, 31(2):178–186, 1990.

[13] JE Gormley, N Clinthorne, GF Knoll, JW LeBlanc, WL Rogers, DK Wehe, and SJ Wilderman. Effects of shared charge collection on compton camera performance using pixellated ge arrays. In *JOURNAL OF NUCLEAR MEDICINE*, volume 37, pages 745–745. SOC NUCLEAR MEDICINE INC 1850 SAMUEL MORSE DR, RESTON, VA 22090-5316, 1996.

[14] JW LeBlanc, NH Clinthorne, C Hua, WL Rogers, DK Wehe, and SJ Wilderman. A compton camera for nuclear medicine applications using 113min1. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 422(1-3):735–739, 1999.

[15] Anne C Sauve, AO Hero, W Leslie Rogers, SJ Wilderman, and NH Clinthorne. 3d image reconstruction for a compton SPECT camera model. *IEEE Transactions on Nuclear Science*, 46(6):2075–2084, 1999.

[16] Jerome Edward Gormley, WL Rogers, NH Clinthorne, DK Wehe, and GF Knoll. Experimental comparison of mechanical and electronic gamma-ray collimation. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 397(2-3):440–447, 1997.

[17] Li Han, W Leslie Rogers, Sam S Huh, and Neal Clinthorne. Statistical performance evaluation and comparison of a compton medical imaging system and a collimated anger camera for higher energy photon imaging. *Physics in Medicine & Biology*, 53(24):7029, 2008.

[18] Benedikt Bergmann, Martin Pichotka, Stanislav Pospisil, Jiri Vycpalek, Petr Burian, Pavel Broulim, and Jan Jakubek. 3D track reconstruction capability of a silicon hybrid active pixel detector. *The European Physical Journal C*, 77(6):421, 2017.

[19] Xavier Lojacono. *Image reconstruction for Compton camera with application to hadrontherapy*. Theses, INSA de Lyon, November 2013.

[20] Arthur H Compton. A quantum theory of the scattering of x-rays by light elements. *Physical review*, 21(5):483, 1923.

[21] Arthur H Compton. The spectrum of scattered x-rays. *Physical Review*, 22(5):409, 1923.

[22] The Medipix Collaboration. The Medipix Chips and Collaborations: from medical imaging to space dosimetry., 2018. `https://kt.cern/success-stories/medipix-chips-and-collaborations-medical-imaging-space-dosimetry` [Accessed: 17/02/2018].

[23] Jakub Begera. Calibration and control software for network of particle pixel detectors within the Atlas experiment at the LHC at CERN. Bachelor's thesis, Faculty of Electrical Engineering, Czech Technical University in Prague, 2016.

[24] D Turecek, J Jakubek, and P Soukup. Usb 3.0 readout and time-walk correction method for timepix3 detector. *Journal of Instrumentation*, 11(12):C12065, 2016.

[25] P Burian, P Broulím, M Jára, V Georgiev, and B Bergmann. Katherine: ethernet embedded readout interface for Timepix3. *Journal of Instrumentation*, 12(11):C11001, 2017.

[26] V Kraus, Michael Holik, Jan Jakubek, M Kroupa, P Soukup, and Z Vykydal. FITPix—fast interface for Timepix pixel detectors. *Journal of Instrumentation*, 6(01):C01079, 2011.

[27] J Visser, M van Beuzekom, Henk Boterenbrood, B van der Heijden, JI Muñoz, S Kulis, B Munneke, and F Schreuder. SPIDR: a read-out system for medipix3 & timepix3. *Journal of Instrumentation*, 10(12):C12028, 2015.

[28] *Katherine: Command Set*, November 2017.

[29] *POSIX Programmer's Manual*, April 2013.

[30] Cameron. A fast lock-free queue for C++, 2013. `http://moodycamel.com/blog/2013/a-fast-lock-free-queue-for-c++` [Accessed: 19/03/2018].

[31] Dan E Dudgeon and Russell M Mersereau. *Multidimensional Digital Signal Processing Prentice-Hall Signal Processing Series*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[32] Gabor T Herman. *Fundamentals of computerized tomography: image reconstruction from projections*. Springer Science & Business Media, 2009.

[33] Earl J Kirkland. Bilinear interpolation. In *Advanced Computing in Electron Microscopy*, pages 261–263. Springer, 2010.

[34] T Holy, E Heijne, J Jakubek, S Pospisil, J Uher, and 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*, 591(1):287–290, 2008.

List of Figures

Acronyms

CERN The European Organization for Nuclear Research (also known as *Organisation européenne pour la recherche nucléaire*)

SPECT Single-photon Emission Computed Tomography

CCD Charge-coupled Device

MPX The Medipix Collaboration

ASIC Application Specific Integrated Circuit

VHDCI Very High Density Cable Interconnect

GPIO General Purpose Input/Output

LEMO Léon Mouttet (connector standard)

FIFO First in, first out

OS Operating System

RAM Random Access Memory

NIC Network Interface Controller

IP Internet Protocol

UDP User Datagram Protocol

SFTP Secure File Transfer Protocol

VCO Voltage-controlled Oscillator

ToT Time-over-Threshold

iToT Integral Time-over-Threshold

ToA Time-of-Arrival

fToA Fast Time-of-Arrival

RMSE Root-mean-square Error

HTML Hypertext Markup Language

YAML YAML Ain't Markup Language

VTK Visualization Toolkit

GNU GNU's Not Unix

POSIX Portable Operating System Interface

A. Attachments

A.1 Contents of the Enclosed DVD

- `README.md` ..... description of the DVD contents
- `data/` ..... examples of input data files
- `outputs/` ..... results of the performed experiments
- `work/` ..... thesis implementation source files and headers
  - `katherine/` ..... Katherine hardware library
  - `ccl/` ..... Compton camera library
  - `krun/` ..... Katherine data acquisition tool
  - `kfind/` ..... Katherine network localization utility
  - `calib_drift_time/` ..... drift time calibration tool
  - `cluster_view/` ..... 3D cluster viewer
  - `random_point_gen/` ..... random point generator
  - `random_cone_gen/` ..... random cone generator
  - `benchmark/` ..... performance benchmark
  - `test_interpolation/` ..... interpolation test program
  - `test_forward_projection/` ..... forward projection test program
  - `test_back_projection/` ..... back projection test program
  - `test_projector/` ..... volume reconstruction test program
- `text/` ..... thesis text
  - `src/` ..... structured thesis text in the LaTeX format
  - `img/` ..... figures used in the thesis text
  - `data/` ..... data files used to generate plots
  - `plots/` ..... plots used in the thesis text
  - `thesis.pdf` ..... compiled thesis text in PDF format

A.2 Software Implementation Overview

This text provides a practical overview of the software included in the electronic attachment.

A.2.1 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 6.14 or newer with the support for Qt, Python and C++17,
- VTK Visualization Framework 8.1 or newer,
- Python 3.6 or newer with the `matplotlib` package,
- Qt GUI Framework 5.10 or newer.

Optionally, the project documentation requires Doxygen 1.8 or newer.

A.2.2 Provided Docker Images

For convenient usage, the electronic attachment is compatible with the Docker virtualization platform. This way, the included programs may be easily compiled without any dependence on the host operating system. In particular, the work provides two Docker images:

`pm_thesis_mff_env` (in `Dockerfile.env`) Extending the latest version of the Archlinux image, this image compiles all required dependencies listed in the previous section.

`pm_thesis_mff` (in `Dockerfile.work`) Extending the environment image, this image contains a working installation of all software tools described in this summary.

For usage of both 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.

A.2.3 Directory Structure

The attached implementation is provided in the form of a single CMake project. For convenient access, the software is semantically organized into multiple *targets* – programs and libraries, which are referenced as subprojects of the main project. This way, all targets are compiled together by default. It is, however, easy to selectively compile or install individual targets, if required.

Generally, targets are represented by individual directories located in the `work` directory. Each directory has a systematic structure:

```
work/
└─ target/
   ├─ src/ ..... source files
   ├─ include/ ..... header files (only in libraries)
   ├─ docs/ ..... documentation (only in libraries)
   ├─ README.md ..... usage information
   └─ CMakeLists.txt ..... build declaration, dependency list
```

Note that the contents of the `docs` directory have been automatically generated by the Doxygen documentation generation utility. If any changes are made to the source files or headers, the documentation may be simply updated by executing the `doxygen` program. For more information about Doxygen, see `http://doxygen.org`.

A.2.4 Redistributable Libraries

To allow easy integration into future works, the majority of the presented implementation is structured in the form of redistributable library packages. In particular, the work is divided into two libraries:

Katherine Hardware Library (`libkatherine`) This C library contains implementation of Katherine proprietary communication protocol. It is further described in Chapter 3.

Compton Camera Library (`ccl`) This C++ library contains implementation of data processing algorithms described in Chapters 4 and 5.

While the first of the two is a development library in a conventional sense, designed to be built and later linked with other applications, the second is *a header-only library*, exclusively comprised of header files. This decision is primarily motivated by its extensive use of C++ template engine and heavy reliance on inline optimizations provided by the compiler backend. For practical usage, this only means that for the second library, no prior build operation and linking is required, however, overall build time of any application referencing the library headers may be increased.

The Katherine library depends on the C11 standard library and GNU/Linux multi-threading and network socket interface. The Compton Camera Library depends on the C++17 standard library, GNU/Linux multi-threading and optionally, the Katherine library.

For a high-level overview of the architecture and features provided by both libraries, the reader is encouraged to examine the referenced chapters of this text. In addition, both implementations include appropriately structured developer documentation, which may be compiled into a HTML page or LaTeXdocument by Doxygen. Build instructions are noted in the provided README files.

A.2.5 Katherine Data Acquisition Tool

Katherine Data Acquisition Tool is capable of performing simple data acquisitions with the Katherine readout. Consistently with the description provided in Section 3.1, the tool establishes connection with the readout device, configures its parameters and initiates data acquisition. The program source files are located-in the `krun` directory and are reference in the main CMake project.

Upon execution, the program expects a valid acquisition configuration file. Among others, the contents of this file determine the acquisition duration and readout mode. The results of the acquisition are printed in the standard output, allowing the user to chain the program with other scripts or applications, or save its results into a file.

A.2.6 Katherine Network Localization Utility

The purpose of the Katherine Network Localization Utility is to identify operational Katherine readouts in a specified region of the network infrastructure. This may be a necessary task when connecting a new Katherine readout to the network for the first time. The utility source files are saved in the `kfind` directory and are referenced in the main CMake project.

When executed, the utility performs a sequential search, during which multiple network addresses are probed one-by-one. The probe message is a state inquiry command, which prompts an immediate informative response by the readout. After transmitting the message, the utility waits for the arrival of such response. If the response arrives within a short timeout period, the tested address is considered to be a found Katherine readout.

A.2.7 Drift Time Calibration Tool

Drift Time Calibration Tool contains an implementation of the calibration procedure, which is necessary in order to reconstruct depth information from pixel timestamp differences [18]. This calibration is not required for the operation of the presented algorithms. It is, however, recommended as it may have positive effect on the accuracy of the subsequent reconstruction. The source files of the tool are saved in the `calib_drift_time` directory and are referenced in the main CMake project.

The drift time calibration operates on a record of a prior measurement, which is expected to be provided in the form of an ASCII file. During the calibration procedure, the spatial & temporal clustering algorithm described in Section 4.2.3 is executed. Identified clusters are then filtered based on their morphological classification [34]. Finally, the duration determined by the smallest and the largest pixel timestamp of each cluster is examined. A bell curve is fitted from the durations and the fit parameters are presented to the user.

A.2.8 3D Cluster Viewer

The purpose of the 3D Cluster Viewer is to demonstrate the evaluation of the drift time calibration function described in Section 2.3.5 and previous works [18]. The program source code is located in the `cluster_view` directory and is referenced in the main CMake project.

Upon execution, the program requires calibration function parameter values, and a measurement ASCII file similar to that read by the Drift Time Calibration Tool. The program presents the user with a graphical user interface, which plots ToA and ToT values of individual clusters in a two-dimensional pixel matrix. The reconstructed depth information is displayed in a three-dimensional rendering of the detector in the bottom part of the screen.

A.2.9 Performance Benchmark

The purpose of the Performance Benchmark is to evaluate the rate of information consumed by selected components of the presented system. It may be used to re-create the results of the experiments described in Section 6.1. The program sources are located in the `benchmark` directory and are referenced in the main CMake project.

When executed, the benchmark program expects an ASCII file containing a recording of an earlier data acquisition as well as calibration function parameter files. The program performs several performance tests and for each test prints the measured time, the hit rate and the event rate (if applicable).

A.2.10 Random Generators

For some of the included programs, input data is required. For occasions, when the user is not in possession of real measurements or experimental devices, random generators are provided in order to produce a faithful substitute of experimental results. In particular, two random generators are provided:

Point Generator (in `random_point_gen`) This generator samples points from a two-dimensional domain. The points are selected uniformly from a set interval.

Cone Generator (in `random_cone_gen`) This generator samples parameters of random Compton cones, which correspond to a single specified point source. The cones are selected uniformly from set prior distributions.

Both generators have the capability to condition the pseudo-random number generator on a specified seed number. This may be used to obtain reproducible results.

A.2.11 Interpolation Test Program

The purpose of the Interpolation Test Program is to benchmark the accuracy of selected interpolation methods. It may be used to reproduce evaluation results presented in Section 6.2. The program source code can be found in the `test_interpolation` directory and is referenced in the main CMake project.

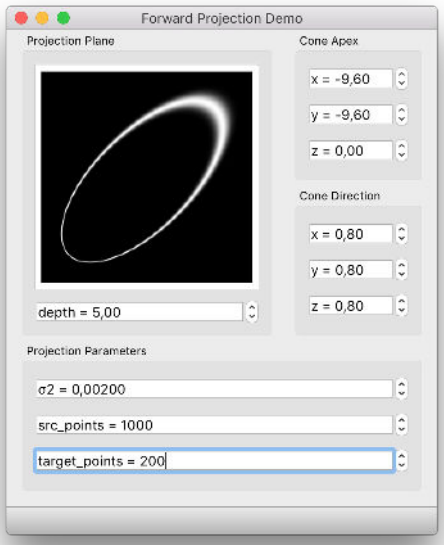


Figure A.1: Forward Projection Test Program

For execution, the test program requires a file containing two-dimensional points, in which the interpolation is performed. These points may be for instance produced earlier by the Point Generator. In the output, the program prints interpolation errors for the given point set, correlated with point count of the look-up table.

A.2.12 Forward Projection Test Program

The purpose of the Forward Projection Test Program (shown in Figure A.1) is to demonstrate effects of changes of various parameters on the results of the forward projection algorithm described in Section 5.1.3. The program source files are located in the `test_forward_projection` directory and are integrated with the main CMake project.

Since the program has a purely demonstrative nature, it has no inputs or outputs. Instead, it presents the user with a graphical interface, in which a forward projection of a cone is displayed. The interface allows the user to change various parameters of the projected cone, the projection plane and the parameters of the projection, and to observe the effects of the change on the displayed rendering.

A.2.13 Back Projection Test Program

Analogous to the Forward Projection Test Program, the purpose of the Back Projection Test Program (shown in Figure A.2) is to demonstrate the operation of the back projection algorithm described in Section 5.1.4. The program source code is saved in the `test_back_projection` directory and is referenced in the main CMake project.

The program shows a graphical interface, in which two projections of a singular cone are depicted. The left view shows the forward projection, which serves as a two-dimensional look-up table for the back projection algorithm. The result

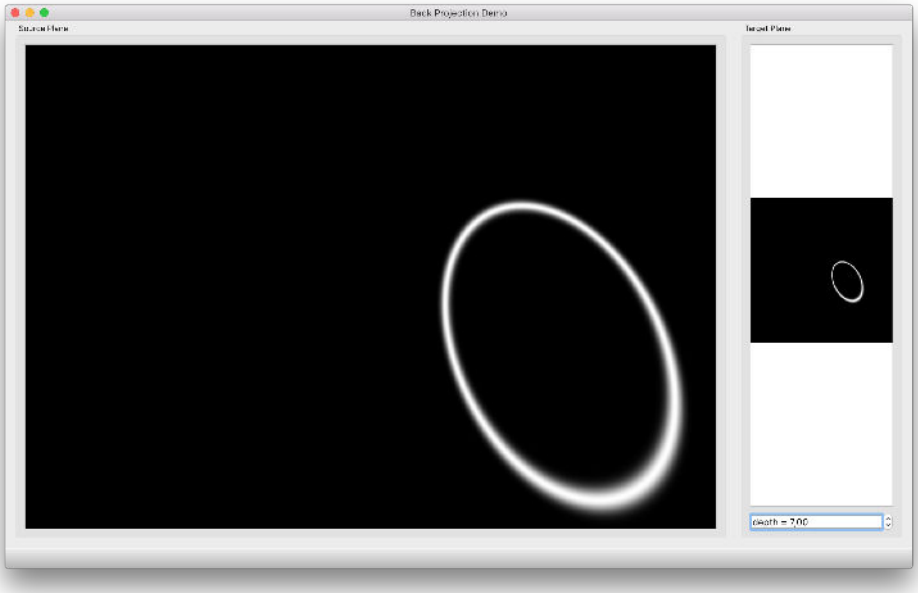


Figure A.2: Back Projection Test Program

of back projection is displayed in the right view, expected to closely mimic the forward projection. By changing the depth of the secondary plane, the user may observe effects of various depths on the rendered image, thereby verifying the correctness of the back projection algorithm.

A.2.14 Volume Reconstruction Test Program

As its name suggests, the purpose of the Volume Reconstruction Test Program is to perform volume reconstruction for a specified set of Compton cones. The list of cones is expected to be provided in an ASCII input file. Such file may be for instance generated earlier by the Random Cone Generator. The reconstructed volume is saved into a sparse list of cells, which may be visualized by commercial or open-source volume viewers, such as ParaView[1]. The program sources are located in the `test_projector` directory and are integrated with the main CMake project.

Upon execution, the program performs Algorithm 10. First, the response function look-up table is calculated for all cones in advance. After that, an empty volume is allocated. Cones then sequentially undergo the projection procedure, which consists of a projection forward and back, incrementing values of all cells within the volume. Once all cones are processed, the output file containing volume cell values is printed.

[1]ParaView is an open-source, multi-platform data analysis and visualization application. Further information about ParaView is available at `https://www.paraview.org`.