Johannes Gutenberg University Mainz

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

JG|U

Faculty 08 Physics, Mathematics and Computer Science
Institute of Computer Science and Institute for Nuclear Physics
AG Berger

Master Thesis

Online Event Selection using GPUs for the Mu3e Experiment

Fritz Valentin Henkys

1. Reviewer Prof. Dr. Niklaus Berger
Institute for Nuclear Physics
Johannes Gutenberg University Mainz

2. Reviewer Prof. Dr. Bertil Schmidt
Institute of Computer Science
Johannes Gutenberg University Mainz

February 09, 2022

Fritz Valentin Henkys

Online Event Selection using GPUs for the Mu3e Experiment

Master Thesis, February 09, 2022

Reviewers: Prof. Dr. Niklaus Berger and Prof. Dr. Bertil Schmidt

Johannes Gutenberg University Mainz

AG Berger

Institute of Computer Science and Institute for Nuclear Physics

Faculty 08 Physics, Mathematics and Computer Science

Staudingerweg 09

55128 Mainz

Abstract

The Mu3e experiment searches for physics beyond the Standard Model using the lepton flavour violating decay $\mu^+ \to e^+e^-e^+$. Observing this heavily suppressed decay or setting a new upper limit on the branching ratio to $2 \cdot 10^{-15}$ requires a high muon rate and a detector with high momentum and vertex resolution. These requirements result in high data rates of 100Gbps of sensor data, mainly consisting of noise.

In this work we present improvements and the implementation of the Online Event Selection algorithm, used to filter this datastream. It is used to reduce the data rate by a factor of over 100, by classifying time slices of detector data as important. We utilize the massively parallel architecture of graphics processing units using CUDA, achieving a speedup of over 2 compared to the previous implementation, while being able identify more than 94% of signal time slices correctly.

Zusammenfassung

Auf der Suche nach neuer Physik jenseits des Standardmodells untersucht das Mu3e Experiment den stark unterdrückten Zerfall des Myons $\mu^+ \to e^+e^-e^+$. Das Beobachten dieses Zerfalls würde neuen physikalischen Theorien den weg bereiten. Mu3e versucht entweder diesen zu entdecken oder ein neues oberes Limit für die Zerfallsbreite des Zerfalls auf $1 \cdot 10^{-15}$ zu setzen.

Der hierfür genutzte Detektor wurde mit Fokus auf einer hohen Impuls- und Vertexpräzision entwickelt um die Zerfälle von $1 \cdot 10^8 \mu$/s beobachten zu können. Diese Menge an Zerfällen erzeugt eine geschätzte Menge von 100Gbps an Daten, die zu großen Teilen aus Hintergrundprozessen bestehen. Um diese Datenrate um den Faktor 100 zu reduzieren wurde die Online Event Selection entwickelt, der Zeitabschnitte der Detektordaten in Echtzeit analysiert. Es wird beurteilt ob der Zeitabschnitt interessant aussieht und nur dann weitergeleitet.

In dieser Arbeit stellen wir diesen Prozess inklusive Verbesserungen vor. Wir haben den Algorithmus auf Grafikkarten von NVIDIA, mit Hilfe von CUDA implementiert und erweitert. Unsere Version ist dabei doppelt so schnell wie die vorherige Implementierung und schafft es mehr als 94% aller Zeitabschnitten mit dem Signalevent richtig zu klassifizieren und über 97% der Partikelpfade korrekt zu rekonstruieren.

Contents

1 Introduction

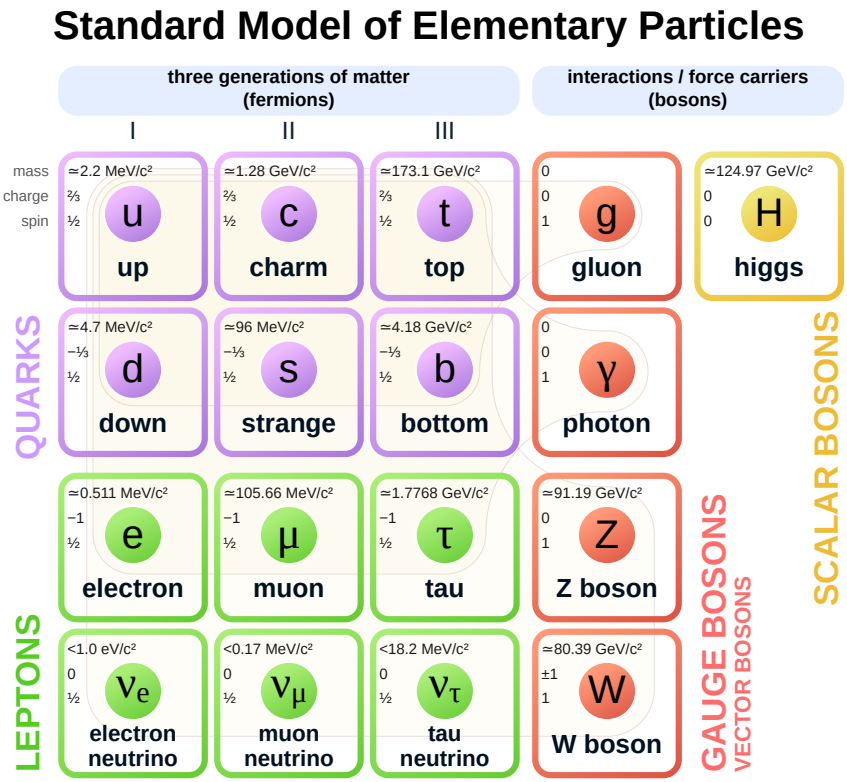


Fig. 1.1: Elementary particles of the Standard Model.

Source: [42]

The *Standard Model* (SM) of particle physics [18] is a well known and established model describing the elementary particles, which are shown in Fig. 1.1, as well as three of the four fundamental interactions. The final particle, the Higgs-Boson, of the SM was observed in 2012 at the Large Hadron Collider (LHC) in CERN, and awarded with the Nobel Prize [1]. While it is a successful and self-contained model, it is missing pieces for describing 'everything'. For example the gravitational interaction, one of the four fundamental interactions, is not covered by it. Other phenomena like dark matter and energy are not explained by it as well, fueling the need for research beyond the SM.

Since the SM does not encompass everything, physicists search for theories to explain those phenomena. One of those phenomena is the observation of neutrino oscillation in the Super Kamiokande and Sudbury Neutrino Observatories, awarded with the Nobel Prize 2015 [39].

The new theories need to be validated by experiments, like *Mu3e*. These experiments have high requirements for precision, resulting in huge amounts of data to be collected. To process and validate the measured data a mixture of online and offline analysis is used, requiring high performant hardware and implementation.

1.1 Lepton Flavour Violation

Neutrino oscillation introduced the first observation of *Lepton Flavour Violation*. Lepton Flavour is a conserved property according to the SM and is described by the *Lepton number* L_ℓ:

$$L_\ell = n_\ell + n_{\bar{\ell}}, \tag{1.1}$$

where n_ℓ and $n_{\bar{\ell}}$ are the number of leptons and antileptons in the family [18]. The lepton families, as seen in Fig. 1.1, are:

- Electrons,
- muons,
- tauons.

L_ℓ, $\ell \in \{e, \mu, \tau\}$ has to hold over all lepton families together as well as for each family itself in each reaction.

Possible reactions are Lepton decays, like the Michel Decay $\mu^+ \to e^+\nu_e\nu_{\bar{\mu}}$, which is the most common decay for muons at rest. It has a predicted probability of close to 100% [6]. This probability of a particle decay happening is called the *branching ratio*.

The observation of neutrino oscillation shows that lepton flavour may not have to be conserved in all reactions. This opens up the possibility for many reactions, previously heavily suppressed by the SM, to be observed, like $\mu^+ \to e^+e^+e^-$. This decay has according to the SM a branching ratio of $< 10^{-54}$, thus making it nigh unobservable by experiments in the near future [6]. Observing this decay would be a big step for finding new physics beyond the SM. The Mu3e experiment searches for the aforementioned decay with higher efficiency than previous experiments, aiming to find the decay or setting a new upper bound for the branching ratio, previously set by SINDRUM to $< 10^{-12}$ [9].

1.2 Online Data Processing

Since new events have low predicted branching ratios and were not previously observed by other experiments, they require high event frequencies to be observable in an acceptable amount of time, as well as good resolutions to be able to observe them at all [6]. Both of these result in a requirement for high frequency particle beams and thus very high rates of measured data by the detectors. As an example the ALICE experiment at LHC expects to detect 3TBps of detector data in their third run [31].

These data rates often include a lot of noise and background signals, not needed for storage or further analysis [6]. Alternatively the data is used in real time for important trigger decisions [21], for online calibration or is compressed [32]. Most of these tasks require a lot of independent calculations, perfectly fit for parallel hardware like *Field Programmable Gate Arrays* (FPGA) or *Graphics Processing Units* (GPU).

A lot of these detectors use a mix of FPGAs and GPUs for different tasks. While the FPGAs are used for packing and streaming the data [6], the GPUs perform many of analysis calculations, like particle identification, by track reconstruction. These tasks are offloaded to GPUs due to their better floating point performance compared to FPGAs and high amount of parallelism compared to CPUs.

In this thesis we build upon the *Online Event Selection* previously introduced by D. vom Bruch [13]. This algorithm filters the incoming data stream for storing and offline analysis, removing out as much unnecessary data as possible. In this thesis we are presenting each step of the algorithm. We have analyzed each step and added improvements in speed and accuracy in each step and fully implemented and tested the algorithm on the GPU. Our implementation achieves a speedup of 2 for the targeted muon rate of $1\mu/\text{s}$ compared to the previous implementation.

We will first introduce the Mu3e experiment in Chapter 2, then explain the algorithm in Chapter 3, leading to the implementation on GPUs in Chapter 4.

2

Mu3e Experiment

In the search for new physics beyond the standard model, the Mu3e experiment [6] searches for the lepton flavour violating decay $\mu^+ \rightarrow e^+e^+e^-$, giving it its name. A previous upper limit on the branching-ratio of this decay was set by the SINDRUM experiment in 1988 [9] to 10^{-12} with a 90% confidence level. Mu3e aims to improve these results by up to three orders of magnitude in its first phase.

The experiment is planned to be performed in two phases. In phase I the project uses the πE5 beam line at the Paul-Scherrer-Institute (PSI) with a muon rate of $1 \cdot 10^8 \mu/s$. Over the course of one year data will be collected observing over $2 \cdot 10^{15}$ muon-decays, setting a new upper limit for or measuring the branching ratio of this decay. For phase II the High Intensitiy Muon Beam (HIMB) at PSI will be used, providing an estimated muon rate of over $1 \cdot 10^9 \mu/s$, with the goal of achieving an even higher sensitivity of 10^{-16} detected muon decays. In this chapter we will introduce the phase I experiment [6].

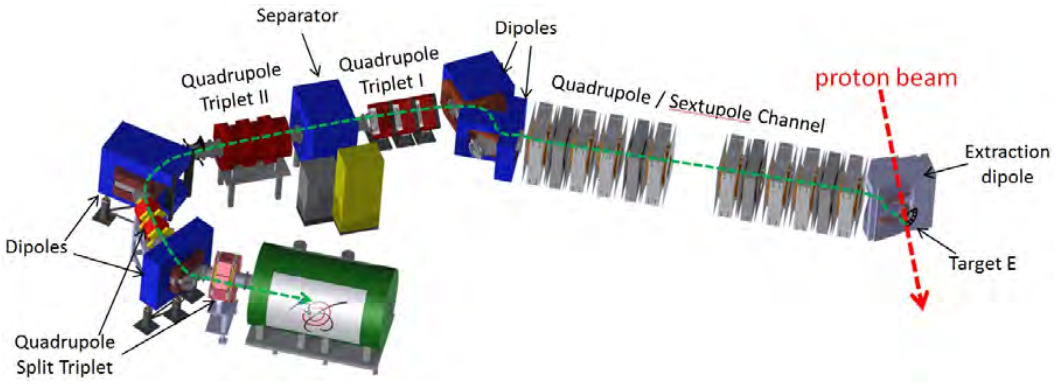
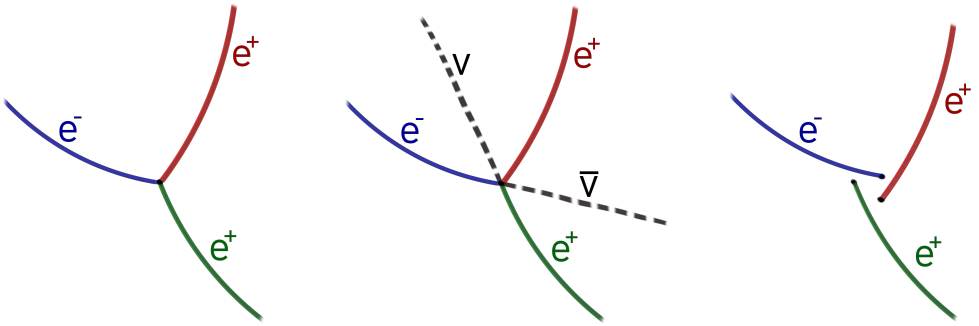


Fig. 2.1: The compact muon beamline used for the Mu3e experiment as a CAD-Model. The incoming proton beam (red) is aimed at target E, where muons are created and bundled into a muon beam (green), which is then aimed at the Mu3e experiment. Source: [16]



a) Signal b) Internal Conversion c) Combinatorial

Fig. 2.2: Schematics of the signal and background topologies. Electron tracks are in blue, positrons in red. The signal a) has a clear point of origin for all tracks and no extra particles, In contrast internal conversion b) produces two extra particles, absorbing some of the electron/positron momentum. For the combinatorial background c) no clear point of origin exists for the three tracks.
Source: [16]

2.1 Experimental Setup

The *High Intensity Proton Accelerator* (HIPA) at PSI [36] creates a proton beam of 590MeV. When directing the beam at a carbon target, it produces positive pions π^+ at the target surface, which then decay into muons via the decay

$$\pi^+ \to \mu^+ \nu_\mu. \tag{2.1}$$

Since this is not the only decay happening [6], the resulting particles are filtered and the muons are collected by magnets into a secondary beamline, called the πE5. This setup is shown in Fig. 2.1. The beamline then directs the $10^8 \mu$/s into the Mu3e experiments' solenoid magnet, where they are stopped by a double cone target and decay at rest.

2.2 Signal and Background Processes

The *signal event* we want to detect is the $\mu^+ \to e^+e^+e^-$ decay. As previously explained in Chapter 1 this decay is very rare and highly suppressed in the SM. Thus a lot of other decays and processes happen, constituting the measured background for this experiment [6]. Differentiating background from signal events is an important part of the detector concept. In this section we will introduce different possible background processes and how to differentiate them from the signal.

Decay	Approx. Branching Ratio	Reference
$\mu^+ \to e^+\nu_e\bar{\nu}_\mu$	100%	Michel decay
$\mu^+ \to e^+\gamma\nu_e\bar{\nu}_\mu$	1.4% (for $E_\gamma > 10$MeV)	Radiative decay [17]
$\mu^+ \to e^+e^-e^+\nu_e\bar{\nu}_\mu$	$3.4 \cdot 10^{-5}$	Internal conversion [11]

Tab. 2.1: List of the most frequent muon decay modes allowed by the standard model building the main sources of background particles in the experiment. The branching ratio describes the fraction of particles decaying in this decay mode.

2.2.1 Signal Event

First we define the characteristics needed to identify a signal event. In this decay all three resulting particles stem from the same decay. As such their point of origin coincides and is called the *event vertex*. The muon decaying at rest defines the combined momentum and energy of the signal event, since the laws of energy and momentum conservation dictate

$$E_{tot} = E_{e_0^+} + E_{e_1^+} + E_{e^-} = m_\mu = 105.7\frac{\text{MeV}}{\text{c}^2} \tag{2.2}$$

and

$$\mathbf{p}_\mu = \mathbf{p}_{e_0^+} + \mathbf{p}_{e_1^+} + \mathbf{p}_{e^-} = 0. \tag{2.3}$$

Using Eq. (2.2) and Eq. (2.3) as well as the knowledge of a single point of origin the signal event is well defined [6].

2.2.2 Background Processes

Due to the rarity of the signal, most electrons and positrons detected will not originate from the signal decay $\mu^+ \to e^+e^+e^-$. Instead, there are other more frequent processes and decays creating positrons end electrons which need to be differentiated from signal particles. The most notorious decays are listed in Table 2.1. These background sources can be separated into two groups, namely *internal conversion* and *combinatorial background*.

During the radiative decay a radiated photon may undergo internal conversion into an electron-positron pair, where the resulting particles look similar to the signal decay. Therefore, they are building one important background process, which is shown in Fig. 2.2 b) and Table 2.1. The created neutrinos by the radiative decay are not directly observable and the created electrons and positrons share the same point of origin. It can not be distinguished from the signal event by reconstructing the event vertex, therefore Eq. (2.2) and Eq. (2.3) need to be used instead. Neither

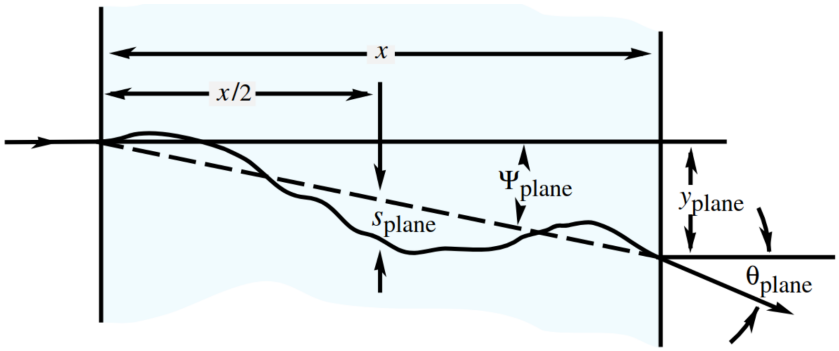


Fig. 2.3: A particle passing through a material gets scattered along the way, e.g. through Coulomb Scattering by nuclei. This sum of these phenomena is called Multiple Scattering. This picture shows the offset and kink angle that can be used to describe this.

Source: [38]

equation is satisfied, since some energy and momentum is taken up by the two neutrinos. Consequently a very good energy/momentum resolution is needed for the detector to properly distinguish internal conversion events from the signal.

The combinatorial background is the mix of superposition of events happening that produce $e^+e^+e^-$ triplets. The main source for positrons is the Michel decay, due to it being the most likely muon decay, as seen in Table 2.1. These positrons alone are not enough to build a fake signal event, but there are other possible sources for electron tracks, including, but not limited to, Bhabha scattering in the target material, creation of an electron-positron pair by the radiative decays' photon or Compton scattering in the detector material [6]. Combinations of these events could look similar to the signal, as seen in Fig. 2.2 c), however these tracks are not created by the same event and therefore do not share the same point or time of origin. For proper detection of the combinatorial background precise timing information and a good vertex reconstruction is needed.

2.3 The Mu3e Detector

In this section, we introduce the Mu3e detector, designed with a good momentum and vertex resolution, as well as precise timing information.

2.3.1 Concept

The design of the particle detector has to meet the performance requirements introduced in the previous section, involving multiple trade-offs. A high momentum

and vertex resolution is required, therefore the positional tracking capabilities of the particles passing through the detector is important.

When passing through material, a particle is deflected multiple times by the atoms resulting in an extra kink θ and offset $\mathbf{y}$ when leaving the material, as shown in Fig. 2.3. This process is called *Multiple Scattering* (MS) and the root mean square of the kink scales with the particles' momentum p

$$\theta \propto \frac{1}{p}\sqrt{\frac{x}{X_0}}, \tag{2.4}$$

with $\frac{x}{X_0}$ as the material's thickness in units of radiation length X_0 [22]. The particles created by the decaying muons have a low momentum and cannot exceed an energy of $53\frac{\text{MeV}}{\text{c}}$ [13]. Therefore the detectors' thickness, called *material budget*, has to be kept small. Gaseous detectors, like time projection chambers [13], are often used in MS dominated environments, but cannot handle the high particle rates needed to reach Mu3e's high sensitivity goals. Respecting this, monolithic pixel sensors are the best trade-off in accuracy, speed and material budget for Mu3e. Since precise timing information is needed as well, extra layers consisting of scintillating fibers and tiles are used in conjunction with the pixels [6].

In order to be able to measure the particle momenta, a magnetic field is required to curve their tracks. This curvature depends on their charge, where electrons are bent into a different direction than positrons, making it possible to differentiate their tracks. Hence, the whole detector is set in a 1T solenoidal magnetic field, designed to be as uniform as possible [6]. The strong magnet adds the extra benefit of keeping the track curves relatively small in diameter, resulting in the need for only a small detector along the radial axis.

For a successful track reconstruction at least three measurements need to be taken, while a good momentum measurement needs to correctly measure the curve of the particles' track. Therefore, multiple detector layers are needed. Due to the curved nature of the tracks, the sensors need to be arranged along the longitudinal axis of the tracks. This leads to the concept shown in Fig. 2.4 introducing three main detector parts, each in a cylindrical shape and consisting of multiple layers. The pixel layers are bundled in sets of two, to keep the MS error low between them [13].

In the central part of the detector one double pixel layer is used with a radius only slightly bigger than the target's radius, for a good vertex resolution. A second set of double layers is set with a bigger radius accompanied by a layer of scintillating fibers just below them. Using these four layers a preliminary particle track can be reconstructed.

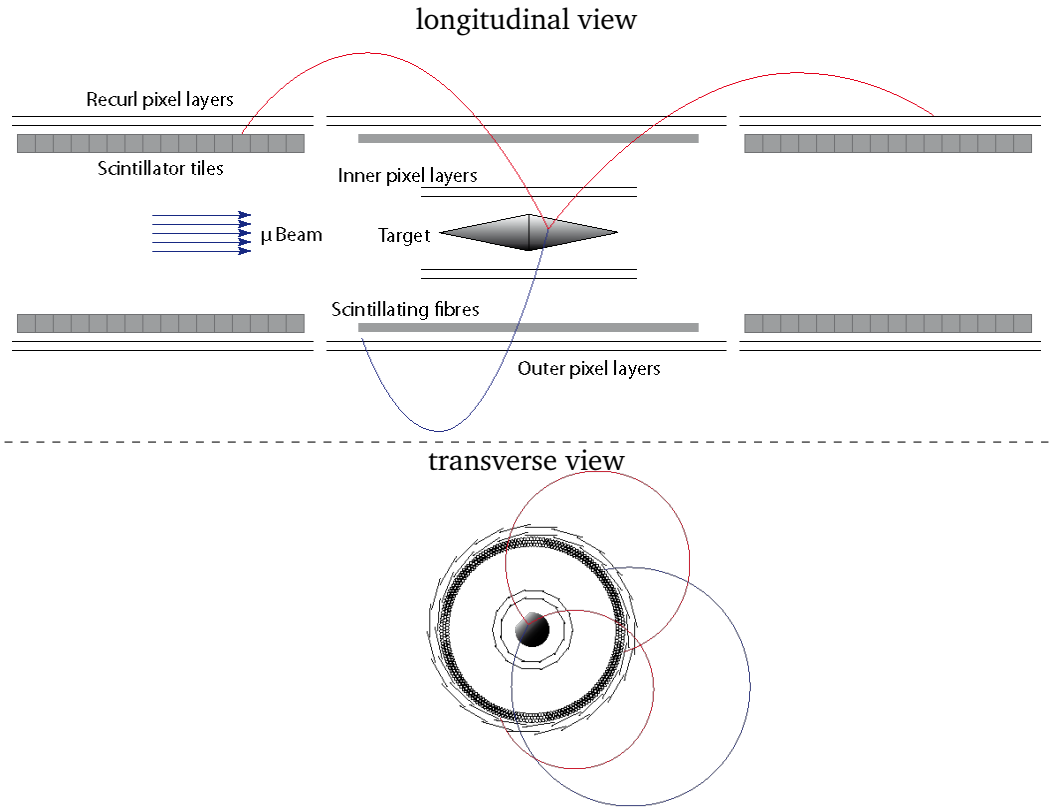


Fig. 2.4: Mu3e detector schematic cut along the longitudinal axis of the barrels, consisting of pixel, scintillating fibers and scintillating tile layers. The muon beam hits the target in the center where electrons (blue) and positrons (red) are created. These move on helical tracks through the multiple layers of the detector.
Source: [16]

The outer parts of the detector are used to detect particles returning back towards the cylinder axis. These are used to better estimate the tracks curve, increasing the momentum resolution. Here, only one set of the pixel layers is used, followed by scintillating tiles inside. For these *recurl stations*, scintillating tiles are used instead of fibers, introducing a higher material budget, since these are the last checkpoints used for reconstruction and have a better background suppression.

2.3.2 The MuPix Pixel Detector

Pixel detectors consist of semiconductor junctions of p- and n-doped regions, resulting in regions free of charge carriers inbetween them, called the depletion zone [19]. This results in an electric field from the p-doped region to the n-doped region. When a particle passes through the depletion zone it interacts with the atoms, creating new electron-hole pairs. These new free charge carriers move along the electric field, inducing a current measurable by readout electronics of the detector. Voltage can be applied to further increase the depletion zone size and the charges' speed, resulting in an increased signal size and better time resolution.

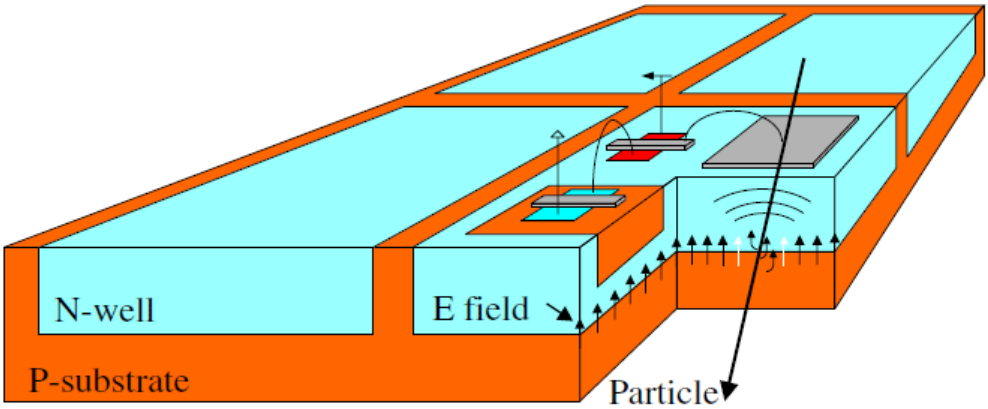


Fig. 2.5: Schematic of four pixels. For the front right pixel the on-chip electronics and a cross-section showing the electric field is shown.
Source: [16]

Active pixel sensors like this are no novelty and are used in other experiments, such as ALICE [8, 14]. Some of them managed to reduce the material budget by including the read-out electronic on the chip itself, but their time resolution is too low for Mu3e. Others have a good enough time resolution, like TIMEPIX [5], but are too thick for Mu3e to use. Consequently, a new pixel sensor, called *MuPix*, was designed [7]. The result is a *High Voltage Monolithic Active Pixel Sensor* (HV-MAPS), which is currently in its 10-th revision, called MuPix11 While it is still in development the final chip is planned to have a pixel size of $80 \times 80 \mu\text{m}$ on a $2 \times 2\text{cm}$ area and can be thinned down to $50\,\mu\text{m}$. The manufacturing process used for this chip allows for high voltages of up to 120V and a time resolution of below 11ns. In Fig. 2.5 the layout of a pixel and its associated electronic is shown. The major part of the chip is the pixel area, which is divided into multiple submatrices. These submatrices are connected downstream by a digital circuit used to evaluate and process pixel states.

In the detector these chips will be glued onto ladders mounted on cages for the layers, building an approximate cylinder for each station. When assembled, alignment of the pixel sensors will be determined on a per chip basis for precise measurements. Possible deformations of the cages happening throughout the beamtimes will be accounted for using an online alignment system [37].

2.3.3 Scintillating fibers

Scintillating fibers are used in the central part of the detector for their precise timing measurements and low material budget. When a charged particle passes through them, it interacts with the material, exciting electrons into a higher, but volatile energy level. The electrons quickly return to a de-excited state, emitting light in visible

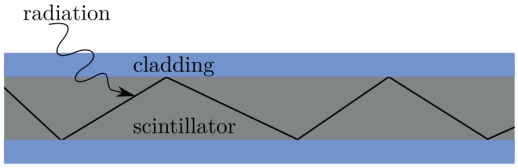


Fig. 2.6: A particle passing through scintillating material excites electrons. The photon created during de-excitation can be guided by reflection to one of the fibers ends. Source: [24]

wavelengths. The scintillating material is coated in a carrier material, which guides the emitted light through internal reflection to one of its ends, seen in Fig. 2.6. At a fibers' end silicon photomultipliers amplify the light on arrival into a measurable electrical charge, read out by the MuTrig system [15].

The fibers are required to have a sensitivity of close to 100% and a time resolution below 0.5ns, for sufficient background suppression of tracks not reaching the recurl stations.

2.3.4 Scintillating Tiles

In the recurl stations scintillating tiles instead of fibers are used. They work similar to the fibers, but since they are located at the end of the particles tracks their material budget has no constraints, besides the tight space they have to fit in. They consist of plastic scintillator split into multiple small tiles, ordered along the inner part of the cylinder. The scintillating tiles are required to have a sensitivity of 100% and a time resolution of less than 100ns for efficient detection of particle triplets and reducing background signals [6].

2.4 Data Acquisition

Data collected by the detector is compressed, sorted and filtered by the multi-layered *data acquisition system* (DAC), which we will introduce in this section. The Mu3e detector has no trigger system, therefore the data is collected and processed in a datastream. This stream is processed in multiple layers before being fed to the mass storage servers for offline processing and analysis. An overview of the scheme and parts involved is shown in Fig. 2.7.

Directly after the sensors, still inside the magnetic field, 114 *Front-end Boards* (FEB), equipped with of AriaV FPGAs are used [13]. The data collected from the pixel sensors is not ordered in time, due to the read-out method. Thus, the FEB's sort the pixel data according to their timestamps. For the fiber detector readout clusters of

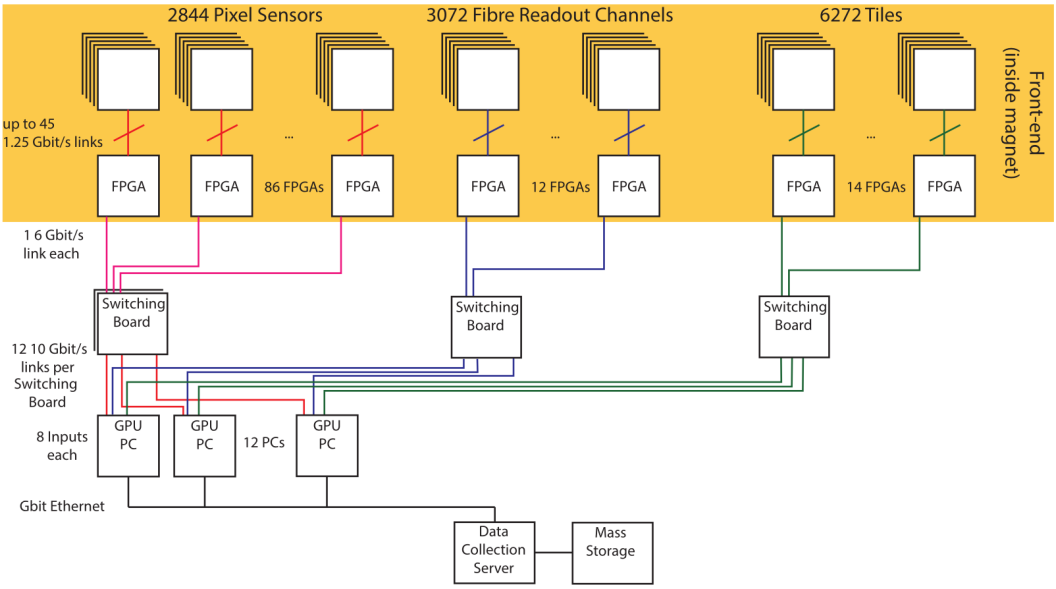


Fig. 2.7: Overview of the Mu3e data acquisition chain.

Source: [6]

fiber hits are formed. The FEB's then collect the data in packages sending them via a 6Gbit/s optical link to the next layer, the *Switching Boards*. These four Switching Boards are used to align and merge data from the different detector parts into single datastreams and bundle them into timeslices [28].

After the data has been merged, it is sent to the final layer of the DAC, the *Filter Farm*, which is used to filter the datastream, removing unimportant data and forwards the rest to the mass storage servers. The filter farm consists of 12 computers daisy-chained together. Each computer receives data using a DE5a-Net FPGA board [6]. The DE5a-Net boards task is to receive the detector data and communicate with the software running on their computer using *direct memory access* (DMA). The software uses GPUs for online processing of the data, deciding which data seems important enough to save for mass storage. This filtering process is the focus of this thesis. It is built on the results of D. vom Bruch [13] and will be introduced and discussed in the rest of this thesis. Finally, the filtered data is sent over a 1Gbit/s Ethernet connection to the mass storage servers, where it is stored for further analysis [6].

2.4.1 Data Rates

The main goal for the filter farm is to reduce the incoming data rate to meet slow storage speeds and reduce the cost of storage. First we will estimate the amount of data collected per second for the $1 \cdot 10^8 \mu$/s rate. Using the detector simulation available, an average of $1056 \cdot 10^6$ hits/s are estimated, ignoring pixel noise [13]. As

an upper limit for this calculation we are using the MuPix7 noise rate of 0.1 Hz/pixel resulting in ≤ 18Mhz noise rate for the 2844 pixel sensors involved. Each hit coming from the pixel sensors is encoded as a 32 bit word using 8b/10b encoding [41], resulting in 40bit per hit. These words include the hit address on the sensor, a time stamp and amplitude information. Adding the pixel sensor information adds 12bit, but the extra data needed is balanced by bundling hits into timeslices. Together this results in a data rate of 43Gbit/s for the pixel data.

The fiber detectors are read out with a relatively low threshold, resulting in a high dark count rate of approximately 70% of particle hits [13]. This rate is reduced by building clusters of on average 3 coinciding signal hits, which are stored in 28bit words, including the number of hits, side of the fiber, photomultiplier channels and a coarse timestamp. For each of the hits a precise timing information is saved using 7 bits resulting in approximately 50bits per cluster. Again using 8b/10b encoding this results in a data rate of 26.3Gbit/s for the fiber detectors.

The final part consists of the tile detectors in the recurl stations, where the lowest number of hits will be detected, resulting in a lower data rate [6]. Consisting of much more scintillating material than the fibers, a higher threshold is used for the tile detectors. This results in a lower dark count rate and an average of 180Mhz of hits for all tile detectors [13]. The time stamp, tile number and time over threshold are stored in 64 bit words, resulting in an estimated data rate of 11.6Gbit/s for the tile detectors.

The total data rate for all detectors is therefore expected to be around 80.9Gbit/s. Thus, the filter farm needs to reduce the rate by a factor of 100.

3 Online Event Selection

In this chapter the algorithm used for the *Online Event Selection* is introduced. It is used in the detectors *Filter Farm*, see Section 2.4, to reduce the collected data by a factor of at least 100 in real-time.

The concept is based on the thesis of D. vom Bruch [13] and divided into three steps:

1. *Selection Cuts*: A simple filter cutting away most infeasible triplet combinations for the track reconstruction.
2. *Track Reconstruction*: A triplet based reconstruction and classification of the estimated particle tracks.
3. *Vertex Reconstruction*: A simplified reconstruction of possible event vertices. For this, all combinations of two positrons and one electron tracks are investigated, searching for a possible singular vertex, which fulfills the signal characteristics defined in Section 2.2.

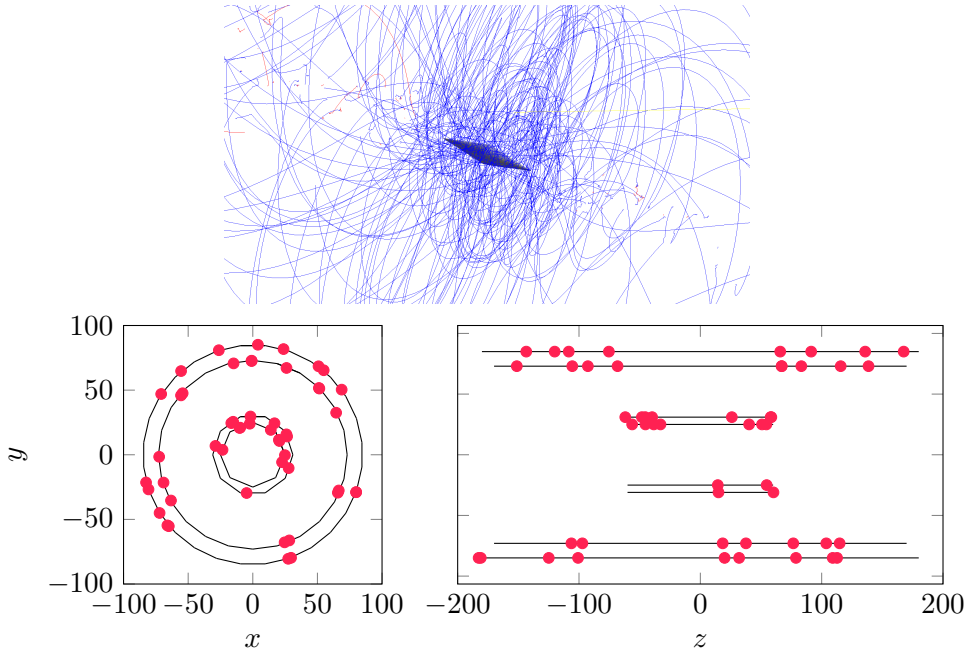


Fig. 3.1: Depictions of a simulated frame. Top: 3D render of the target and simulated tracks [16]. Bottom: Detected hits in the detector layers, projected onto the transverse (bottom-left) and longitudinal (bottom-right) plane. The y-position for the longitudinal plane are projected onto their layer.

To accommodate for the time resolution the detected hits are bundled into timeslices, called a *frame*. Each frame is a snapshot of hits detected in a timeframe of 64ns[1]. Due to the stringent performance requirements of real-time filtering the problem size is reduced by only using the four central pixel layers for the online selection. An example frame can be seen in Fig. 3.1. Reducing the amount of detector layers used for reconstruction reduces problem size and momentum resolution. Since the online selection process is only used to reduce the datarate as much as possible and a full offline reconstruction is done at a later stage, this reduction is acceptable and shown to be good enough [13].

Tests of the algorithm are performed and evaluated on Monte Carlo data collected from a Geant4 [2] simulation of the full Mu3e detector.

3.1 Mathematical Notation

We define the coordinate system using the magnetic fields' direction. Its axis is set along the longitudinal axis of the detector, designed to be as homogenous as possible. Along this axis the z-axis of the cartesian coordinate system is defined. The transverse plane perpendicular to the magnetic field defines the x- and y-axes.

We will use the following notation in this chapter:

- $\mathbf{x} \in \mathbb{R}^3$ denotes a vector, with $x = |\mathbf{x}|$.
- $\mathbf{h_i}$ is the position of a hit in the i-th layer.
- $\mathbf{x_t}$ is the transverse projection of the vector $\mathbf{x}$, so only the x and y coordinates are used.
- $\mathbf{h}_{ij} = \mathbf{h}_i - \mathbf{h}_j$.
- $\mathbf{h}_{i,z}$ is the z-coordinate of the vector $\mathbf{h}_i$.
- $\hat{\mathbf{h}}$ denotes a unit vector.
- $\hat{\mathbf{e}}_z$ is the unit vector in the direction of the z-axis.

3.2 Helical Tracks

Mu3e only directly observes charged particles inside a solenoidal magnetic field [6]. The motion of these particles is described using a helix. In this section the particle track, its parametrization and its relation to physical parameters will be presented.

[1] This number is based on 3σ uncertainty of the pixel detectors estimated time resolution and may change for the final MuPix design used in the detector.

A charged particle moving through a magnetic field, in a vacuum devoid of any electrical field is determined by the *Lorentz Force* [19]:

$$\mathbf{F} = \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = m\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2} = q\mathbf{v} \times \mathbf{B}, \tag{3.1}$$

where

- $\mathbf{B} = B\hat{\mathbf{e}}_\mathbf{z}$ is the magnetic field with field strength B,
- q the particles signed charge,
- $\mathbf{v} \in \mathbb{R}^3$ the particle's velocity,
- m is the particles mass,
- $\mathbf{x}$ the position of the particle,
- and natural units with $\hbar h = c = 1$ are used.

Therefore, the Lorentz Force only acts perpendicular to the magnetic field and to the particles x- and y-movement, while the z-direction is unaffected. Assuming a constant magnetic field, no other forces acting on the particle and no energy loss the absolute momentum p of the particle, with $\mathbf{p} = m\gamma\mathbf{v}$, is conserved [23].

We can rewrite Eq. (3.1) using only geometric quantities, by replacing the time t dependence with the path length $s(t)$, which is the distance traveled along the particles' trajectory. This is done using the relation $\frac{\mathrm{d}s}{\mathrm{d}t} = v$ [20]:

$$\frac{\mathrm{d}\mathbf{x}(s)}{\mathrm{d}t} = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\frac{\mathrm{d}s}{\mathrm{d}t} = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s}v \tag{3.2}$$

$$\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2} = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s}v = \frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}s^2}v^2 \tag{3.3}$$

Inserting these two equations into Eq. (3.1) results in

$$\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}s^2} = \frac{qB}{mv^2}(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \times \hat{\mathbf{e}}_z) = \frac{qB}{mv}(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s} \times \hat{\mathbf{e}}) \tag{3.4}$$

$$\Rightarrow \quad \frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}s^2} = \frac{qB}{p}(\hat{\mathbf{p}} \times \hat{\mathbf{e}}_z). \tag{3.5}$$

The normalized momentum vector $\hat{\mathbf{p}}$ is the tangent, pointing in the direction the particle moves. This transformation is favorable, since the detector has a better spatial than temporal resolution [23].

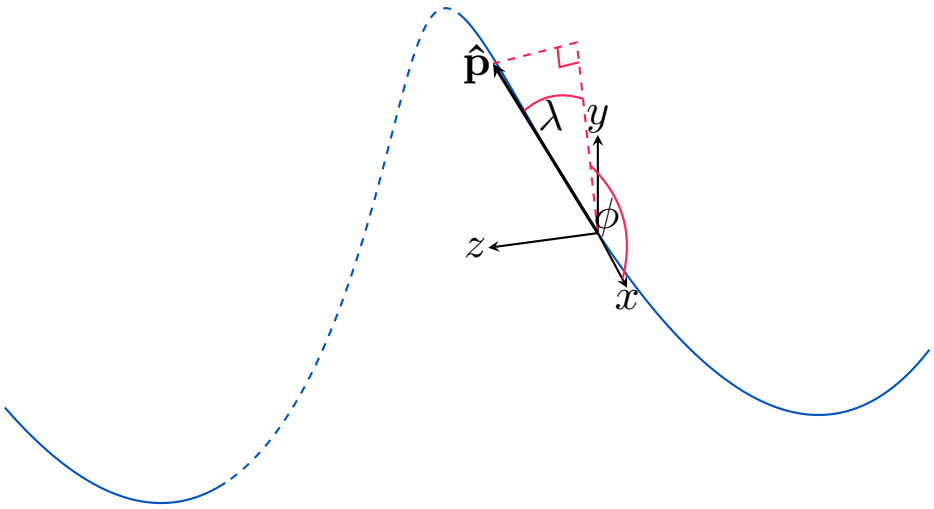


Fig. 3.2: Schematic of parameters used for describing a particles' helical track (blue). The parameters are shown for the helix when passing through the coordinate systems origin. $\hat{\mathbf{p}}$ describes the tangent, λ the angle between the tangent and the xy-plane and ϕ the angle of to the x-axis.

The solution to the three differential equations in Eq. (3.5) is a helix, which can be described by the equation

$$\mathbf{x}(s) = \mathbf{h}_0 + \frac{p}{qB}\begin{pmatrix}\sin\Phi\\ -\cos\Phi\\ \Phi\tan\lambda\end{pmatrix} = \mathbf{h}_0 + \frac{1}{\kappa}\begin{pmatrix}-\sin\Phi\\ \cos\Phi\\ -\Phi\tan\lambda\end{pmatrix}, \tag{3.6}$$

using 6 parameters [20]. These parameters, as shown in Fig. 3.2, are correlated and defined by [20]

$$\Phi(s) = \tfrac{s}{R}\cos\lambda, \tag{3.7}$$

$$\lambda = \arcsin\frac{\mathrm{d}z}{\mathrm{d}s} \in (-\tfrac{\pi}{2}, \tfrac{\pi}{2}), \tag{3.8}$$

$$R = -\frac{p}{qB}, \tag{3.9}$$

and the initial hit position $\mathbf{h}_0$.

We can now split the helical motion into two planes, the transverse plane defined by the x- and y-axes and the z, s-plane. The helix projected onto the x, y plane is a circle with curvature

$$\kappa_t = \kappa\cos\lambda. \tag{3.10}$$

In the z, s plane the helix is a linear function, described by

$$z(s) = \mathbf{h}_{0,z} + \sin\lambda = \frac{\Phi(s)}{\kappa}\sin\lambda. \tag{3.11}$$

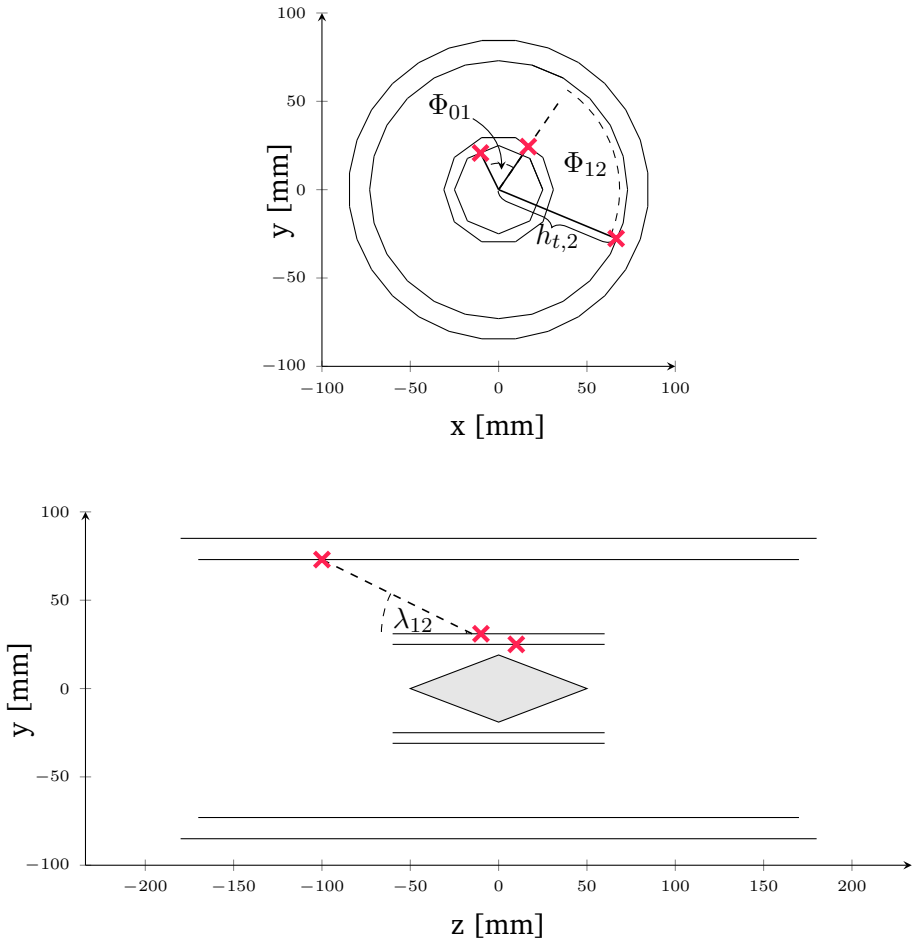


Fig. 3.3: Sketch of Selection Cuts and the geometric quantities used. The red crosses are hits from a triplet combination of the first three detector layers.
Source: Based on [13]

3.3 Selection Cuts

The track reconstruction uses a triplet of hits from the first three layer as seed for its first step. There are $n_0 \cdot n_1 \cdot n_2$ combinations possible, where n_i (with $i \in \{0, 1, 2\}$), is the number of detected hits in layer i. Since particles and noise may generate more than one hit, only $N_{\text{true}} \leq \min(n_0, n_1, n_2)$ true tracks are available in the frame. Performing a full track reconstruction on all these combinations, knowing only a fraction is real, is computationally inefficient. As a consequence we introduce Selection Cuts, a filter consisting of four relatively simple calculations for removing over 95% of all combinations, while keeping over 99% of triplet combinations from true tracks. We improve upon the algorithm introduced by D. vom Bruch [13], by analyzing and modifying the filters used, improving the cut away combinations by over 1.5%, while decreasing the computational cost.

The basis for these filter steps is set by the observation that created particles always have low momenta. Consequently, we can study simple geometric quantities and their behaviour on true tracks. The quantities used for all filter steps are shown in Fig. 3.3.

We start by observing the angle between hits in neighbouring layers [13]

$$\cos \Phi_{ij} = \frac{\mathbf{h}_{t,i} \cdot \mathbf{h}_{t,j}}{h_{t,i} h_{t,j}}, \qquad i \in 0, 1, j = i + 1. \tag{3.12}$$

This angle is used as a rough indicator for the path length traversed between hits, similar to Eq. (3.7). It is important to note that this is only a rough estimate, because Eq. (3.7) is based on the circle center of the transverse track, while Eq. (3.12) is based on the origin of the coordinate system.

On simulated data (Fig. 3.4 a) and c)) we can still see a clear drop-off for true track combinations, while the angles are more evenly distributed for all triplet combinations. Thus, we can choose the cuts pictured by the dashed black line as threshold for the angles, cutting away over 60% of possible combinations over both angles. We have to be careful, when choosing the cut values. Each false track cut saves us expensive computations in the next steps of the Online Event Selection. But cutting away true combinations may cut away true tracks belonging to a signal event.

Eq. (3.12) can be simplified even further. While the detector layers are no perfect cylinders, they attempt to resemble one. As consequence we can save the computation of the $h_{t,i}$'s by replacing them with the mean radii $\bar{r}_{t,i}$ of all points, precalculated on simulation data. Using these removes the need for the square roots involved in calculating $h_{t,i}$, a multiplication, as well as one division.

$$\bar{\Phi}_{ij} = \underbrace{\bar{r}_{t,i} \cdot \bar{r}_{t,j} \cdot \cos \hat{\Phi}_{ij}}_{=\text{const.}} = \mathbf{h}_{t,i} \cdot \mathbf{h}_{t,j}. \tag{3.13}$$

Using these simplifications the tail ends of the distributions lose their clear cut, but drop off more slowly (Fig. 3.4 b) and d)), allowing for the use of a minimum and maximum value for $\hat{\Phi}_{01}$. This reduction in computation leads to similarly good results, with $\sim 40.7\%$ of all tracks cut away and $\sim 99.7\%$ true combinations kept in the simulation for $\cos \hat{\Phi}_{01}$ compared to $\sim 40.5\%$ cut away from all and 99.9% true track combinations kept by Eq. (3.12). To conclude Eq. (3.13) is chosen instead of Eq. (3.12).

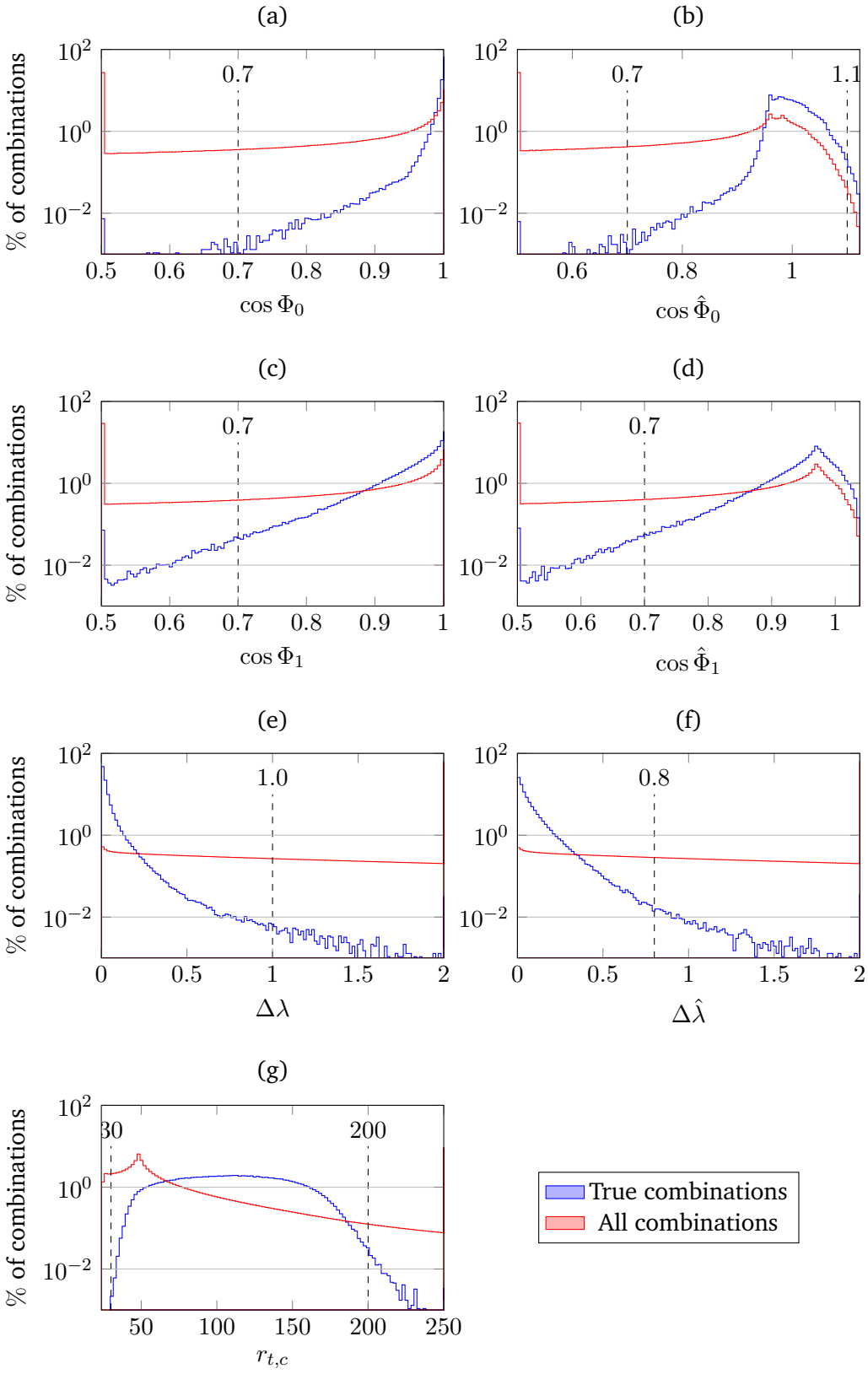


Fig. 3.4: Histograms for the values specified on the x-axis, showing the percentage of combinations for each bin. The dashed black line represents the value this quantity is cut at. On the left side the original cuts [13] are shown and used, while on the right side the improved versions introduced in this section are used.

As third filter we use the slope λ between neighbouring layers along the z-direction

$$\tan \lambda_{ij} = \frac{z_j - z_i}{h_{t,j} - h_{t,i}}, \qquad i \in \{0, 1\}, j = i + 1. \tag{3.14}$$

Similar to Eq. (3.12) these are only rough estimations for the track helix parameters, in this case the dip angle λ. For the filter we are not interested in $\tan \lambda_{ij}$, but in the difference

$$\Delta\lambda = \tan \lambda_{12} - \tan \lambda_{01}.\text{[13]} \tag{3.15}$$

The distributions for true and all track combinations can be seen in Fig. 3.4 e), with clear differences in their shape. This allows to cut away $\sim 82.75\%$ of combinations, while keeping 99.65% of true combinations.

We can observe for Eq. (3.15) a similar behaviour to Eq. (3.12) regarding the $h_{t,i}$. Using the same constant radii $r_{t,i}$ we can simplify Eq. (3.15) to

$$\begin{aligned} \Delta\bar{\lambda} = \underbrace{(\bar{r}_{t,2} - \bar{r}_{t,1}) \cdot \Delta\hat{\lambda}}_{=\text{const.}} &= z_2 - z_1 - (z_1 - z_0) \underbrace{\frac{\bar{r}_{t,2} - \bar{r}_{t,1}}{\bar{r}_{t,1} - \bar{r}_{t,0}}}_{=\text{const.}} \\ &= z_2 - z_1 - (z_1 - z_0)\Delta\hat{r}_{\text{ratio}}. \end{aligned} \tag{3.16}$$

Results for this simplification and the chosen cut are shown in Fig. 3.4 f).

As one final filter step we use the radius $r_{t,c}$ of the circle defined by the three hits in the transverse plane [13]. This circle is clearly defined and will also be needed as basis for the track reconstruction shown in the next section. The radius $r_{t,c}$ is defined by

$$r_{t,c} = \frac{d_{01}d_{12}d_{20}}{2[(\mathbf{h}_0 - \mathbf{h}_1) \times (\mathbf{h}_2 - \mathbf{h}_1)]_z}, \tag{3.17}$$

with $d_{ij} = |\mathbf{h}_i - \mathbf{h}_j|$ [10]. The distribution and chosen threshold values of $r_{t,c}$ are shown in Fig. 3.4 g). Using $r_{t,c}$ as filter is computationally more expensive, while also having less impact, cutting only away 19% of all combinations. Since $r_{t,c}$ is needed for the track reconstruction, we can offset the computational cost by storing it for chosen hit triplets, to be reused in the track reconstruction. Therefore, $r_{t,c}$ is chosen as last filter, cutting away $\sim 1\%$ more percent from all possible combinations.

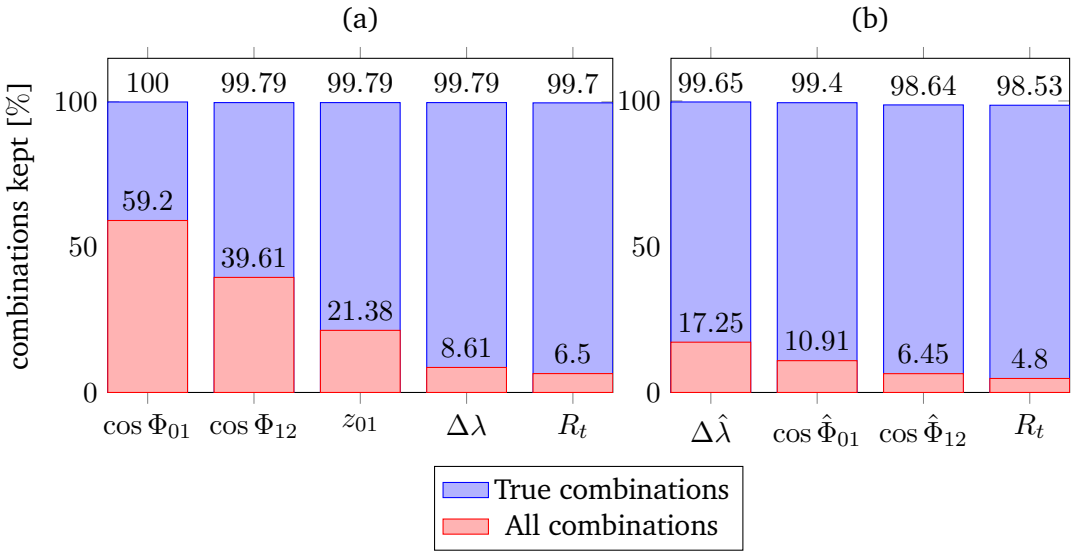


Fig. 3.5: Comparison between new cuts chosen (b) and cuts from D. vom Bruch's work [13] (a) performed on simulation data. The graphs show how many combinations are kept (y-axis) after each step (x-axis). The final results of both methods are similar. We cut away 1.7% more combinations using our method (b), while only cutting away $\sim 1.2\%$ of additional true combinations. All while needing less and cheaper computations.

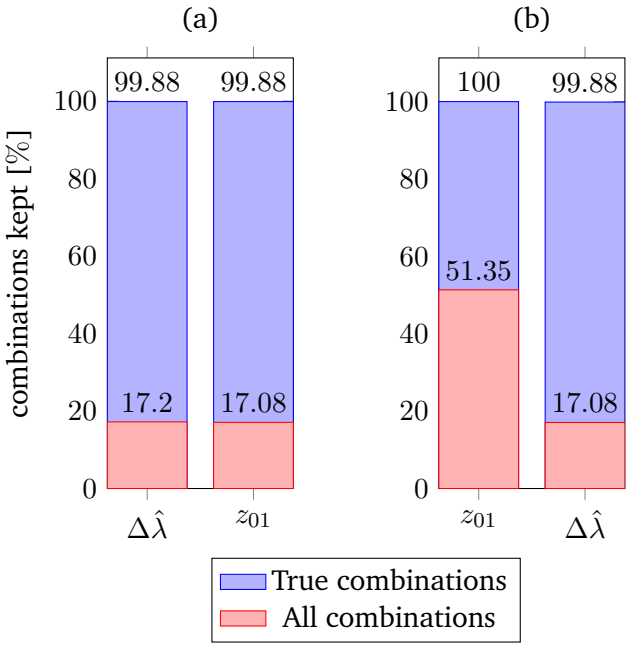


Fig. 3.6: Simulation showing correlations between consecutive hits. Although Eq. (3.18) cuts away almost 50% from all combinations (b), performing this cut after Eq. (3.16) (a) yields only an extra of 0.12% combinations removed.

Applying these filters after each other in the order shown in Fig. 3.5 b) results in $\sim 4\%$ of all combinations left while keeping $\sim 99\%$ of true combinations. These filter remove over 1.5% extra combinations compared to the filters and cuts previously suggested by D. vom Bruch [13]. The previous version does not use the mean radii $\bar{r}_{t,i}$ and adds one more filter

$$z_{10} = z_1 - z_0. \tag{3.18}$$

As seen in Fig. 3.6 b), applying Eq. (3.18) first can remove almost 50% of combinations, with Eq. (3.16) applied afterwards reduces it to 17.08% kept. But changing the order shows that Eq. (3.16) alone already reduces the combinations to 17.2%, making Eq. (3.18) as filter almost obsolete, removing only 0.12% additional combinations. So we decided to skip this step. These observations also show the importance of the order that the cuts are applied in.

We have chosen Eq. (3.16) as first filter, removing the largest amount of combinations, quickly reducing the problem size to a fifth using only 4 subtractions and one multiplication. As second and third step Eq. (3.13) is used for both angles. Since both have the same computational cost, we are using the angle between the first and second layer first, having a greater impact. As last filter the transverse radius $r_{t,c}$ of the circle solution is used, being the computationally most expensive operation, it is used only on a small fraction of the initial problem.

3.4 Track Reconstruction

Starting from the previously selected triplets full, particle tracks need to be reconstructed. For the reconstruction the *Triplet Fit* algorithm [26, 10] is used, which is designed for MS dominated environments, like the Mu3e experiment. The Triplet Fit is based on the assumption that in each detector layer the particle gets scattered, deviating from its previous path, following the curve of a new helix. As such between each pair of detector layers a new helical track is assumed, with the same curvature, but with a slight kink in its direction. Therefore, combinations of hits in three consecutive layers are used, where a kink in the central hit is assumed, and the two helices are reconstructed, as shown in Section 3.4.1. This algorithm neglects the error in hit position, introduced by the size of the pixel sensors pixel. To refine the reconstructed tracks multiple consecutive hit triplets are used and combined, reducing the overall reconstruction error, described in Section 3.4.2. For the online reconstruction only the four inner pixel layers are used, while the full offline reconstruction uses all available pixel layers.

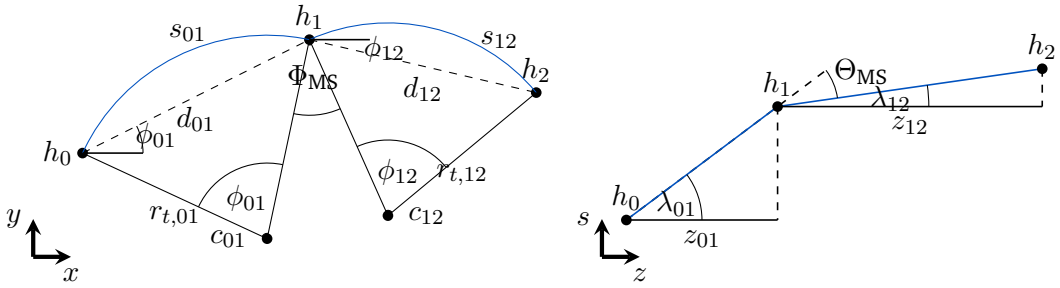


Fig. 3.7: Sketch of a reconstructed track going through three hits h_0, h_1, h_2 with a kink from MS in the central hit. Geometric quantities involved in the Triplet Fit are shown.

Source: Based on [10]

Multiple Scattering is also considered by other, more commonly used algorithms like the *Kalman-Filter* [4, 3, 33, 21] and *Broken-Lines-Fit* [23, 12]. The Kalman-Filter uses an iterative approach for reconstruction, making it a bad fit for the performance goals for the online reconstruction. On the other hand the Broken-Lines-Fit requires a reference track as starting point and then refines the track using a compute heavy matrix inversion [12]. For the reference track a rough track reconstruction needs to be done, e.g. using the Triplet Fit algorithm, therefore resulting in extra work. A detailed comparison of the Triplet-Fit and the Broken-Lines-Fit is performed by M. Kiehn [23].

3.4.1 Single Triplet Fit

The hit triplet used consists of measured pixel detector hits interpreted as space points, assumed to have negligible spatial uncertainties. When passing through the central layer the particle gets scattered resulting in a kink in its trajectory. Assuming no momentum loss and thus a constant curvature κ, this trajectory change is described by a sudden change in the track angles ϕ and λ, called Φ_{MS} and Θ_{MS} respectively. An example trajectory including the variables used in this section is shown in Fig. 3.7.

The distribution of the scattering angles Φ_{MS} and Θ_{MS} is assumed to be Gaussian with a width which can be approximated using the *Highland formula* [22, 27, 13]

$$\bar{\Theta}_{\text{MS}} = \bar{\Phi}_{\text{MS}} = 0, \tag{3.19}$$

$$\sigma^2_{\text{MS}} = 13.6\text{MeV}\frac{|q|}{\beta cp}\sqrt{\frac{x}{X_0}}\left[1 + 0.038\ln\frac{x}{X_0}\right], \tag{3.20}$$

$$\sigma^2_{\Theta} = \sigma^2_{\text{MS}}, \tag{3.21}$$

$$\sigma^2_{\Phi} = \frac{\sigma^2_{\text{MS}}}{\sin^2\lambda}, \tag{3.22}$$

where q is the particles' charge, β the relativistic velocity, c the speed of light and $\frac{x}{X_0}$ the traversed material in units of radiation length. Using the correlation $\kappa \propto \frac{1}{p}$, we define our objective as finding the minimal scattering angles for a constant curvature κ using the *method of least-squares* [10]

$$\chi^2(\kappa) = \frac{\Phi_{\text{MS}}(\kappa)^2}{\sigma_\Phi^2} + \frac{\Theta_{\text{MS}}(\kappa)^2}{\sigma_\Theta^2}. \tag{3.23}$$

We can assume $\frac{\mathrm{d}\sigma_{\text{MS}}}{\mathrm{d}\kappa} = 0$, since only weak MS occurs [10]. Using this approximation the minimum of Eq. (3.23) can be found by solving its normal equation

$$\begin{aligned}\frac{d\chi^2}{d\kappa} &= \frac{d\Phi_{\text{MS}}(\kappa)}{d\kappa}\frac{\Phi_{\text{MS}}(\kappa)}{\sigma_\Phi^2} + \frac{d\Theta_{\text{MS}}(\kappa)}{d\kappa}\frac{\Theta_{\text{MS}}(\kappa)}{\sigma_\Theta^2} \\ &= \frac{\sin^2\lambda}{\sigma_{MS}^2}\cdot\frac{\mathrm{d}\Phi_{\text{MS}}(\kappa)}{\mathrm{d}\kappa}\Phi_{\text{MS}}(\kappa) + \frac{1}{\sigma_{MS}^2}\cdot\frac{\mathrm{d}\Theta_{\text{MS}}(\kappa)}{\mathrm{d}\kappa}\Theta_{\text{MS}}(\kappa) \overset{!}{=} 0.\end{aligned} \tag{3.24}$$

Alternatively to κ the 3D radius $R = \frac{1}{\kappa}$ could be used as argument for Eq. (3.23). While the idea is the same, the derived final equations differ. We chose κ for better numerical stability and less complex computations for the final formulas.

The scattering angle Φ_{MS} is defined as [23]

$$\Phi_{\text{MS}}(\kappa) = \phi_{12} - \phi_{01} - \frac{\Phi_{01}(\kappa) + \Phi_{12}(\kappa)}{2}, \tag{3.25}$$

using the angles ϕ_i and the transverse bending angles $\Phi_i(\kappa)$ for $i \in \{01, 12\}$. This notation for the index i is used for the rest of the section.

To derive a formula for the propagation angles $\Phi_i(\kappa)$ we use the relations

$$r^2 = r_{t,i}^2 + \frac{z_i^2}{\Phi_i^2} \tag{3.26}$$

$$\text{and} \qquad r_{t,i} = \frac{d_i}{2}\cdot\frac{1}{\sin\frac{\phi_i}{2}}. \tag{3.27}$$

Inserting Eq. (3.27) into Eq. (3.26) results in the transcendental equation for $\Phi_i(\kappa)$ [23]

$$\begin{aligned}\sin^2\frac{\Phi_i}{2} &= \frac{d_i^2}{4r^2} + \frac{z_i^2}{r^2}\frac{\sin^2\frac{\Phi_i}{2}}{\Phi_i^2} \\ &= \frac{d_i^2\kappa^2}{4} + z_i^2\kappa^2\frac{\sin^2\frac{\Phi_i}{2}}{\Phi_i^2}.\end{aligned} \tag{3.28}$$

This equation has no algebraic solution and thus care has to be taken which solution is chosen [10]. Analogous the longitudinal scattering angle Θ_{MS} is defined by

$$\Theta_{\text{MS}}(\kappa) = \lambda_{12}(\kappa) - \lambda_{01}(\kappa), \tag{3.29}$$

using the relation between λ_i and Φ_i [23]

$$\Phi_i(\kappa) = \frac{z_i}{R \sin \lambda_i(\kappa)} = \frac{z_i \kappa}{\sin \lambda_i(\kappa)}. \tag{3.30}$$

The non-linearity of Eq. (3.28) and Eq. (3.30) results in a non-linear objective function Eq. (3.23). Solving this directly would require complex numerical computations. So instead we linearize the problem around a suitable solution using a *first order Taylor expansion* [10]. Based on Eq. (3.19) the scattering angles are generally low, and therefore we can use the solution $\Phi_{\text{MS}} = 0$ as suitable starting point. This solution describes a circle in the transverse plane through all three triplet points with radius $r_{t,c}$. For this solution $r_{t,01} = r_{t,12} = r_{t,c}$ applies, which is already used and calculated by Eq. (3.17) during the Selection Cuts. It is important to note that this solution is in general not the physically correct solution, since this solution would result in different three-dimensional radii for the two helices, and thus it would violate the assumed momentum conservation [23].

Using the circle solution as base, the Taylor expansion for the angles Φ_i and λ_i is defined as

$$\Phi_i(\kappa) = \Phi_{i,c} + \frac{\mathrm{d}\Phi_i}{\mathrm{d}\kappa}\Big|_{\kappa_{i,c}} (\kappa - \kappa_{i,c}), \tag{3.31}$$

$$\text{and} \qquad \lambda_i(\kappa) = \lambda_{i,c} + \frac{\mathrm{d}\lambda_i}{\mathrm{d}\kappa}\Big|_{\kappa_{i,c}} (\kappa - \kappa_{i,c}). \tag{3.32}$$

The variables denoted by a c in their index are derived from the circle solution. $\Phi_{i,c}$ can be calculated using trigonometry by

$$\begin{aligned} \sin \frac{\Phi_{i,c}}{2} &= \frac{d_i}{2r_{t,c}} \\ \Leftrightarrow \Phi_{i,c} &= 2 \arcsin \frac{d_i}{2r_{t,c}}. \end{aligned} \tag{3.33}$$

While this equation has multiple solutions, we are only interested in the first revolution of the helix. Therefore, we assume that $\frac{\Phi_{i,c}}{2} \in (-\pi, \pi)$. Using Eq. (3.33) and insert it into Eq. (3.28) gives us a relation for the three-dimensional radii $r_{i,c}$

$$r_{i,c}^2 = r_{t,c}^2 + \frac{z_i^2}{\Phi_{i,c}^2} = \frac{1}{\kappa_{i,c}^2} \tag{3.34}$$

Similarly, we can derive the bend angles $\lambda_{i,c}$ for the circle solution from Eq. (3.30)

$$\lambda_{i,c} = \arcsin \frac{z_i}{\Phi_{i,c} r_{i,c}}. \tag{3.35}$$

The derivatives are defined by [25]

$$\left.\frac{\mathrm{d}\Phi_i}{\mathrm{d}\kappa}\right|_{\kappa_{i,c}} = \frac{\Phi_{i,c}}{2\kappa_{i,c}} \cdot \alpha_i \tag{3.36}$$

$$\text{and} \qquad \left.\frac{\mathrm{d}\lambda_i}{\mathrm{d}\kappa}\right|_{\kappa_{i,c}} = -\frac{\sin\lambda_{i,c}}{\kappa_{i,c}} \cdot \beta_i \tag{3.37}$$

using the coefficients

$$\alpha_i = \frac{1}{1-\delta_i}, \tag{3.38}$$

$$\beta_i = \frac{\delta_i}{1-\delta_i}, \tag{3.39}$$

$$\text{and} \qquad \delta_i = \cos^2(\lambda_{i,c}) \cdot \left(1 - \frac{\frac{\Phi_{i,c}}{2}}{\tan\frac{\Phi_{i,c}}{2}}\right). \tag{3.40}$$

With these linearized expressions we can now derive solutions for Φ_{MS} and Θ_{MS} as a function of the track curvature correction $\Delta\kappa = (\kappa - \kappa_c)$:

$$\begin{aligned}\Phi_{\text{MS}}(\kappa) &= \phi_{12} - \phi_{01} - \frac{\Phi_{01,c} + (\kappa - \kappa_{01,c})\frac{\mathrm{d}\Phi_{01,c}}{\mathrm{d}\kappa}}{2} - \frac{\Phi_{12,c} + (\kappa - \kappa_{12,c})\frac{\mathrm{d}\Phi_{12,c}}{\mathrm{d}\kappa}}{2} \\ &= \Phi_{MS,c} + \frac{(\kappa - \kappa_{01,c})\frac{\mathrm{d}\Phi_{01,c}}{\mathrm{d}\kappa}}{2} + \frac{(\kappa - \kappa_{12,c})\frac{\mathrm{d}\Phi_{12,c}}{\mathrm{d}\kappa}}{2},\end{aligned} \tag{3.41}$$

$$\begin{aligned}\Theta_{\text{MS}}(\kappa) &= \lambda_{12,c} + (\kappa - \kappa_{12,c})\frac{\mathrm{d}\lambda_{12}}{\mathrm{d}\kappa} - \lambda_{01,c} - (\kappa - \kappa_{01,c})\frac{\mathrm{d}\lambda_{01}}{\mathrm{d}\kappa} \\ &= \Theta_{MS,c} + (\kappa - \kappa_{01,c})\frac{\mathrm{d}\lambda_{01}}{\mathrm{d}\kappa} - (\kappa - \kappa_{12,c})\frac{\mathrm{d}\lambda_{12}}{\mathrm{d}\kappa},\end{aligned} \tag{3.42}$$

with their derivatives

$$\frac{\mathrm{d}\Phi_{\text{MS}}}{\mathrm{d}\kappa} = \frac{1}{2} \cdot \left(\frac{\mathrm{d}\Phi_{01,c}}{\mathrm{d}\kappa} + \frac{\mathrm{d}\Phi_{12,c}}{\mathrm{d}\kappa}\right), \tag{3.43}$$

$$\text{and} \qquad \frac{\mathrm{d}\Theta_{\text{MS}}}{\mathrm{d}\kappa} = \left(\frac{\mathrm{d}\lambda_{01}}{\mathrm{d}\kappa} - \frac{\mathrm{d}\lambda_{12}}{\mathrm{d}\kappa}\right). \tag{3.44}$$

These can now be used to solve the normal equation Eq. (3.24) for κ, resulting in [25]

$$\begin{aligned}\kappa &= \frac{\sin^2\lambda\left(\Phi_{MS,c} + \kappa_{01,c}\frac{\mathrm{d}\Phi_{01,c}}{\mathrm{d}\kappa} + \kappa_{12,c}\frac{\mathrm{d}\Phi_{12,c}}{\mathrm{d}\kappa}\right) + \Theta_{MS,c} + \kappa_{01,c}\frac{\mathrm{d}\lambda_{01}}{\mathrm{d}\kappa} - \kappa_{12,c}\frac{\mathrm{d}\lambda_{12}}{\mathrm{d}\kappa}}{\sin^2\lambda \cdot \left(\frac{\mathrm{d}\Phi_{01,c}}{\mathrm{d}\kappa} + \frac{\mathrm{d}\Phi_{12,c}}{\mathrm{d}\kappa}\right)^2 + \left(\frac{\mathrm{d}\lambda_{01}}{\mathrm{d}\kappa} - \frac{\mathrm{d}\lambda_{12}}{\mathrm{d}\kappa}\right)^2} \cdot \frac{1}{\sigma^2_{MS}} \\ &= \frac{\sin^2\lambda\left(\Phi_{MS,c} + \kappa_{01,c}\frac{\mathrm{d}\Phi_{01,c}}{\mathrm{d}\kappa} + \kappa_{12,c}\frac{\mathrm{d}\Phi_{12,c}}{\mathrm{d}\kappa}\right) + \Theta_{MS,c} + \kappa_{01,c}\frac{\mathrm{d}\lambda_{01}}{\mathrm{d}\kappa} - \kappa_{12,c}\frac{\mathrm{d}\lambda_{12}}{\mathrm{d}\kappa}}{\sigma^2_\kappa} \cdot \frac{1}{\sigma^2_{MS}}.\end{aligned} \tag{3.45}$$

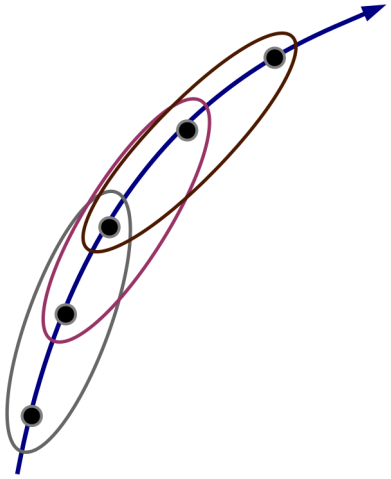


Fig. 3.8: Sketch on how triplets are grouped. Each black dot represents a hit and the brown ellipsis represents a triplet group.

Source: [35]

Using the results of these calculations, χ^2 from Eq. (3.23) can be calculated and used as a measure to describe the quality of the fit, where closer to 0 is better.

3.4.2 Triplets Fit

Incorporating more than three hits in the algorithm for a full track fit requires to accommodate for multiple triplets in the objective function. Each consecutive triplet uses the last two hits of the previous triplet as first two and adds a hit from the next layer as third hit. This schema is pictured in Fig. 3.8. The goal is to find one global curvature for all triplet combinations minimising the MS angles for each triplet. This results in the new, global objective function [13]

$$\chi^2_{\text{global}}(\kappa) = \sum_{t}^{n_{\text{triplets}}} \chi^2_t(\kappa). \tag{3.46}$$

The scattering angles $\Phi_{MS,t}$ and $\Theta_{MS,t}$ for each triplet are independent of the other triplets. Consequently, each χ^2_t is minimised individually, and the global curvature is defined as the weighted average [13]

$$\bar{\kappa} = \frac{\sum_t^{n_{\text{triplets}}} \frac{\kappa_t}{\sigma^2_{\kappa,t}}}{\sum_t^{n_{\text{triplets}}} \frac{1}{\sigma^2_{\kappa,t}}}. \tag{3.47}$$

Adding more hits, and thus triplets increases the momentum resolution [10]. For the Online Event Selection only the first four layers are used, thus two triplets need to be fit. First the helix for the combinations chosen by the Selection Cuts is fit. Using this preliminary helix the hit position in the fourth layer is estimated. The

estimated point is used for finding the closest fourth layer hit, which is then used to build the second triplet and perform a fit. Finally, only tracks with a χ^2 error of smaller than 32 are kept, resulting in 94% of true tracks being kept.

For the tracks kept, the track parameters are calculated, and the tracks are classified as electrons and positrons, depending on the sign of their global curvature κ, as described in Section 3.2.

3.5 Vertex Fit

Having possible tracks for a frame reconstructed, we now want to check if the frame looks like a signal event has happened. Keeping the computational costs low, we define our goal to not have an exact reconstruction, but rather having a rough estimate of a possible spatial event vertex. Each track triplet consisting of two positron tracks and one electron track is analyzed and checked for a possible event signature, which is defined in Section 2.2. The concept behind the algorithm is to first reduce the problem complexity by finding a possible event vertex in the transverse plane and only if such a vertex was found to project it back into the third dimension. Back in three dimensions the possible vertex is checked for signal compatibility. If a compatible vertex is found the track will be kept for storage and offline analysis [13].

Before a possible event vertex is searched for, we start by calculating the total energy of all particles in the triplet using their curvature κ. The total energy for all three particles using the relativistic kinetic energy $E_{\text{kin}} = (\gamma - 1)m$ and natural units is defined as

$$E_{\text{tot}} = \sum_{i=0}^{3} E_i = \sum_{i=0}^{3} (\sqrt{p_i^2 + m_e^2} - m_e). \tag{3.48}$$

If the total energy of all particles, including some margin of error, differs too much from the muons rest energy Eq. (2.2) is violated. Thus, the triplet is no signal triplet and no further processing needs to be done. Otherwise, the tracks are now processed and checked for a possible vertex.

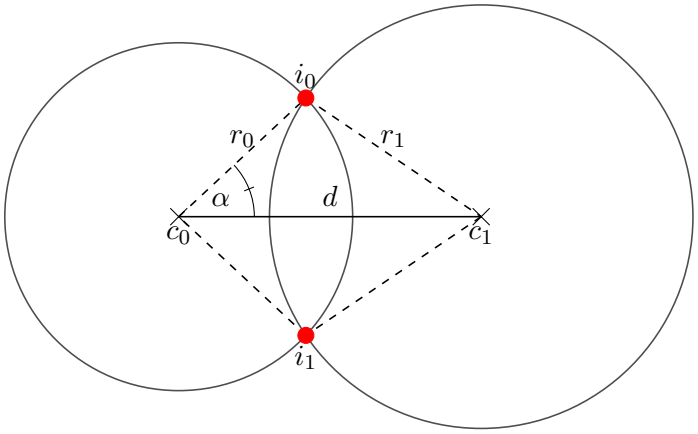


Fig. 3.9: Sketch of two circles intersecting, with the variables needed for calculating.

3.5.1 Finding Possible Event Vertices

Next the problem complexity is reduced by searching for a vertex in two dimensions, namely the transverse plane. The transverse projection of a helix is a circle, as described in Section 3.2. A circle is defined by its center

$$\mathbf{c} = \mathbf{h_{t,0}} + \begin{pmatrix} \sin\phi \\ \cos\phi \end{pmatrix} \cdot r_t, \tag{3.49}$$

and its radius

$$r_t = \frac{1}{\kappa_t}, \tag{3.50}$$

using the same parameters as in the previous sections.

Next all circle-circle intersections are calculated, by first calculating each intersection for each pair of circles, as seen in Fig. 3.9. Two different circles may intersect in one single degenerate point, two distinct points or not at all in $\mathbb{R}^2$ [40]. This happens, if the following two equations hold

$$|c_1 - c_0| = d \leq |r_1 - r_0|\,, \tag{3.51}$$

$$d \geq r_1 + r_0. \tag{3.52}$$

When the circles do intersect, the points of intersection have to be calculated. The degenerate case with only one intersection point happens only if the distance is $d = r_1 + r_0$. For both intersection points, the triangles $\triangle c_0 i_0 c_1$ and $\triangle c_0 i_1 c_1$ are symmetric, meaning the opening angle α is the same for both triangles. Using the cosine rule for triangles the angle can be calculated with

$$\alpha = \arccos\left(\frac{(r_0 + r_1 + d)^2}{2 r_0 d} - 1\right). \tag{3.53}$$

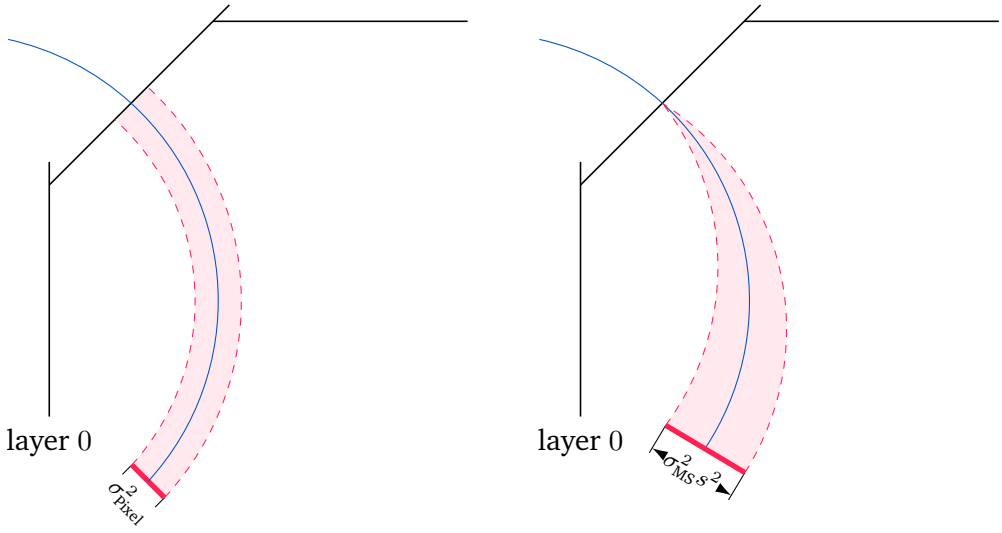


Fig. 3.10: Comparison of pixel error (left) and MS error (right) at some point after passing through a pixel layer. The pixel error remains constant, while the MS error increases per path length traveled.

One more angle is needed, the slope of d

$$\beta = \arctan(\frac{d_x}{d_y}). \tag{3.54}$$

Using β, α and $\mathbf{c_0}$ we can calculate the intersection points using

$$i_{0/1} = \mathbf{c_0} + r_0 \cdot \cos(\beta \pm \alpha). \tag{3.55}$$

We are only interested in intersections close to the target, represented by a disk with a radius of 19mm. Therefore, all intersections are thrown away that are farther away than 6mm, respecting uncertainties. All intersections left are then used for further analysis, wherre each triplet combination consisting of one intersection per track pair are used.

The event vertex is specified by a specific point, while the intersection points, in general, describes a triangle. We define the two-dimensional event vertex as weighted mean μ_t of the intersection points $p_i, i \in \{0, 1, 2\}$ weighted by their uncertainties σ_i^2 [13]

$$\mu_t = \frac{\sum_{i=0}^{3} \frac{p_i}{\sigma_i^2}}{\sum_{i=0}^{3} \frac{1}{\sigma_i^2}}. \tag{3.56}$$

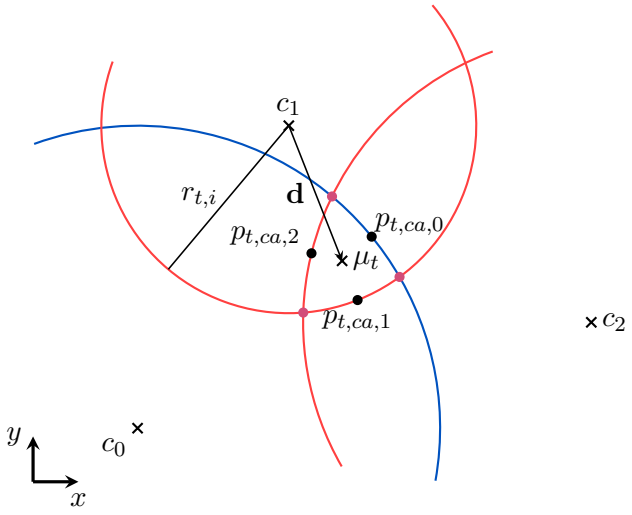


Fig. 3.11: The track transverse points for a vertex are estimated by calculating the weighted mean of three circle-circle intersections μ_t. Then the points of closest approach $p_{t,ca,i}$ (black) for each circle are found by extending the distance vector $\mathbf{d}_i$ to the circles hull.

Source: Based on [13]

Since the first layer is wrapped closely around the target, the intersection points may be relatively close to the detector layer, where the pixel error dominates over the MS induced error, as shown in Fig. 3.10. Therefore, the uncertainties for each point are defined by both, the pixels spatial resolution σ^2_{Pixel} and the tracks σ^2_{MS} error [13]

$$\sigma_i^2 = \sigma_{MS}^2 \cdot s_i^2 + \sigma_{\text{Pixel}}, \tag{3.57}$$

with s_i as the path length along the circle from the measured detector hit in layer 0.

Next we want to find the point corresponding to the event vertex. This point is defined as the point of closest approach $p_{ca,i}$ to the mean position on each track, calculated as shown in Fig. 3.11

$$p_{t,ca,i} = \frac{\mathbf{d}_i}{d_i} \cdot r_{t,i} \tag{3.58}$$

$$\text{with} \qquad \mathbf{d} = \bar{\mu}_t - \mathbf{c}. \tag{3.59}$$

This circle point is now projected back onto the 3D helix track. To achieve this the angle $\Delta\Phi$ traveled from the initial hit $\mathbf{h}_{i,t}$ to $p_{t,ca,i}$ is calculated and used to find the corresponding z-position on the helix

$$z_i = \mathbf{h}_{i,z} - \frac{\Delta\Phi \sin\lambda_{01}}{\kappa}. \tag{3.60}$$

Again the error is calculated, following Eq. (3.57). Using all three z-positions the mean μ_z is calculated and used as z-coordinate for our estimated event vertex. The possible signal vertex position μ for this triplet combination is defined using μ_z and μ_t.

3.5.2 Signal Estimation

Using all points calculated in the previous section, we now calculate the distance from each closest helix point to the vertex position μ and calculate its error [13]

$$\chi^2 = \sum_{i=0}^{3} \frac{\mu_i}{\sigma_i^2}. \tag{3.61}$$

Among all vertices found from all track triplets we are using only the one with the smallest χ^2 error. If the smallest error is bigger than a threshold of 800, it is discarded as well. Otherwise, we test the proximity to the target surface for the chosen vertex. The surfaces radius is approximated as linear slope along the beam line mirrored at the origin

$$r_{\text{target}}(z) = 50mm - \frac{19mm}{50mm} \cdot |z|\,, \tag{3.62}$$

where the targets half-length is 50mm and its maximal radius 19mm. Thus, the distance of the point to the targets surface is defined as

$$d_\mu = \mu_t - r_{\text{target}}(\mu_z). \tag{3.63}$$

Event vertices farther away than 20mm are discarded. Otherwise, the total momentum of all tracks at the points of closest approach is estimated. If the vectorial momentum is too high it violates momentum conservation as defined in Section 2.2 and the frame is discarded. In all other cases the frame is kept.

4 Implementation and Testing

In this chapter we will present the implementation of the Online Event Selection algorithm. The algorithm is implemented on GPUs using the *CUDA Programming Model* (CUDA) [29]. While our implementation is based on D. vom Bruch's work [13], her version implemented only the track and vertex reconstruction steps. We are adding our improved version of the Selection Cuts and improve performance and accuracy for the other two steps.

The Online Event Selection is going to run on 12 computers. With a timeframe size of 64ns, one computer has to evaluate over $1.302 \cdot 10^6$ frames per second. We will show in this section that our implementation satisfies these requirements for phase I on a 1080Ti.

We start this chapter by introducing CUDA basics, followed by implementation details. Finally we present and discuss our test results.

4.1 The CUDA Programming Model

CUDA is an API, provided by NVIDIA, used to create applications, running on NVIDIA GPUs. GPUs were initially designed for massively parallel tasks needed in graphics processing, such as pixel coloring for high resolutions or simultaneous transformations for thousands of models, needed to render 3D scenes. With CUDA we can utilize this hardware for general purpose computing [30].

We will first introduce the basic concept behind the CUDA programming model. The main building block are functions executed on the GPU, called *kernels*. Each kernel is executed by a set amount of *threads* and is called by a host, which is usually the CPU. These threads are executed on *streaming multiprocessors* (SM), which execute the same instructions for 32 threads at once. This grouping is called a *warp*. When encountering branching paths in the program, threads inside a warp follow different instructions. Each branch has to be executed separately. This *thread divergence* slows down computation, since the SM can only utilize a part of its computational power at once [34, 30].

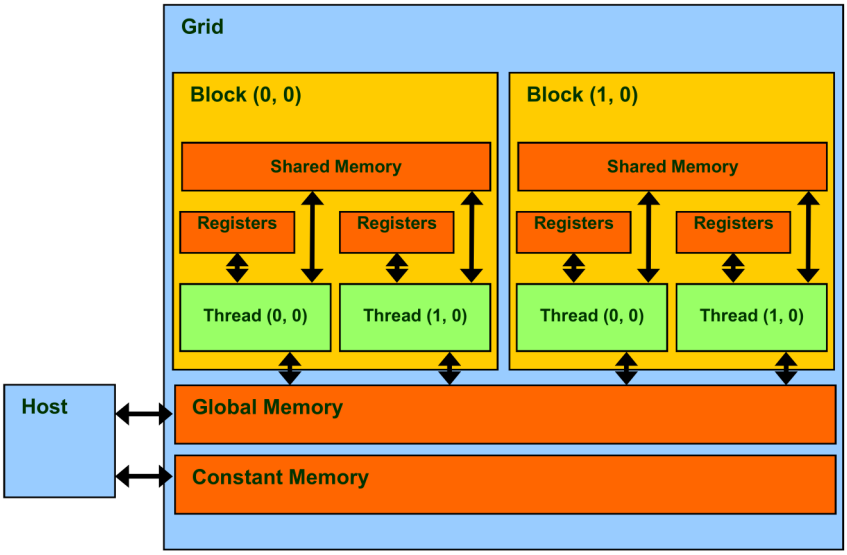


Fig. 4.1: Overview of the cuda threading and memory hierarchy.

Source: [30]

CUDA threads are organized in a hierarchy. Multiple threads are grouped into equally shaped blocks, where each thread has a unique identifier inside its block. These blocks are then grouped together in a grid, where each block also has its unique identifier. When a kernel is launched by the host, the grid and block dimensions are specified. The number of threads and blocks also is generally data and task-dependent and should exceed the number of processors available on the GPU. This leads to high occupancy and when a thread waits for an expensive operation, such as memory access to complete, other threads can be executed during that time.

CUDA threads have access to data from multiple memory spaces, as outlined in Fig. 4.1. Each thread has its own private register memory, which is rather small but fast. Additionally, all threads inside a block share a memory space, *shared memory*, similar in speed to registers. Moreover, all threads from all blocks share one *global memory* space. This memory is the largest, but rather slow. It profits highly from *coalesced memory access*, where each thread in a warp reads a consecutive memory address and therefore the whole read operation can be grouped into a single transaction. One more available memory type is *constant memory*, also shared by all threads. Constant memory can only be written to by the host, but is rather fast, when all threads inside a warp access the same location.

Launching a CUDA kernel blocks the host and forces all CUDA calls to be sequentially executed. But often memory transfer is slow and can be split into multiple chunks, resulting in time wasted, while waiting, when the GPU could have started with computations. For situations like these *streams* exist. Streams allow for tasks

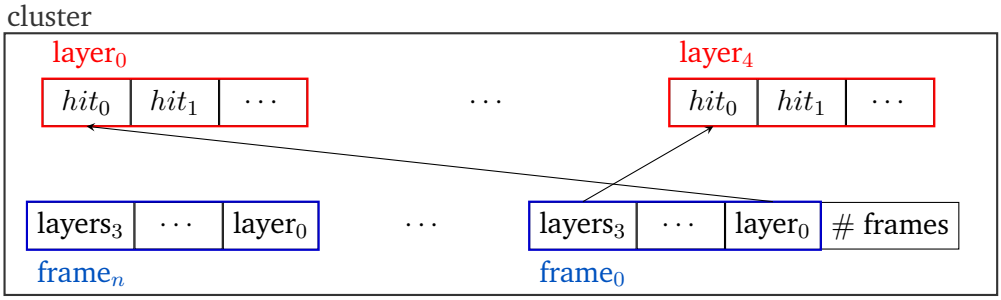


Fig. 4.2: Memory layout for the Online Event Selection. Each cluster of frames consists of two arrays, one filled with hits and the other with pointers to layer start positions. The hits are grouped into layers, while the pointers are grouped into frames.

to be performed concurrently. Using this technique data can be split up into multiple chunks and when one chunk has finished transferring, one stream can start working, while the next chunk is getting transferred.

4.2 Global Memory layout

The global memory layout for the algorithm is set by the communication with the FPGA inside the farm PCs. As data arrives on the FPGA, it is bundled into chunks of multiple frames. These chunks then are transferred to the CPUs main memory using *direct memory access*. Ideally no further processing is done by the CPU and the data is directly transferred to the GPU.

Therefore, the FPGA gathers the frames in chunks, which are already aligned for the GPUs coalesced memory access pattern. Due to frames varying in size, the chunk size is not directly set by the number of frames. Having to provide all hit information, with their layer and frame associations, all in one chunk, results in a specific memory layout built by the FPGA.

Chunks are build from the start and end at the same time. The last 4 bytes are reserved for the final number of frames inside the chunk. On arrival a frames hit data is added to an array at the beginning of the chunk, sorted by their corresponding layers. When a new layer is started its address in the array is added to a reversed array at the end of the chunk, where all start positions for all layers of all frames are stored. If the next frame does not fit into the chunk anymore, the number of frames stored is stored in the last 4 bytes, the chunk is sent to CPU main memory and a new chunk is started. This memory schema is shown in Fig. 4.2.

4.3 Parallelization of the Algorithm

In this section we introduce how the algorithm outlined in Chapter 3 is parallelised. Taking a look at the dependencies between the different steps, a few observations can be made:

- Each frames computation is independent of other frames.
- Each step is directly dependent on the results of the previous step.
- Each hit triplet/track/track triplet is independent of one another.

We use these observations as guideline for the parallelisation of the algorithm. First we divide the biggest chunks, the frames, among the CUDA blocks, choosing a cyclic distribution, even though some variance in the frames' complexity is given. Tests with a dynamic distribution, where each block takes the next unprocessed frame, were performed, and no performance difference was measured.

The outline of the main kernel is shown in Listing 1, with details for each step following in the next sections.

Listing 1 Code outline for the main kernel used for the Online Event Selection.

```
__global__ void main_kernel(Frames *frames_global) {
  __shared Frame frame;
  while (frames_global.get_next(frame)) {
    selection_cuts(frame);
    __syncthreads();
    track_reconstruction(frame);
    __syncthreads();
    vertex_fit(frame);
    __syncthreads();
    frame.save_to_global(frames_global);
  }
}
```

4.3.1 Loading a Frame

During processing of the frame the hit data needs to be accessed frequently. Therefore, we load the frame into shared memory, increasing access performance. Due to the high variance in hits per frame and layer and the resulting exponential increase in complexity, we skip a frame and mark it for storage directly, if there are to many hits across all layers.

Respecting coalesced memory access each thread loads the data from a consecutive memory address. CUDA allows reading from memory in 4, 8 and 16 byte chunks with one operation, while a hit, consisting of three floats, has a size of 12 bytes. We compared the performance of explicit 4, 8 and 16 byte loads, to loading hit by hit, where the compiler decides which operation is used. No measurable difference in performance was noticed, since the actual data loaded for a frame is relatively small. Thus, the hit by hit version was chosen.

4.3.2 Selection Cuts

Listing 2 Selection Cuts Outline

```
__device__ bool selection_cuts(Frame &frame) {
  const size_t max_combs =
      frame.layer_hits[0] *
      frame.layer_hits[1] *
      frame.layer_hits[2];
  /// Cyclic iteration over all combinations
  for (size_t comb_idx = threadIdx.x;
        i < max_combs;
        i += blockDim.x) {
    /// Calculate hit indices
    const size_t i0 = comb_idx /
        (frame.layer_hits[1] * frame.layer_hits[2]);
    const size_t i1 = (comb_idx /
        frame.layer_hits[2]) % frame.layer_hits[1];
    const size_t i2 = comb_idx
        % frame.layer_hits[2];

    const hit h0 = frame.layer[0][i0];
    const hit h1 = frame.layer[1][i1];
    const hit h2 = frame.layer[2][i2];

    if (test_phi0(h0, h1))
      if (test_phi1(h1, h2))
        if (test_delta_lambda(h0, h1, h2))
          if (test_R_t(h0, h1, h2))
            frame.add_combination(i0, i1, i2);
  }
  return frame.keep;
}
```

During Selection Cuts we need to iterate over all possible triplet combinations and check for each combination if they meet the cut criteria defined in Section 3.3. Problems arise at the possible branching point after each check, where thread divergence may arise. We could counteract this by performing each cut over a combination chunk first, before continuing to the next cut. By doing so we introduce

the need for synchronization between each chunk and possibly each cut, costing performance.

The alternative way, which we chose, is performing all cuts on a triplet combination using a cyclic distribution among all threads, as seen in Listing 2. Using Fig. 3.5 from Section 3.3 as reference, we sorted the filters this way, so only a fraction of each combination is kept after each cut. Therefore, the probability of a thread in a warp diverging, by keeping the combination, is kept low. Additionally, with the selection cuts being kept short and low on computations, the duration for a diverged thread is kept low.

A combination passing all cuts is kept and stored in shared memory for processing in the next step. If more combinations than a set maximum are found, the frame is seen as to complex for online reconstruction, marked for storage and all further processing for the frame is skipped.

4.3.3 Track Reconstruction

Listing 3 Track Reconstruction Outline

```
__device__ bool track_reconstruction(Frame &frame) {
  // Cyclic iteration over all combinations
  for (size_t comb_idx = threadIdx.x; comb_idx < frame.num_combs;
       comb_idx += blockDim.x) {
    // Retrieve hit indices calculated by Selection Cuts
    const auto i0 = frame.combinations[comb_idx][0],
               i1 = frame.combinations[comb_idx][1],
               i2 = frame.combinations[comb_idx][2];
    const auto &hit0 = frame.layer[0][i0],
               &hit1 = frame.layer[i][i1],
               &hit2 = frame.layer[2][i2];

    Track track = fit_first_triplet(hit0, hit1, hit2);

    const auto hit3_estimated = track.estimate_layer3_hit();
    const auto hit3 = frame.closest_layer3_hit(hit3_estimated);

    track.fit_fourth_hit(hit3);
    if (track.chi2 > TRACK_CHI2_MAX)
      continue;

    frame.add_track(track);
  }
  return frame.keep;
}
```

Each track can be reconstructed independently, with the reconstruction itself being rather expensive. The reconstruction itself does not provide many opportunities for parallelisation, but has a constant runtime with only a very low amount of possible branches. These branches are used for edge cases to increase numerical stability. Consequently, each thread performs a full reconstruction for a triplet combination, where the triplets are distributed cyclic across all threads. An outline of the steps used is shown in Listing 3.

After the reconstruction, as described in Section 3.4, is performed, a track is only kept if the χ^2 error is low enough. If kept, the track parameters needed for the vertex reconstruction are calculated and stored for further processing. Similar to the previous step, if too many tracks are found, the complexity for the vertex reconstruction is deemed to high and therefore the frame is marked for storage.

4.3.4 Vertex Reconstruction

We split the vertex reconstruction into two steps, shown in Listing 4. The first part is described by lines 7 to 20, where we first iterate over all triplet combinations. Since all combinations only have to be checked once, independent of their order, we divide this step into two loops. One loop over all positron-electron pairs and one loop over all additional positrons. We parallelise over the first loop in a cyclic distribution. The second loop decreases in size for each increase of `idx_pos0`, introducing possible thread divergence. To reduce the amount of divergence, we first iterate over all positron indices. This leads to delayed increase of `idx_pos0`, resulting in similar amount of iterations for the second loop for threads in a warp.

If to many track triplets were found, we keep the frame and return, otherwise possible event vertices are reconstructed in the second step. This step is outlined in lines 27 to 49 of Listing 4 and follows the steps explained in Section 3.5. Here each thread takes one track triplet and performs a full vertex reconstruction, leaving early if another thread has found a possible vertex.

First all circle-circle intersections are calculated, then a vertex position is estimated. If the error is small enough, the distance to the target is evaluated using Eq. (3.63). Is the vertex close enough the momentum at the points of closest approach is tested for signal compatibility, with some margin of error. A track triplet passing all these tests is seen as a possible event vertex, and therefore the frame is selected, and any further computations for this frame are stopped at their next branching points.

Listing 4 Vertex Reconstruction Outline

```
__device__ void vertex_reconstruction(Frame &frame) {
  const size_t num_combs = frame.num_el_tracks * frame.num_pos_tracks;
  Track tracks[3];
  size_t idx_pos0, idx_pos1, idx_el;

  // Gather triplets not violating energy conservation
  while (frame.get_next_track_indices(idx_pos0, idx_el) && !track.keep) {
    tracks[0] = frame.get_el_track(idx_el);
    tracks[1] = frame.get_pos_track(idx_pos0);
    for (size_t idx_pos1 = idx_pos0 + 1;
         idx_pos1 < frame.num_pos_tracks && !frame.keep; idx_pos1++) {
      tracks[2] = frame.get_pos_track(idx_pos1);
      if (test_total_energy(tracks) || frame.keep)
        continue;

      // Automatically adds triplet to list and marks frame for storage
      // if to many triplets were found.
      frame.add_track_triplet(idx_el, idx_pos0, idx_pos1);
    }
  }
  __syncthreads();

  if (frame.keep)
    return;

  // Full vertex reconstruction starting here
  while (frame.get_next_track_triplet(idx_pos0, idx_pos1, idx_el) &&
         !frame.keep) {
    Intersection intersections[3][2];
    intersections[0] = get_intersections(tracks[0], tracks[1]);
    intersections[1] = get_intersections[tracks[1], tracks[2]);
    intersections[2] = get_intersections[tracks[2], tracks[0]);

    Vertex vertex_estimate = get_vertex_estimate(intersections);

    if (vertex_estimate.chi2 > CHI2_MAX || frame.keep)
      continue;

    if (vertex_estimate.calc_target_dist() > TARGET_DIST_MAX
        || frame.keep)
      continue;

    if (vertex_estimate.calc_total_momentum(tracks)
        > MOMENTUM_MAX || frame.keep)
      continue;

    frame.keep = true;
    return;
  }
}
```

4.4 Online Monitoring

During runtime of the Mu3e detector it is important to be able to check if everything is working correctly or if differences to simulations or test runs arise. Therefore, monitoring systems need to be created to notice and react to issues occurring. One example system is the online alignment for the Mu3e detector [37], which monitors the detector positions, making it possible to react to small changes in the alignments between sensors.

Monitoring for the Online Event Selection is implemented by creating histograms of important data at multiple points of the algorithm. These histograms can be used for tracking different detector data, based on the reconstructed tracks or compare the spectrograms with other physics theories, searching for new physics [13]. Additionally, differences to the simulation or test runs can be detected, allowing to adjust parameters, while the detector is running. These histograms are copied back from the GPU in regular intervals and send to the *Maximum Integrated Data Acquisition System* (MIDAS) at PSI, used for displaying and storing monitoring data [6].

Each block builds its own histograms in shared memory. This limits the histograms size, since shared memory is limited and the algorithm needs quite a lot for buffering triplets and tracks between the steps. But it also avoids issues of global memory, where access is slow in general, and slowed down even further by multiple threads possibly accessing the same location. After a block is finished with its assigned frames, it adds its values to its counterpart in global memory. This allows for coalesced memory access, improving the global memory access times.

An example histogram gathered during simulation is shown in Fig. 4.3, showing a good checkpoint for monitoring. We propose to monitor following data:

- The momentum of reconstructed particles, used as energy spectrometer of particles present in the detector [13].
- The number of kept frames ordered by reason.
- Vertex distance to target.
- Momentum and energy distributions for event vertices and their particles.

4.5 Benchmarks

We will evaluate the performance of our implementation in this section and compare it to the reference implementation by D. vom Bruch [13]. It is shown that our

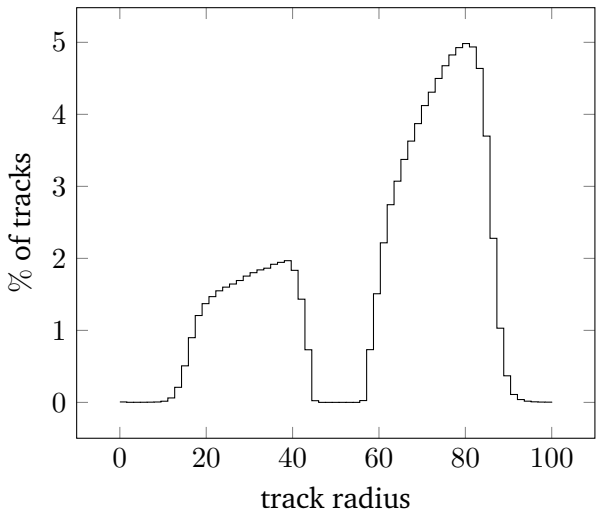


Fig. 4.3: Example histogram for the radius r of reconstructed tracks.

implementation is able to perform the Online Event Selection on a NVIDIA Geforce GTX 1080Ti meeting the performance requirements set.

4.5.1 Test Setup

In previous work by D. vom Bruch [13] it was stated that 12 *NVIDIA Geforce GTX 1080Ti* have sufficient performance for the Online Event Selection with 50ns time-frames. Therefore, we are using the 1080Ti as base for our measurements in the first test machine. Its hardware specifications are:

- CPU: Intel(R) Core(TM) i7-5930K
- GPU: NVIDIA Geforce GTX 1080Ti
- RAM: 16GB

Since D. vom Bruch's [13] new GPU generations released, the newest being the RTX 3000-series. Due to the ongoing worldwide chip shortage we could not get a hold on one of those, thus the newest GPU we are using as comparison is a *NVIDIA Geforce RTX 2080Ti*, in our second test setup:

- CPU: AMD Ryzen Threadripper 2970WX
- GPU: NVIDIA Geforce RTX 2080Ti
- RAM: 128GB

The graphics card being the most relevant part for our testings, both setups are named after their respective Graphic cards in use.

Test data is obtained using the Mu3e simulation framework [6]. This framework is based on Geant4 [2]. While phase I of the Mu3e project has a planned muon rate of $1 \cdot 10^8 \mu$/s, the algorithms' performance for higher muon rates is studied as well. This data can then be used as base for planning phase II of the project, with higher muon rates. Thus, data is simulated for different muon rates between $1 \cdot 10^8$ and $1 \cdot 10^9 \mu$/s.

4.5.2 Accuracy

	approx. % kept
signal tracks	97.45
true tracks	94.56
signal frames	94.42

Tab. 4.1: Measured accuracy for different parts of the algorithm.

Before testing the computational performance of our algorithm we take a look at its accuracy. For these tests a simulation with one signal event per frame for a muon rate of $1 \cdot 10^8 \mu$/s is used, and the results are summarized in Table 4.1. First we measure the amount of tracks we are able to reconstruct. The track reconstruction achieves to reconstruct more than 97% of signal tracks and over 94% of true simulated particle tracks. But the total number of tracks reconstructed is actually about 4 times as high as the amount of true simulated tracks, so the amount of false tracks is quite high.

We are able to correctly identify more than 94% of all frames using the GPU implementation. Applying the same track and vertex reconstruction on hits produced by simulated hits, yields over 98% of correctly identified frames. Therefore, the difference can be linked to errors in track reconstruction, failing to reconstruct the correct signal tracks. We can reduce this difference partially by adjusting the cut values for the vertex reconstruction. But this would also lead to more frames kept wrongfully, decreasing the benefit gained by this algorithm, making this less feasible than trying to increase the reconstruction accuracy.

4.5.3 Parameter Tuning

It is important to find the right parameters to keep the accuracy of the algorithm high, while keeping the computational cost low. For example the parameters in Section 3.3 are chosen using simulation data for their high impact on the data cut away. In this subsection we analyze the performance and accuracy impact for different cut values for different steps. This analysis is performed on data for a

muon rate of $1 \cdot 10^8 \mu$ using the 1080Ti setup, but the performance and accuracy impact is device independent.

We start by testing the impact of maximum number of triplets and tracks allowed, results are shown in Fig. 4.4. Both results show the expected results, where higher caps result in less kept frames and higher computational costs. Changes in the cap for maximum tracks have higher performance impact than for selection cuts, showing the extensive computational cost for the vertex reconstruction. To reach our goal of reducing the datarate by a factor of > 100, values have to be chosen that stay below 1%, even when vertices are found and some fluctuation in the processed data is detected. Therefore, 512 maximum triplets and 20 maximum tracks are chosen, keeping the number of frames kept between them at 0.68% in our simulation.

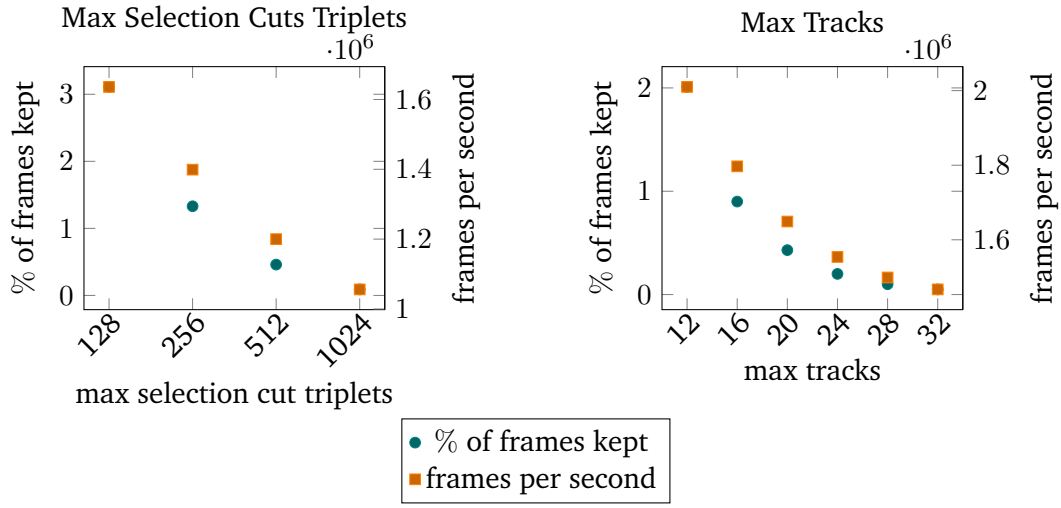


Fig. 4.4: Parameter measurements for the maximum cuts of Selection Cut Triplets (left) and number of tracks (right). Each plot shows the percentage of frames kept, and the Event Selections performance, with this cap in place.

4.5.4 Thread, Block and Stream Count

In this subsection tests are performed to determine the best number of threads per block, number of blocks and streams for a muon rate of $1 \cdot 10^8 \mu$/s. Goal is to find the best workload balance for the GPU. First we determine the number of threads per block, the test results are shown in Fig. 4.5 a). The amount of threads need to be a good trade-off for the varying amount of parallelism in each step. Selection cuts still have exponential many parallel tasks, while the track reconsruction starts with the reduced workload. The vertex reconstruction again has exponential branches, but only in the number of tracks selected. The results for the 1080Ti setup show a clear peak for 128 threads, which is chosen for this setup. In contrast, the 2080Ti does not show a peak, therefore the highest number of threads 256 is chosen.

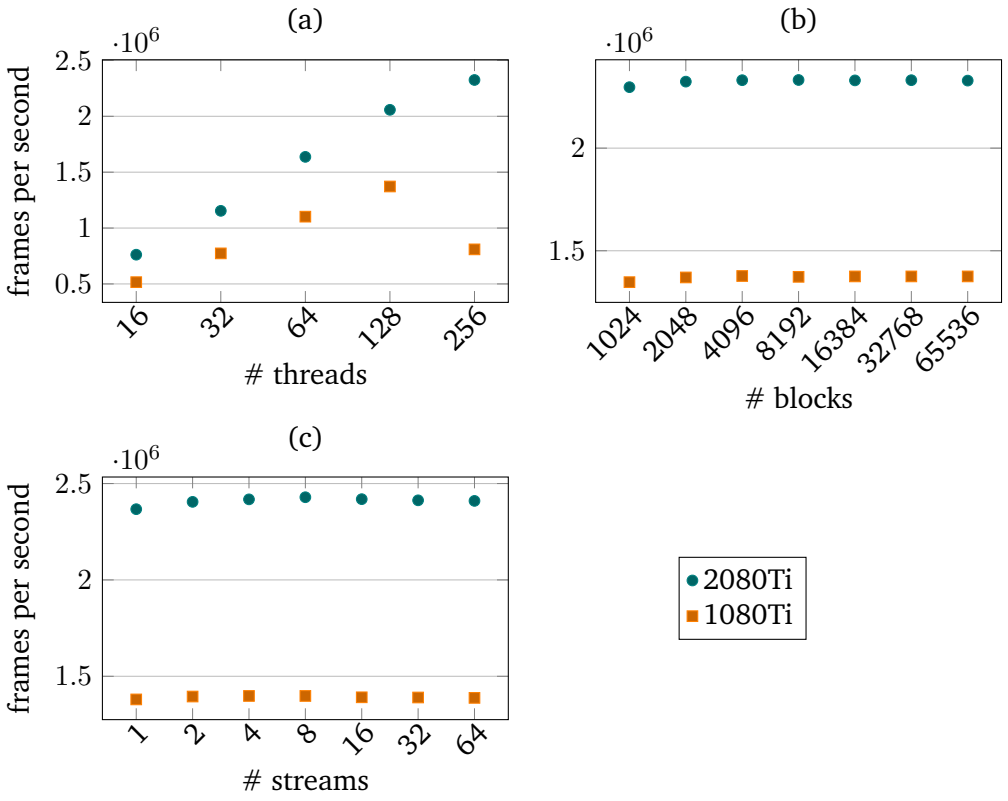


Fig. 4.5: Benchmarks for different number of threads (a), blocks (b) and streams(c). While different amount of threads have a high impact, the other have almost no impact.

The test results for number of blocks and number of streams in Fig. 4.5 b) and c) are not as clear. Both show no drastic increase in performance, for any of the tested values. The time the data needs to be transferred is rather small compared to the computation time, therefore the streams are not having a great impact. We are choosing 4 streams for the next benchmarks. For the final integration the minimum number of streams still has to be determined, since this depends on the final chunk size the FPGA transfers. As for block size, we are choosing 16384 blocks, resulting in 4096 blocks per stream.

4.5.5 Muon Rates

After optimizing the parameters we test the performance of our implementation on different muon rates. While the goal of this thesis is to write an implementation for phase I of the Mu3e experiment, with a muon rate of $1 \cdot 10^8 \mu$/s, studying the impact of higher muon rates is important as preparation for phase II. The results of these tests are shown in Fig. 4.6.

It is easily noticeable that higher muon rates, therefore higher number of hits and resulting combinatorics per frame, decrease performance drastically. A double muon

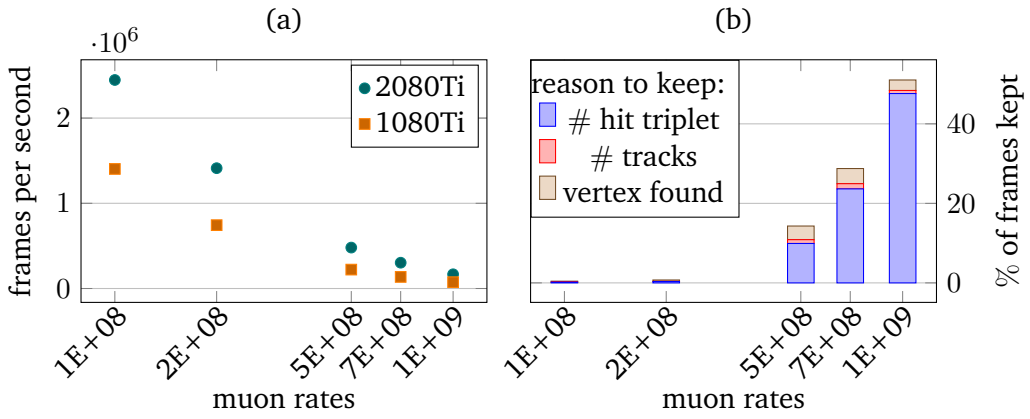


Fig. 4.6: Measurements for different muon rates. Left a) shows performance measurements for both systems on different muon rates, where the performance drops proportional to the muon rate. A different measurement is shown in b). Here the frames kept and the reasons why they are kept is shown. Higher muon rates introduce more frames kept, where the dominating reason is to many hit triplets produced during Selection Cuts.

rate roughly halves the frames per seconds evaluated. But it is also clearly visible that a 1080Ti performs fast enough to meet the requirements for phase I. It is able to evaluate $1.4 \cdot 10^6$ frames per second, while the 2080Ti trumps this performance with $2.45 \cdot 10^6$. The 2080Ti even has enough spare performance to allow for muon rates of up to $2 \cdot 10^8 \mu$/s to be executed on 12 computers.

Besides a higher requirement in computing power, the algorithm needs to be adapted and cuts chosen differently for higher muon rates. Increasing muon rates keep more and more frames. For our target muon rate $1 \cdot 10^8 \mu$/s only 0.25% of frames are kept in the simulation, reaching our goal of reducing the data rate by a factor of over 100.

One reason for more frames being kept are the higher amount of hits, resulting in to many triplets and tracks for each frame. As seen in Fig. 4.6 b) the main reasons frames are kept is to many hit triplets passing the Selection Cuts. The other reasons are negligible in comparison. The caps used have been optimized for a muon rate of $1 \cdot 10^8 \mu$/s, and therefore have to be adjusted for higher rates as well.

Increasing the caps alone is not enough for $1 \cdot 10^9 \mu$/s, since their maximum is defined by the amount of shared memory available. At this rate to many hits in one frame may exist, allowing for to many possible triplet combinations. Selection Cuts alone is not enough to filter the number of triplet combinations to a low enough level to fit into shared memory. Therefore, an adaption of the algorithm would have to be developed, where selection cuts and track reconstruction is bundled together and split into chunks. During computation of these chunks shared memory could be reused, keeping general memory usage low and allow for possibly higher amounts

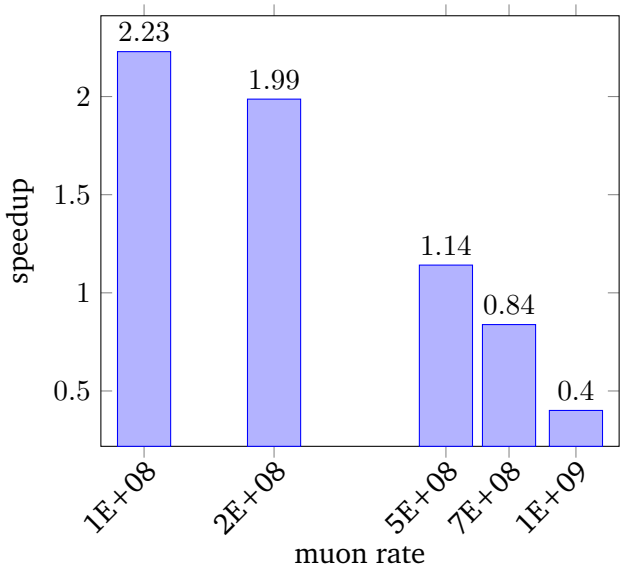


Fig. 4.7: Performance comparison of our implementation with D. vom Bruch's [13] version displayed as speedup. Our implementation has a speedup of over 2 for lower muon rates, but loses this advantage for higher rates.

of tracks to be saved. As we have seen in Section 4.5.3 the current caps allow for better performance, which is needed to meet the performance requirements set. The adaption described above has higher performance requirements, and does not fit into the requirements for phase I of the Mu3e experiment and is therefore not tested in this thesis.

Finally, we are using this simulation setup to compare the performance of our implementation with the implementation by D. vom Bruch [13]. It is important to note, that our implementation includes the whole Online Event Selection algorithm, while D. vom Bruch's work only includes the track reconstruction and vertex fit. This implementation works on previously generated triplet combinations, cut by the old Selection Cut algorithm. All frames are skipped automatically, where more than 1024 hit triplets are possible. These combinations are calculated using $n_0 \cdot n_1 \cdot n_2$, where n_i is number of hits in the i-th layer.

Benchmarks are created using the 1080Ti setup and are shown in Fig. 4.7. For the target muon rate of $1 \cdot 10^8 \mu/\text{s}$ we are achieving a speedup of 2.23 with our implementation. The speedup diminishes for increasing muon rates, due to frames being skipped for to many possible triplet combinations in D. vom Bruch's version. In contrast our implementation always performs the Selection Cuts and only stops with the frame if during that step to many triplet combinations are found. This extra computation slows the whole process down for higher rates, where the result is ultimately the same for a growing number of frames, as seen in Fig. 4.7 b).

5 Summary and Outlook

In search for physics beyond the Standard Model the Mu3e project searches for the heavily suppressed $\mu^+ \to e^+e^-e^+$. Detecting the decay would be a critical step for new physics. Not observing this decay during phase I of the experiment would lead to a new upper limit for the branching ratio to $2 \cdot 10^{-15}$. For achieving this goal a high muon rate and a detector with good momentum and vertex resolution is required, introducing high amounts of data to be collected, including a high amount of background. This thesis focused on filtering this data, reducing the actual data rate collected by a factor of over 100% in realtime.

We have introduced an algorithm consisting of three steps for the filter process, called *Online Event Selection*. This algorithm was previously introduced by D. vom Bruch [13] and improved in this work. The algorithm uses detected hits from the detectors four inner layers as input, bundled into timeframes. It reconstructs possible particle tracks from these hits using combinations of three hits in the first three layers. As first step these hit combinations are filtered using *Selection Cuts*, where unlikely combinations are filtered out using simple calculations. We have introduced new improvements in this step, by reducing the amount of steps needed and their computational complexity. These improvements lead to only 5% of hit combinations being kept for the track reconstruction.

After filtering the hit combinations they are used as base for track reconstruction. The track reconstruction used is based on the offline track reconstruction for the Mu3e detector using a novel triplet fit for reconstructing the tracks [10]. It is based on the assumption of Multiple Scattering being the only source of uncertainty, bending particle tracks when passing through detector material. These kinks introduce the need for reconstruction of two helices with same curvature and meeting in the central hit. Multiple Scattering and linearization are used to derive an analytical solution for these two helices.

With the reconstructed track as basis we are searching for possible spatial event vertices in this frame. If such an vertex is detected, the frame is kept, otherwise it is discarded. The vertex reconstruction uses triplets of two positron tracks and one electron track. These tracks are tested for signal compatibility, by calculating their combined energy. Passing this test the track triplets are projected into two

dimensions, finding their intersections. If intersections between all tracks are found, a vertex estimate is estimated as weighted mean of these intersections. Then the closest points on the tracks to this vertex are calculated, projected back into three dimensions and checked for proximity to the target. If this proximity is given for all three points on their tracks, the total momentum is calculated and tested for signal compatibility. Passing this test leads to the assumption, that a signal event could have happened in this frame, and the frame is kept for storage.

After introducing the algorithm we introduced our implementation on GPUs using CUDA, including a memory layout, online monitoring and the parallelisation schema. This implementation was then tested and shown to reach the goal of reducing the data rate by a factor of 100 for the targeted muon rate of $1 \cdot 10^8 \mu$/s. We achieve an accuracy of $> 97\%$ for signal track reconstruction and are able to keep more than 94% of all signal tracks. Finally, we compared our implementation with the previous work by D. vom Bruch [13], which only implemented the track and vertex reconstruction on GPUs. Still we achieved a speedup of more than 2 for our targeted muon rate. These results show that a NVIDIA Geforce GTX 1080Ti is a good lower bound for computational power needed for the Mu3e detector.

5.1 Outlook

The vertex reconstruction algorithm provides good results, but is bound by the track reconstruction efficiency. This can be improved by including more layers into the reconstruction algorithm, increasing computational requirements. But as the test results have shown newer graphics cards, like the NVIDIA Geforce RTX 2080 Ti have spare performance, which could be utilized for track reconstruction using more layers.

The algorithm introduced only uses fixed frames as basis for reconstruction, not respecting a particle passing between multiple layers over neighbouring frames. Therefore, the algorithm should be improved in the future by considering these exceptions as well. One way to achieve this is to always reconstruct two frames together, possibly reducing the general frame window below 64ns. This introduces a lot of extra computational complexity. New and faster GPU generations could introduce the performance needed for this step.

Another direction to increase performance could be evaluated, by using direct memory access from FPGAs to GPUs. This is not possible with the gaming graphic cards used, but the ongoing chip shortage shortens the gap in price between gaming and professional graphic cards, making this a feasible option to explore.

Bibliography

[1] G Aad, T Abajyan, B Abbott, et al. „Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC“. In: *Physics Letters B* 716 (1 2012), pp. 1–29 (cit. on p. 1).

[2] S. Agostinelli, J. Allison, K. Amako, et al. „Geant4a simulation toolkit“. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506 (3 July 2003) (cit. on pp. 16, 45).

[3] Xiaocong Ai, Corentin Allaire, Noemi Calace, et al. *A Common Tracking Software Project*. 2021 (cit. on p. 25).

[4] Xiaocong Ai, Georgiana Mania, Heather M. Gray, Michael Kuhn, and Nicholas Styles. „A GPU-based Kalman Filter for Track Fitting“. In: *Computing and Software for Big Science* (2021) (cit. on p. 25).

[5] Kazuyoshi Akiba, Marina Artuso, Ryan Badman, et al. „Charged particle tracking with the Timepix ASIC“. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 661.1 (Jan. 2012), pp. 31–49 (cit. on p. 11).

[6] K. Arndt, H. Augustin, P. Baesso, et al. „Technical design of the phase I Mu3e experiment“. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1014 (Oct. 2021), p. 165679 (cit. on pp. 2, 3, 5–9, 12–14, 16, 43, 45).

[7] Heiko Augustin, Niklaus Berger, Sebastian Dittmeier, et al. „The MuPix system-on-chip for the Mu3e experiment“. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 845 (Feb. 2017), pp. 194–198 (cit. on p. 11).

[8] J. Baudot, G. Bertolone, A. Brogna, et al. „First test results Of MIMOSA-26, a fast CMOS sensor with integrated zero suppression and digitized output“. In: IEEE, Oct. 2009, pp. 1169–1173 (cit. on p. 11).

[9] U. Bellgardt, G. Otter, R. Eichler, et al. „Search for the decay $\mu^+ \to e^+e^+e^-$“. In: *Nuclear Physics B* 299 (1 Mar. 1988), pp. 1–6 (cit. on pp. 2, 5).

[10] Niklaus Berger, Moritz Kiehn, Alexandr Kozlinskiy, and Andre Schöning. „A New Three-Dimensional Track Fit with Multiple Scattering“. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* (2016) (cit. on pp. 22, 24–27, 29, 51).

[11]W. Bertl, S. Egli, R. Eichler, et al. „Search for the decay $\mu^+ \to e^+e^+e^-$“. In: *Nuclear Physics B* 260 (1 Oct. 1985), pp. 1–31 (cit. on p. 7).

[12]V. Blobel. „A new fast track-fit algorithm based on broken lines“. In: *Statistical Problems in Particle Physics, Astrophysics and Cosmology - Proceedings of PHYSTAT 2005* (February 2006), pp. 68–71 (cit. on p. 25).

[13]D. vom Bruch. „Pixel Sensor Evaluation and Online Event Selection for the Mu3e Experiment“. PhD thesis. Heidelberg University, 2017 (cit. on pp. 3, 9, 12–16, 19–25, 29, 30, 32–35, 43, 44, 49, 51, 52).

[14]C. Cavicchioli, P.L. Chalmet, P. Giubilato, et al. „Design and characterization of novel monolithic pixel sensors for the ALICE ITS upgrade“. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 765 (Nov. 2014), pp. 177–182 (cit. on p. 11).

[15]Huangshan CHEN, Wei Shen, Konrad Briggl, et al. „Characterization Measurement Results of MuTRiG - A Silicon Photomultiplier Readout ASIC with High Timing Precision and High Event Rate Capability“. In: *Proceedings of Topical Workshop on Electronics for Particle Physics PoS* (Mar. 2018), p. 008 (cit. on p. 12).

[16]Mu3e Collaboration. *Mu3e internal Wiki* (cit. on pp. 5, 6, 10, 11, 15).

[17]R. R. Crittenden, W. D. Walker, and J. Ballam. „Radiative Decay Modes of the Muon“. In: *Physical Review* 121 (6 Mar. 1961), pp. 1823–1832 (cit. on p. 7).

[18]W. Demtröder. *Experimentalphysik 4 - Kern-, Teilchen- und Astrophysik* (cit. on pp. 1, 2).

[19]Wolfgang Demtröder. *Experimentalphysik 2: Elektrizität und Optik*. Springer, 2004 (cit. on pp. 10, 17).

[20]R. Frühwirth and R. K. Bock. *Data Analysis Techniques for High-Energy Physics*. Cambridge University Press (cit. on pp. 17, 18).

[21]Andreas Herten. „GPU-based Online Track Reconstruction for PANDA and Application to the Analysis of $D \to K\pi\pi$“. In: (2015), 255 S. (Cit. on pp. 3, 25).

[22]Virgil L. Highland. „Some practical remarks on multiple scattering“. In: *Nuclear Instruments and Methods* 129 (2 1975), pp. 497–499 (cit. on pp. 9, 25).

[23]M. Kiehn. „Pixel Sensor Evaluation and Track Fitting for the Mu3e Experiment“. PhD thesis. Heidelberg University, 2016 (cit. on pp. 17, 25–27).

[24]M. Köppel. „Data Flow in the Mu3e Filter Farm“. MA thesis. Mainz University, 2019 (cit. on p. 12).

[25]Alexandr Kozlinskiy. *Personal Notes*. 2022 (cit. on p. 28).

[26]Alexandr Kozlinskiy. „Track reconstruction for the Mu3e experiment based on a novel Multiple Scattering fit“. In: *EPJ Web of Conferences* 150 (2017), pp. 1–10 (cit. on p. 24).

[27]Gerald R. Lynch and Orin I. Dahl. „Approximations to multiple Coulomb scattering“. In: *Nuclear Inst. and Methods in Physics Research, B* 58 (1 1991), pp. 6–10 (cit. on p. 25).

[28]M. Müller. „A Control System for the Mu3e Data Acquisition“. MA thesis. Mainz University, 2019 (cit. on p. 13).

[29]NVIDIA. *CUDA Toolkit*. URL: https://developer.nvidia.com/cuda-toolkit (cit. on p. 35).

[30]NVidia and University of Illinois. *NVidia Teaching Kit (License: Creative Commons BY-NC-SA)* (cit. on pp. 35, 36).

[31]David Rohr. „Usage of GPUs in ALICE Online and Offline processing during LHC Run 3“. In: *EPJ Web of Conferences* 251 (2021), p. 04026 (cit. on p. 3).

[32]David Rohr, Sergey Gorbunov, and Volker Lindenstruth. „GPU-accelerated track reconstruction in the ALICE High Level Trigger“. In: *Journal of Physics: Conference Series* 898 (3 2017) (cit. on p. 3).

[33]David Rohr, Sergey Gorbunov, Schmidt Ole Marten, and Ruben Shahoyan. „GPU-Based Online Track Reconstruction for the ALICE TPC in Run 3 With Continuous Read-Out“. In: *EPJ Web of Conferences* 214 (2019), p. 01050 (cit. on p. 25).

[34]Bertil Schmidt, Jorge Gonzalez-Dominguez, Christian Hundt, and Moritz Schlarb. *Parallel Programming: Concepts and Practice*. 1st. Morgan Kaufmann Publishers Inc., 2017 (cit. on p. 35).

[35]A. Schöning. *A Three-Dimensional Helix Fit with Multiple Scattering using Hit Triplets*. June 2014. Internal Notes (cit. on p. 29).

[36]Mike Seidel et al. „Production of a 1.3 MW Proton Beam at PSI“. In: *Conf. Proc. C* 100523 (2010). Ed. by Akira Noda, Christine Petit-Jean-Genaz, Volker R W Schaa, Toshiyuki Shirai, and Akihiro Shirakawa, TUYRA03 (cit. on p. 6).

[37]G. Stanic. „A Camera Alignment System for the Mu3e Experiment“. MA thesis. Mainz University, 2021 (cit. on pp. 11, 43).

[38]M. Tanabashi, K. Hagiwara, K. Hikasa, et al. „Review of Particle Physics“. In: *Physical Review D* 98 (3 Aug. 2018), p. 030001 (cit. on p. 8).

[39]*The Nobel Prize in Physics 2015 - Press Release*. URL: https://www.nobelprize.org/prizes/physics/2015/press-release/ (cit. on p. 1).

[40]Eric W. Weisstein. *Circle-Circle Intersection*. URL: https://mathworld.wolfram.com/Circle-CircleIntersection.html (cit. on p. 31).

[41]A. X. Widmer and P. A. Franaszek. „A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code“. In: *IBM Journal of Research and Development* 27 (5 Sept. 1983), pp. 440–451 (cit. on p. 14).

[42]*Wikipedia, Standard-Model*. URL: https://en.wikipedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg (cit. on p. 1).

List of Figures

List of Tables

Colophon

This thesis was typeset with LaTeX 2$_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

I hereby declare that I have written the present thesis independently and without use of other than the indicated means. I also declare that to the best of my knowledge all passages taken from published and unpublished sources have been referenced. The paper has not been submitted for evaluation to any other examining authority nor has it been published in any form whatsoever.

Mainz, February 09, 2022

Fritz Valentin Henkys