Department of Physics and Astronomy

University of Heidelberg

Bachelor thesis

in Physics

submitted by

Erik Steinkamp

born in Lemgo

Friday 11th March, 2022

Hit Clustering in the *Mu3e* Tile Detector

This Bachelor thesis has been carried out by

Erik Steinkamp

at the Department of Physics and Astronomy

under the supervision of

Prof. Dr. Hans-Christian Schultz-Coulon

ABSTRACT

The *Mu3e* experiment searches for the lepton flavor violating decay $\mu^+ \rightarrow e^+e^+e^-$, whose discovery would be a strong indication for physics beyond the Standard Model. The detector aims at a sensitivity of 10^{-16}. This is achieved via high vertex, momentum and time resolution, using a combination of different detector technologies.
A high time resolution is achieved by the tile detector, which uses scintillating tiles as well as silicon photomultipliers for readout. To ensure that correct timing information is used, a clustering algorithm for tile hits has been developed and analyzed during this work. For the analysis of the algorithm, data from a previously developed simulation of the detector has been used. Limiting the size of tile hit clusters to a realistic value yields an overall efficiency of 86.7%.

KURZFASSUNG

Das *Mu3e* Experiment sucht nach dem leptonflavorverletzenden Zerfall $\mu^+ \rightarrow e^+e^+e^-$, dessen Entdeckung ein starker Indikator für Physik jenseits des Standardmodells wäre. Ziel des Experiments ist eine Sensitivität von 10^{-16}. Dies wird durch eine hohe Vertex-, Impuls- und Zeitauflösung erreicht, wobei eine Kombination verschiedener Detektionstechnologien zum Einsatz kommt.
Eine hohe Zeitauflösung wird durch den Kacheldetektor erreicht, der sowohl szintillierende Kacheln als auch Silizium Photomultiplier zum Auslesen verwendet. Um sicherzustellen, dass korrekte Zeitinformationen verwendet werden, wurde im Rahmen dieser Arbeit ein Clustering-Algorithmus für Kacheltreffer entwickelt und analysiert. Für die Analyse des Algorithmus wurden Daten aus einer zuvor entwickelten Simulation des Detektors verwendet. Die Begrenzung der Größe der Kacheltreffer-Cluster auf einen realistischen Wert führt zu einer Gesamteffizienz von 86,7%.

Contents

Acronyms

SM	Standard Model of particle physics
BSM	beyond the Standard Model
PSI	Paul Scherrer Institute
BR	branching ratio
CLFV	charged lepton flavour violating
SUSY	supersymmetric
MS	Multiple Scattering
HV-MAPS	High Voltage Active Monolithic Pixel Sensors
SciFi	Scintillating Fibre
SiPM	Silicon Photomultiplier
PCB	Printed Circuit Board
TMB	Tile Module Board
ESR	Enhanced Specular Reflector
TID	track ID

1 Introduction

1.1 Motivation

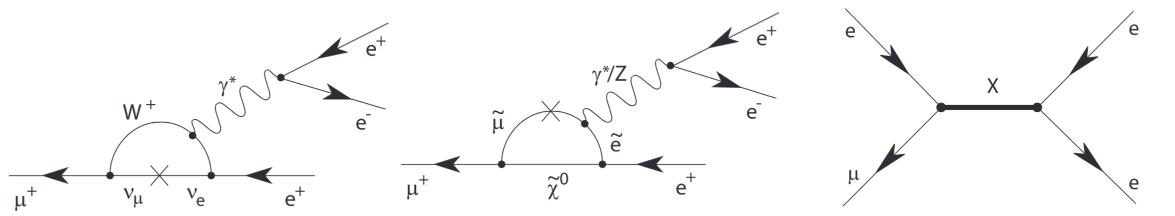
The decay of the muon is very interesting for searches for physics beyond the Standard Model (BSM). A recent experiment searching for physics BSM using the muon decay is the *g-2* experiment at Fermilab, which was able to measure a precise value for the muon anomalous magnetic moment, $a_\mu = (g-2)/2$. Combined with a previous measurement from the Brookhaven National Laboratory, the current discrepancy to the theoretical Standard Model of particle physics (SM) prediction is $4.2\,\sigma$ [6]. Another very interesting process to study with muons is the charged lepton flavour violating (CLFV) decay $\mu^+ \to e^+e^+e^-$. The current limit of 10^{-12} for this decay was measured by the SINDRUM-I experiment in 1987 [4]. The *Mu3e* experiment now aims to set a new limit for the CLFV decay $\mu^+ \to e^+e^+e^-$ [5]. It was proposed in 2012 and will be conducted at the Paul Scherrer Institute (PSI) in Villingen, Switzerland.

The *Mu3e* decay is especially interesting since the branching ratio (BR) of the decay in the SM is suppressed by 10^{-54} and therefore theoretically, there are no indistinguishable background processes. Background suppression is therefore only limited by the resolution of the detector. The *Mu3e* experiment will be operated in two phases, and in its second phase it aims to achieve a sensitivity of 10^{-16}, after the beamline at PSI as well as the detector have received an upgrade, which enables a muon rate of 2×10^9 muons per second. This in return means that the current limit will be improved by four orders of magnitude.

1.2 Physics beyond the Standard Model

The SM has been developed to describe how the basic building blocks of matter interact, governed by four fundamental forces. It has been well tested through countless experiments in the last decades and is currently the best description of the subatomic world. However, the SM still has some shortcomings, with the most obvious being the fact that the SM just includes the electromagnetic, the weak and the strong interaction while omitting gravity. Moreover, the SM cannot explain dark matter and dark energy. [2]

A search for a CLFV decay implies a search for physics BSM. The SM including neutrino oscillations (see Figure 1.1a) predicts a BR of 10^{-54} for the decay $\mu^+ \to e^+e^+e^-$ [4]. Theories that extend the SM by introducing various new heavy particles have been developed and are still lacking evidence, since no new particles or other deviations from the SM have been found. These heavy particles can mediate the CLFV either in virtual loops such as in supersymmetric (SUSY) models (Figure 1.1b), at tree level (Figure 1.1c), or in box diagrams [4].



(a) SM (b) BSM SUSY (c) BSM tree level

Figure 1.1: Feynman diagrams for CLFV in $\mu^+ \to e^+e^+e^-$. (a) in the SM by massive neutrino mixing; (b) by heavy mediating particles, such as in SUSY models; (c) tree level mediating particles. [4]

2 The *Mu3e* Experiment

The *Mu3e* experiment will be conducted in two phases. The first phase will be using the current muon beamline ($\pi 5e$) at PSI which provides a muon rate of 10^8 muons per second. This first phase aims at a sensitivity of 2×10^{-15}. The second phase will be conducted after the muon beamline at PSI has been upgraded to achieve a rate of 2×10^9 muons per second. For the second phase, the detector itself will also receive significant upgrades. The goal of the second phase is then to achieve the desired sensitivity of 10^{-16}. [1]

2.1 Signal and Background

The decay $\mu^+ \to e^+e^+e^-$ possesses some key characteristics that enables the differentiation between the signal decay and background events. Firstly, the decay is prompt, which means that the three electron[1] tracks coincide and emerge from the same vertex in time and space. Furthermore, the muons are stopped by a target and therefore decay at rest. Due to energy and momentum conservation, the energy sum of the electrons must equal the muon rest mass and the vectorial sum of the momenta has to vanish:

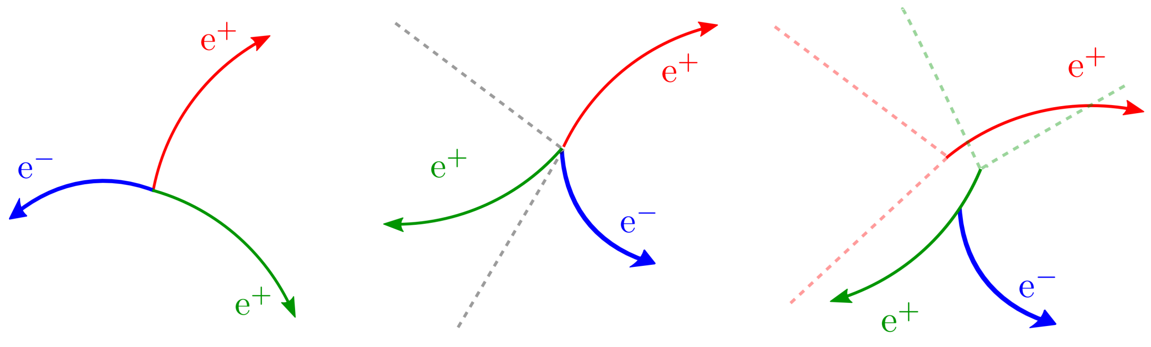
$$E_{tot} = \sum_{i=0}^{3} E_i = m_\mu \tag{2.1}$$

$$\vec{p}_{tot} = \sum_{i=0}^{3} \vec{p}_i = \vec{0} \tag{2.2}$$

[1]From here on, the term “electron” denotes generically both decay electrons and positrons.

Therefore, in order to suppress background processes, the *Mu3e* experiment aims at an excellent momentum, vertex and time resolution [3]. These requirements drive all aspects of the design of the *Mu3e* experiment.

The two types of background are shown in Figure 2.1. The irreducible background is caused by *internal conversion decays* ($\mu^+ \to e^+e^+e^-\nu_e\bar{\nu}_\mu$) which have a branching ratio (BR) of 3.4×10^{-5} [7] (see Figure 2.1b). Compared to the signal decay, two additional neutrinos are produced in the internal conversion decay. Taking advantage of this, the internal conversion background can be distinguished from the signal decays, provided that an excellent momentum resolution is given. The second type of background is the accidental background which can be caused by different processes. The dominating accidental background process is an overlay of a Michel decay ($\mu^+ \to e^+\nu_e\bar{\nu}_\mu$) and Bhabha scattering ($e^+e^- \to e^+e^-$) of a positron and an electron in the stopping target (see Figure 2.1c). This type of accidental background can therefore be suppressed if the two different processes can be separated in space and time. This in fact raises the requirement for an excellent time and vertex resolution of the detector [7].



(a) Signal (b) Internal Conversion (c) Accidental background

Figure 2.1: Comparison of the decay topologies of signal and background events. [7]

2.2 Design of the *Mu3e* Experiment

A schematic view of the *Mu3e* experiment in the configuration for phase 1 is shown in Figure 2.2. The muons will be stopped in the hollow double cone target, located in the center station of the experiment, where they decay into electrons. From the target, the electrons will pass the first two layers of the pixel detector. These inner pixel layers provide the vertex information of the decays. Since Multiple Scattering (MS) has a significant effect at the relative low momenta of the electrons, the experiment is constrained by a very low material budget. For this reason, the pixel layers have to be thinned down significantly, which is mainly achieved by using High Voltage Active Monolithic Pixel Sensors (HV-MAPS). The next part of the experiment, which is passed by the electrons, is the Scintillating Fibre (SciFi) detector, which provides the first timing information with a resolution of about 1 ns. The fibers have to be used due to the low material budget, compromising on light yield and therefore also on timing resolution. Another two layers of pixel detectors are placed outside the SciFi detector to provide further spatial information to track the electrons. To either side of the central station, a recurl station is placed to enhance the momentum resolution via two more pixel layers. The tile detector is also placed inside the two recurl stations. The material budget for the tile detector is not as stringent, as it is the last detector that will be hit. Therefore, its design is just constrained by the tight space in the recurl stations. This allows for a detector design, with a higher material budget, to be used in order to achieve a much higher timing resolution, compared to the SciFi detector, of better than 100 ps [1]. Since this thesis focuses on the tile detector, it will be discussed in detail in the next section.

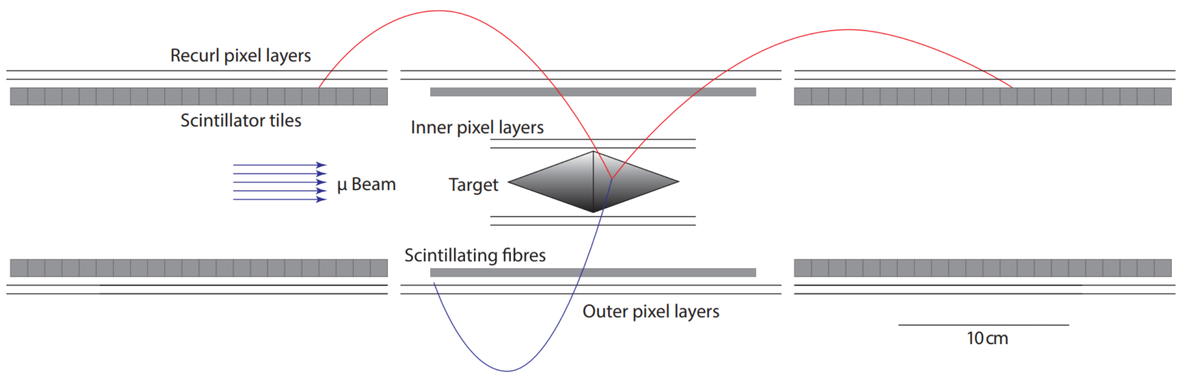
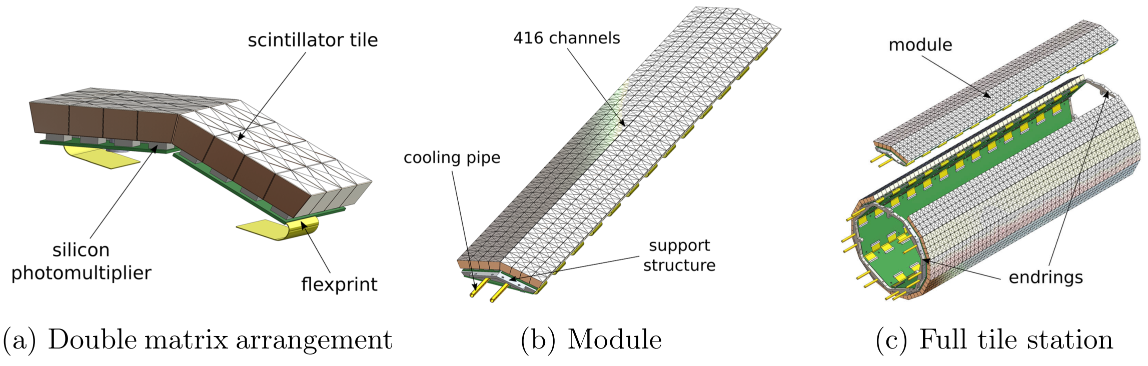


Figure 2.2: Schematic view of the experiment cut along the beam axis in the phase 1 configuration. [1]

2.3 The Tile Detector

In order to be able to correctly identify coincident signal decays and suppress accidental background, the goal of the tile detector is to provide the most precise timing information possible, with a timing resolution of better than 100 ps and a detection efficiency close to 100%.



(a) Double matrix arrangement (b) Module (c) Full tile station

Figure 2.3: CAD rendered views of the tile detector. [1]

The tile detector consists of two stations, placed in each recurl station below the outer pixel layers. The stations have a length of 32.4 cm along the beam direction (z-direction) and an outer radius of 6.4 cm. A CAD drawing of a full station is shown in Figure 2.3c. The full station is segmented into 52 tiles in z-direction and 56 tiles in ϕ-direction (along the azimuthal angle), adding up to 2912 tiles in total per station. The tiles have a size of $6.3 \times 6.2 \times 5.0\,\mathrm{mm}^3$ and are individually wrapped in Enhanced Specular Reflector (ESR) foil to isolate the tiles from one another and also increase light yield. Each tile is read out by a $3 \times 3\mathrm{mm}^2$ Silicon Photomultiplier (SiPM), which is soldered to the tile matrix Printed Circuit Board (PCB), assembling a 4×4 channel matrix. A whole *module* of the tile detector is composed of 26 matrices (see Figure 2.3b). The matrices are connected to the Tile Module Board (TMB) which hosts 13 readout ASICs (MuTRIG), one for every double matrix arrangement (see Figure 2.3a). A full station of the tile detector is then made up of 7 modules arranged in a circle (see Figure 2.3c).

2.4 Detector Simulation

All data that was used in the frame of this thesis has been produced by a Geant4 [1] simulation of the *Mu3e* detector.
The Geant4 simulation also uses ROOT [2] as a data framework. The main way ROOT saves data, which was also used for the *Mu3e* simulation software, is in so-called *trees*, *branches* and *leaves*. In the case of the *Mu3e* simulation, the different trees contain the simulation data, the corresponding Monte Carlo truth data and alignment information. Examples for branches would then be the tile IDs of the tiles that have been hit, the time when the tiles have been hit, and so forth. Possibly the most important concept to note in terms of the simulation, in order to understand how the developed clustering algorithms work, is the *frame*. A frame is a time period of 64 ns in which the simulation data is split up. Frames originate from the way the reconstruction operates on GPUs (Graphical Processing Unit). Since a large dataset can not be computed all at once while maintaining a reasonable hardware requirement, the simulation works by performing the necessary computations frame by frame. This way, the production of smaller datasets is even possible on regular consumer hardware. Figures 2.4 and 2.5 show a Visualization of events from one frame in a cross-sectional and a side view respectively. These figures were created using the *Mu3e* Display tool included in the simulation software package.

[1]"Geant4 (for GEometry ANd Tracking) is a platform for "the simulation of the passage of particles through matter" using Monte Carlo methods. It is the successor of the GEANT series of software toolkits developed by The Geant4 collaboration, and the first to use object oriented programming (in C++)."[8]

[2]"ROOT is an object-oriented program and library developed by CERN. It was originally designed for particle physics data analysis and contains several features specific to this field, but it is also used in other applications such as astronomy and data mining."[9]

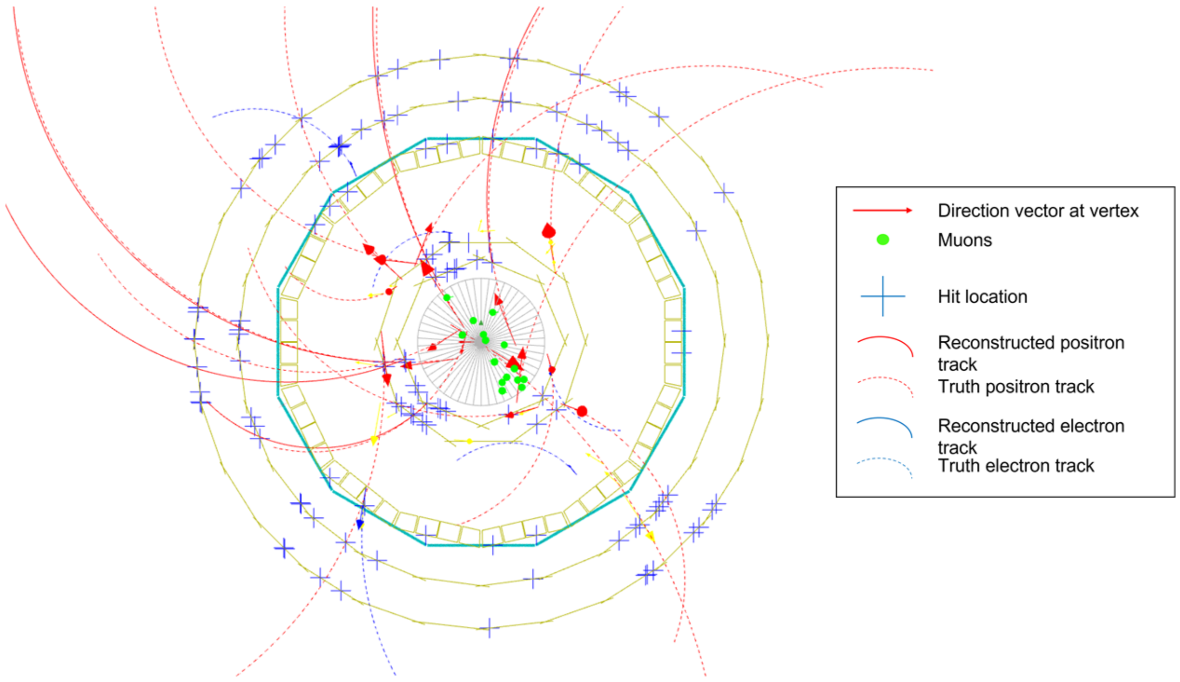

Direction vector at vertex
Muons
Hit location
Reconstructed positron track
Truth positron track
Reconstructed electron track
Truth electron track

Figure 2.4: Cross-sectional view of the detector showing the events from one frame.

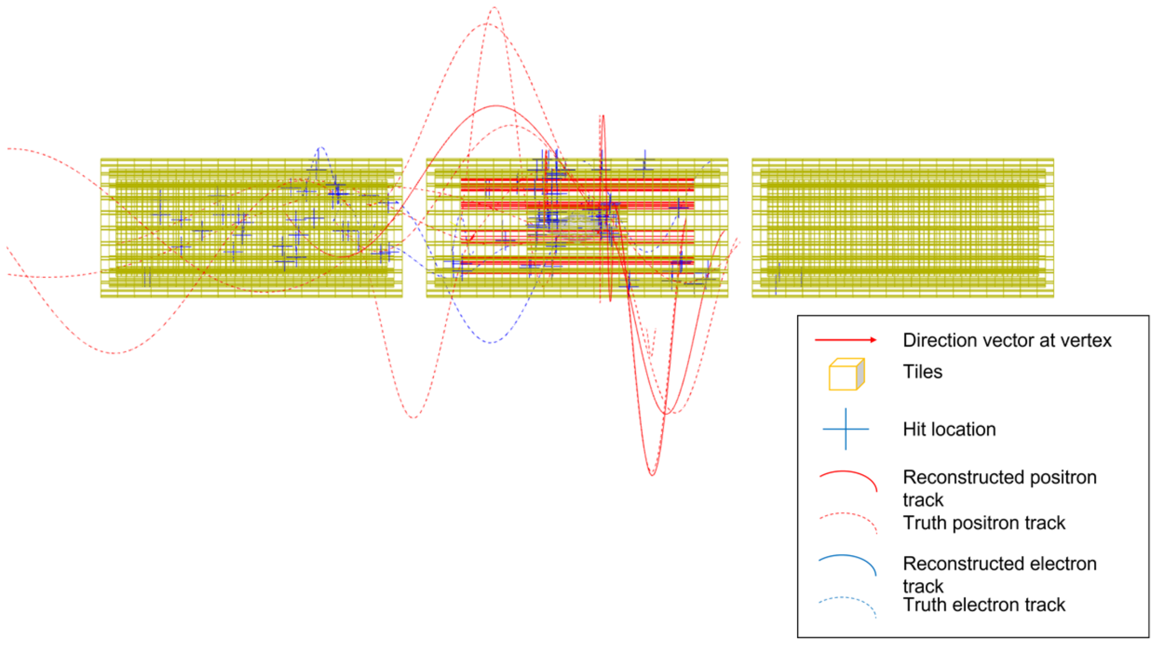

Direction vector at vertex
Tiles
Hit location
Reconstructed positron track
Truth positron track
Reconstructed electron track
Truth electron track

Figure 2.5: Side view of the detector showing the events from one frame.

3 Tile Hit Clustering

The development and evaluation of a clustering algorithm for the tile detector is the goal of this thesis. The final clustering algorithm, which will be discussed in close detail in the following chapters, uses a combination of time and spatial clustering. In this chapter, the concept of clusters, in the context of the experiment is explained. This is followed by a detailed explanation of the algorithms.

3.1 Origins of Clusters

A cluster in the context of the tile detector is defined as multiple tile hits caused by the same particle. As an illustration of these clusters, Figure 3.1 shows one station of the tile detector that has been rolled out into a two-dimensional plane.

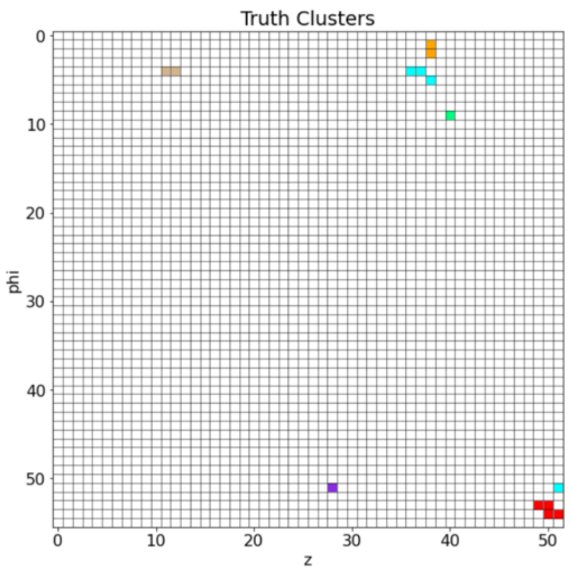


Figure 3.1: Map of hits on the tile detector.

The colored tiles are all tiles that have been hit in a particular frame, where the different colors indicate different clusters.
Clusters in general can have different origins or also originate from a combination of different processes. The most direct and obvious cause is when a particle hits multiple tiles in a row. For this to happen, the incident angle of the track on the tile surface has to be flat enough. A sketch illustrating this is shown in Figure 3.2.

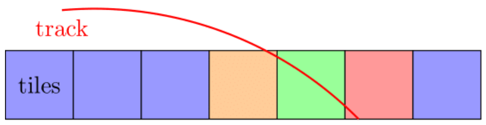
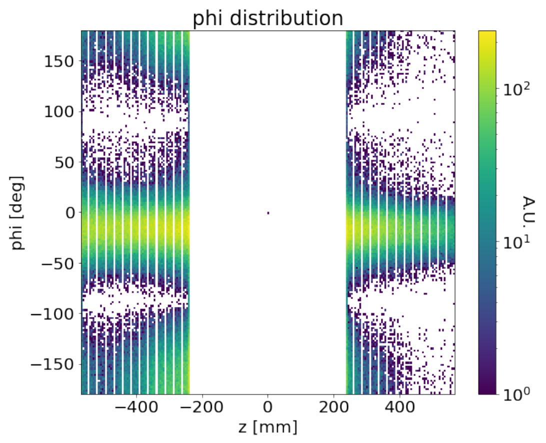
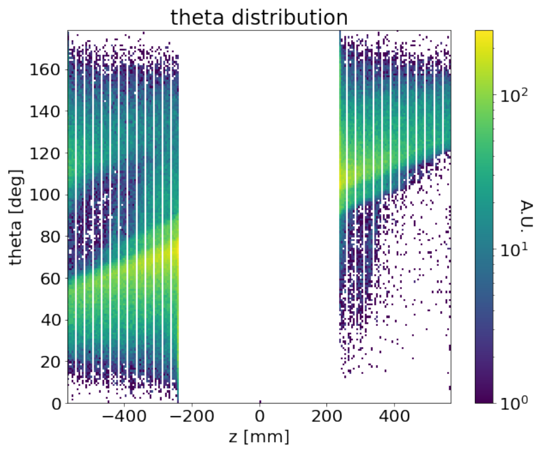


Figure 3.2: Sketch showing how one track hits multiple tiles, and therefore causes a cluster to form.

The ϕ-distribution as well as the θ-distribution have been studied, where θ is the angle in z-direction (beam direction), ranging from 0° to 180°, and ϕ is the azimuthal angle to the normal vector of the tile surface, ranging from $-180°$ to $+180°$. The corresponding distributions are shown in Figure 3.3.



(a) ϕ-distribution



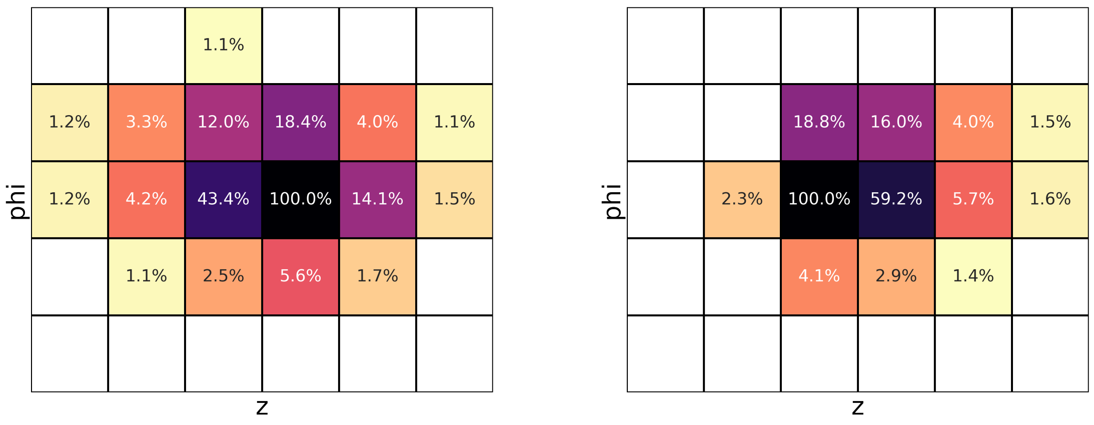
(b) θ-distribution

Figure 3.3: ϕ-distribution (a) and θ-distribution (b) of the incident angle of the tile hits. The angles are plotted against the z-direction and the density is shown using a logarithmic scale.

From these distributions, it can be concluded that the majority of tracks are vertical to the tile surface in ϕ-direction but rather flat in θ-direction, which results in many tracks hitting multiple tiles in a row. Optical crosstalk between two tiles is the more indirect cause of clusters. Here, light from one tile enters a second tile and is therefore detected as two hits. In order to minimize the optical crosstalk, all tiles are individually wrapped in foil. Apart from the optical crosstalk, it is also possible that an electron scatters in the first tile and triggers a hit in a second tile.

Another case of cluster formation can be caused by multiple tracks being very close to one another and therefore hit the same tile. If the tile is hit with enough time difference between two consecutive tracks, both hits can be detected. In this case, the algorithm ideally should be able to assign the timestamps in the correct order. However, if the hits are too close together in time so that the second hit occurs during the dead time, a pile-up effect occurs and the time information of all hits except the first one is lost, and therefore the clusters originating from these hits are invalid.

Figure 3.4 shows the probability to have a hit in the surrounding tiles for truth clusters with at least two hits. This is shown for the upstream and the downstream recurl stations. In both plots, a cluster hit in the center tile is denoted by a probability of 100%. For the surrounding tiles, the probability for that tile to be hit in the same cluster is given. Here, all probabilities under 1% have been left out due to their insignificance. This gives a visualization of the shape and size of typical clusters.



(a) Upstream (b) Downstream

Figure 3.4: Cluster maps showing the probabilities for hits in clusters with multiple hits for neighboring tiles around a center tile for the upstream (a) and downstream (b) stations.

3.2 Initial Studies

As a starting point to develop the clustering algorithm, different methods to build clusters from prior knowledge or other event information were studied. These methods are generally track-based.

3.2.1 Spatial Clustering

The general spatial clustering algorithm was developed on the basis of the current implementation in the reconstruction software (*Mu3e Trirec*), which is part of the simulation software package that has been developed for the *Mu3e* experiment. The implemented approach has also already been briefly studied by Christian Graf in his master's thesis [7].

The general idea behind the algorithm is to check every hit in a frame and see whether there are hits in the neighboring tiles. This is implemented by building masks around the tile that has been hit, and then all tiles included in the masks are checked whether they were hit as well. All tiles that have been hit within a mask are then added to the same cluster. In the thesis of Christian Graf, three different mask sizes were studied, as well as their efficiency and fake rate (see Figure 3.5) [7].

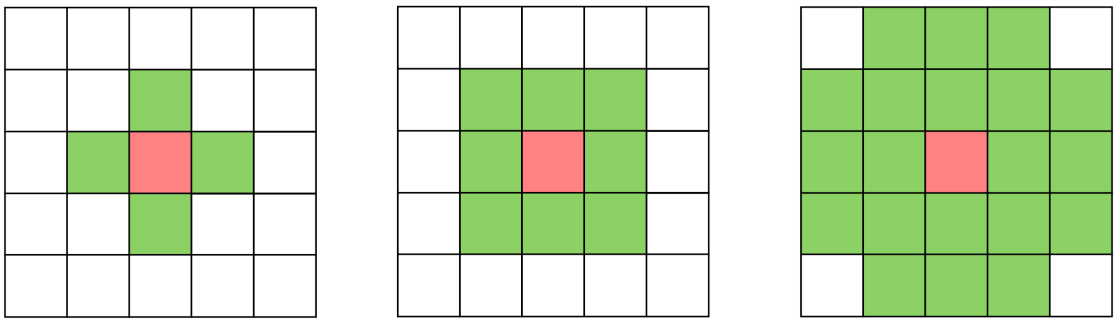


Figure 3.5: Definition of neighboring tiles for a small, medium and large mask in a flat detector projection. The red tile in the center marks the reference tile. For each mask, all green tiles are defined to be neighbors. [7]

In his studies, Christian Graf found that the large masks perform best for the experimental setup of phase 1. This is largely due to the fact that the muon rate is not as high as in phase 2 and the larger masks therefore do not have a large effect on the fake rate [7]. The large masks also make it possible to detect cluster hits next to a tile that was technically hit but where the hit was not recognized, for example due to dead time [7]. This also holds true for the algorithms developed for this thesis, therefore the efficiency analysis in the following chapter was completely carried out using the large masks.

3.2.2 Simple Masks

The first method that has been tried out using this type of spatial clustering was the simple mask method. It is very similar to the clustering algorithm that is currently implemented in *Mu3e* Trirec and to the method that has been studied by Christian Graf [7], which rely on the information of *seed tiles* (the tiles the mask is built around), typically from a pixel track or a hit-duplet from the two pixel detector layers surrounding the tile detector, which is projected onto the tile surface. Hits are searched in the surrounding of the seed tile by the use of masks, as shown in Figure 3.5. The same masks are used by the simple mask method. With this simple method, different seed tiles have been studied. The simplest approach of selecting the seed tiles is to use all hits and always build the masks around the next hit tile that has not been added to any cluster yet. Another method that has been investigated, as a reference case, is to use Monte-Carlo-truth data, to select tile hits that are tagged as primary, which are the first hit of a track in the tile detector. The concept of this hit-ID (HID) numbering is sketched in Figure 3.6. For the cluster seeds, all hits with HID=1 are used. A problem of this method is, that a tile would have an HID of one assigned to it even if the hit was not detected due to dead time or low energy deposition, since the HID is part of the truth information. Therefore, a lot of clusters can not be included as they do not have a hit with a HID of one. This results in over 20% of all hits not being associated to any cluster. For this reason the method has not been further investigated.

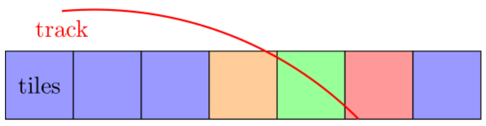


Figure 3.6: Explanation of the HID. The HIDs are assigned to every hit in a single track and are given by the number of tiles that have been hit by the track so far. Orange tile: HID=1. Green tile: HID=2. Red tile: HID=3.

The last method extending the clustering using seed tile hits uses a simple tracking algorithm, which uses hits in both pixel detector layers in the recurl stations to project a hit onto the tile surface. These hits are then selected as seed tiles. A sketch explaining this method is shown in Figure 3.7. Here, the red curve is the actual track and the black line through the two pixel hits is the linear projection. Unfortunately, this tracking method is not precise enough for clustering purposes, and therefore was not further investigated. For this reason, no special seed tiles were used in all upcoming analyses, since technically all tiles were used as seeds.

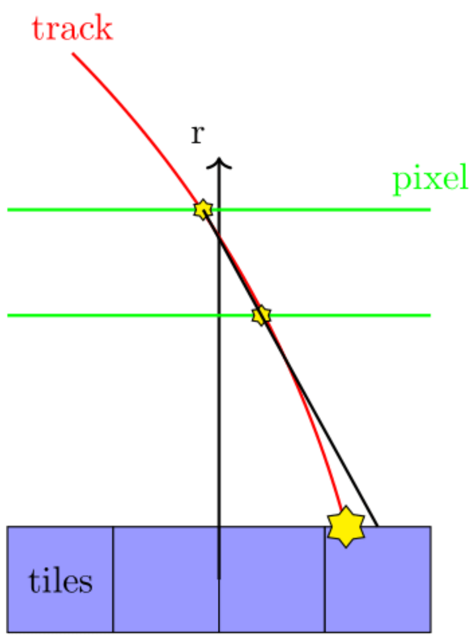


Figure 3.7: Tracking method using hits in both pixel layers of the recurl stations to project a track onto the tile surface.

3.2.3 Merging of Frames

When building events from fixed time intervals, there is always some inefficiency, related to clusters at the edge of the time windows. Therefore, after testing multiple methods which used one frame at a time, a frame merging algorithm has been developed to check whether there are significant differences in the overall efficiency of the clustering when considering multiple frames at the same time. This method now takes three frames into account with each iteration. While the masks are just built around the hits of the frame that is currently being clustered, the other cluster constituents are searched in all three frames simultaneously. This can be imagined as the three frames being stacked on top of one another before the masks are built. The result of this is an overlap of different frames that is sketched in Figure 3.8. The frame 0 is the one that is currently being clustered, and it also provides the seed tiles for the masks. The frames ± 1 are the neighboring frames that are checked simultaneously for cluster constituents.
It was found, that using this method around 6% of cluster hits come from neighboring frames. But without using time information a large error was introduced, since all three

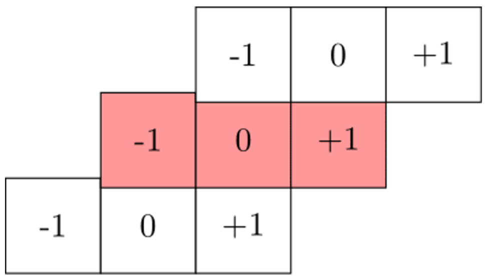


Figure 3.8: Sketch explaining the three-frame clustering.

frames were checked completely and not just the edges of the neighboring frames. This resulted in clusters from different frames being to close in space, and thus being combined by the algorithm. This was not further investigated, but the effect could be studied in the future with additional time information.

3.3 Time Segmentation

The time segmentation is the first step in the complete algorithm, which only uses information from the tile detector. The intention behind it is to provide a first rough clustering. This method can be applied since the rate in phase 1 of the experiment is not that high, and there is a high enough chance that two clusters are further apart in time than the hits in each individual cluster. If the rate was significantly higher, this method would fail since the clusters can not be separated on a time axis anymore. In order to show that the clusters are indeed relatively well distinguishable on a time axis, Figure 3.9 shows the time differences Δt between two hits in a single frame whose hits have been sorted in time (blue histogram). Without clusters, a completely random distribution within the interval of the framelength (64 ns) is to be expected. This would mean that an exponential distribution is to be expected, which would be linear on a logarithmic scale. For the most part this is also what can be seen in Figure 3.9, except for very small Δt. Here, a strong deviation from random behavior can be observed, which is caused by the clusters. To prove this, the part of the histogram, where Δt is the time difference between two hits from the same truth cluster is colored in orange in Figure 3.9. From this it is clear that the characteristic behavior is caused by the clusters.
In Figure 3.10 the first 5 ns of the same histogram as in Figure 3.9 are shown. Here, a fit

has been applied to the histogram in order to extract the characteristic time t_c of the peak caused by the clusters. The characteristic time t_c describes the time in which a cluster is usually formed, and therefore all hits of a given cluster should be generally found within this time. As a fit function, an exponential function with another exponential function as offset has been used. This now provides an estimate for the characteristic time t_c of 0.34 ns, which is also close to the time threshold of 0.4 ns, which was found to yield maximum efficiency for the pure time segmentation.

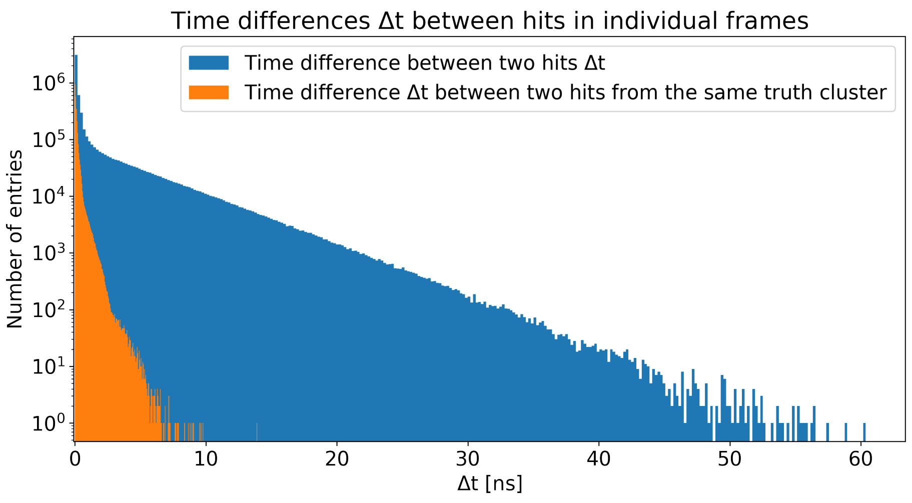


Figure 3.9: Time differences Δt between two hits.

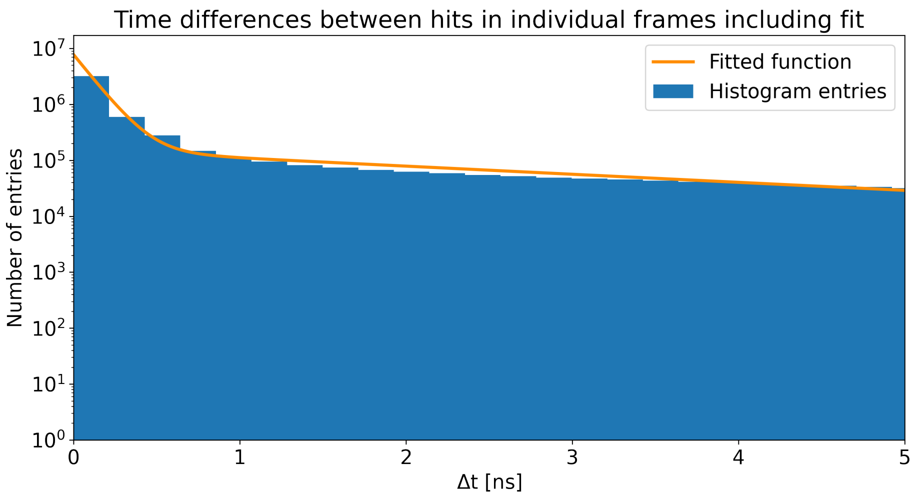


Figure 3.10: Fitted time differences Δt between two hits.

This threshold can now be used in the time segmentation, which works as follows. The first step of the segmentation is to sort all hits in time (see Figure 3.11). Then start with the first hit on the time axis (left side of Figure 3.11) and apply a time threshold to it and see if the next hit is included in this threshold. If the second hit is included, the same threshold is applied to the second hit to see whether the third hit is also included. This is repeated until the threshold does not include the next hit anymore. The end of the hit search for an individual cluster is indicated by the red arrows in Figure 3.11. All hits in this current iteration are then added to the same cluster and the next cluster starts with the first hit that has not yet been added to any cluster. The beginning and the end of each cluster is represented by green and red vertical lines respectively.

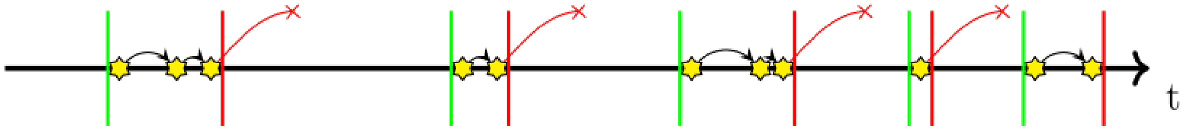


Figure 3.11: Sketch of the time clustering method, where the hits in a single frame are represented by stars on the time axis.

If the time segmentation is used in combination with the spatial clustering, the time threshold is chosen larger than what would be ideal if just the time segmentation was used. The intention behind this is to make sure the time clusters are rather too big and almost never too small. Therefore, a time cluster is likely to contain multiple truth clusters, but almost never split up truth clusters. The spatial clustering then provides additional information to separate time clusters, if they contain multiple truth clusters.

3.4 Full clustering algorithm

In the software developed for this thesis, the spatial clustering is also implemented using the masks which include the neighboring tiles, but here an iterative approach has proven to be the most efficient. As a first step, the time segmentation is applied to split hits in a frame into multiple hit collections, forming only clusters in the same time frame. This results in clusters with an average size of three hits and 10 cluster candidates per frame on average, compared to 16 true tracks hitting the tile detector on average per frame.

After all hits in a frame have been split up into hit collections during the time segmentation step, they are iteratively split further into clusters. This process is sketched in Figure 3.12. First, masks are iteratively applied to the hits in the individual hit collections, starting with the earliest one. All hits included in these masks are added to the same cluster. The next step is to apply another iteration of masks to the hits that have been found in the first iteration, and check whether there are further hits of the same hit collection. If further hits are found, they are added to the same cluster. This will be repeated until every hit in a given hit collection has been assigned to a cluster.

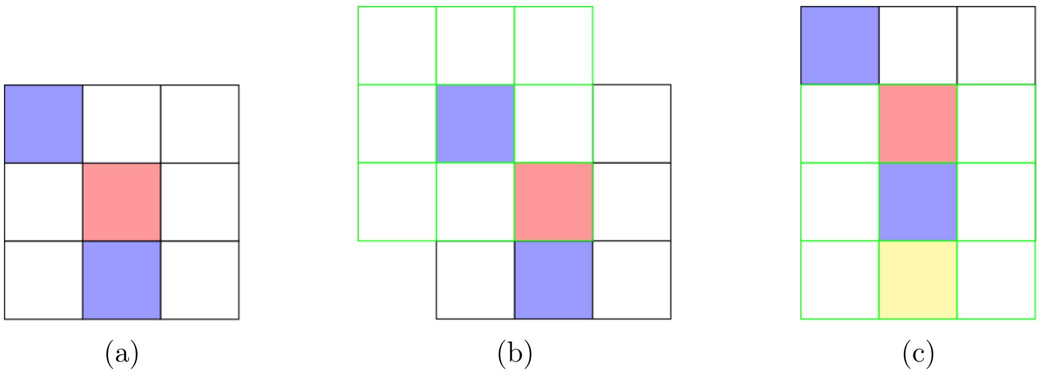

(a) (b) (c)

Figure 3.12: Sketch of the iterative spatial clustering process. a) Mask built around time cluster hit (red). b) Mask (green) built around first hit (blue) in mask around time cluster hit (red). c) Mask (green) built around second hit (blue) in mask around time cluster hit (red). The yellow tile is a time cluster hit found in the second iteration of masks (around blue hits).

4 Efficiency Analysis

The definition of a quality measure for the effectiveness of the different algorithms is critical to compare the methods, yet difficult to implement in practice. This is due to the fact that the Monte Carlo truth data provided by the simulation does not include the truth clusters directly. The closest information provided is the track ID (TID) which is assigned at the vertex to every track. Problematic is that if the hits with the same TID are taken as reference clusters, an intrinsic error is introduced, since some particles undergo elastic scattering processes which leads to multiple hits with the same TID, that are far apart in space and time. These would be examples of clusters that can not be correctly identified without Monte Carlo truth data but are also irrelevant for the purpose of the tile detector, since any viable timing information is lost. It would be ideal to discard all these clusters since the timing information after scattering processes is invalid in terms of track reconstruction. However, this is made nearly impossible by the fact that there is no data saved by the simulation that would make it easy to directly filter this. The only method left to solve this problem to some extent is to limit the size a cluster may have, in order to be considered for the efficiency.

In general the efficiency is defined as the average fraction of correctly assigned hits, when comparing to truth clusters using the TID.

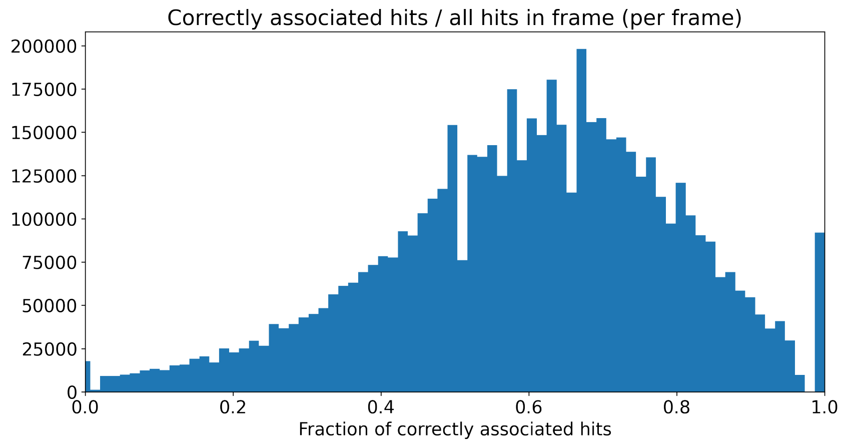
It is important to note here that, if no special seed tiles are used and a time segmentation is involved, a simple comparison of the TID fails as a method to determine the efficiency, if the time thresholds are too small. This occurs because the TID of hits in a cluster are then compared to the TID of the first hit in a cluster, as a replacement for the seed tile. In this case the time segmentation yields clusters with an average number of hits close to one. Therefore, the simple comparison would mostly compare the one hit to itself.

This would give an efficiency close to 100%, which obviously would be a false result. In order to avoid this and make the algorithm stable for all time thresholds, it also has to be checked whether there are multiple clusters with the same TID, since this would mean that all but one of the clusters with the same TID have been assigned incorrectly, which needs to be accounted for when calculating the efficiency. The cluster with the earliest hit is selected, as the only correct cluster. This is done automatically for all clustering algorithms, since it also has a small effect on the spatial clustering.

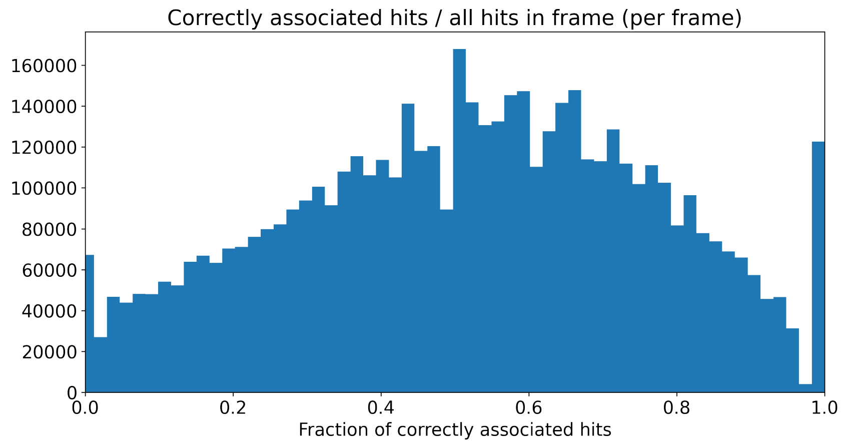
4.1 Comparison to Truth Information

As mentioned before, the direct comparison of the clusters to the TID does not represent a realistic efficiency analysis, because the Monte Carlo truth data includes clusters with hits that are far apart in space and time. Nonetheless, it still gives an idea of how close to the truth clusters the algorithms can get, even if the truth clusters include clusters from which the timing information generally is not wanted. For all upcoming analyses in this section, up to six million frames, which are equal to about 380 ms, were analyzed to provide enough statistics.

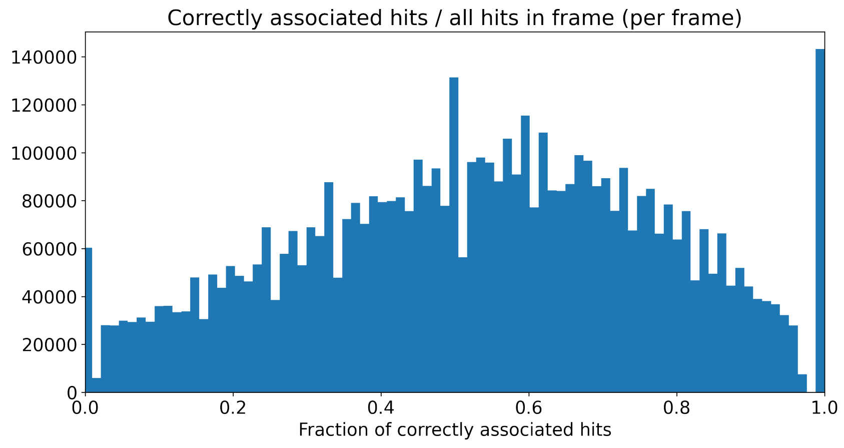
Pure time segmentation with a time threshold of 0.4 ns results in an efficiency of 56.1% (see Figure 4.1a). A time threshold of 0.4 ns was found to deliver the maximum efficiency for pure time segmentation.
The simple spatial clustering with straightforward TID comparison yields a lower efficiency of just 45.4% (see Figure 4.1b). An inferior efficiency for this method is not surprising, since the masks are often big enough to include hits from other clusters, which cannot be detected without timing information.
Combining the time segmentation and the mask based clustering also yields a low efficiency of 43.1% (see Figure 4.1c). Here it was found that the best result is achieved with a time threshold of 1.2 ns.



(a) Time segmentation



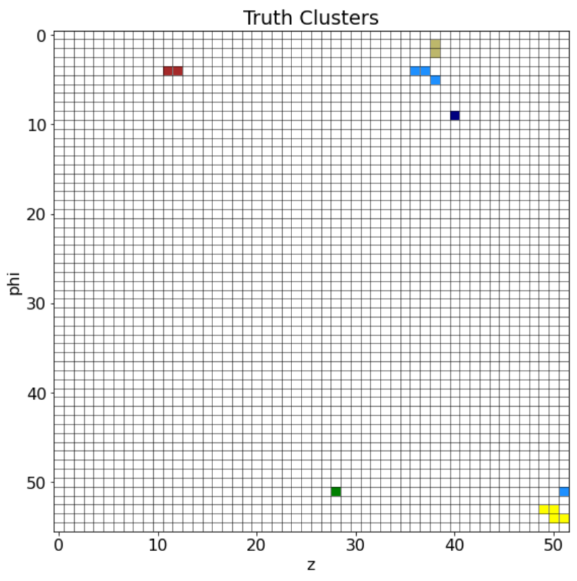
(b) Spatial clustering



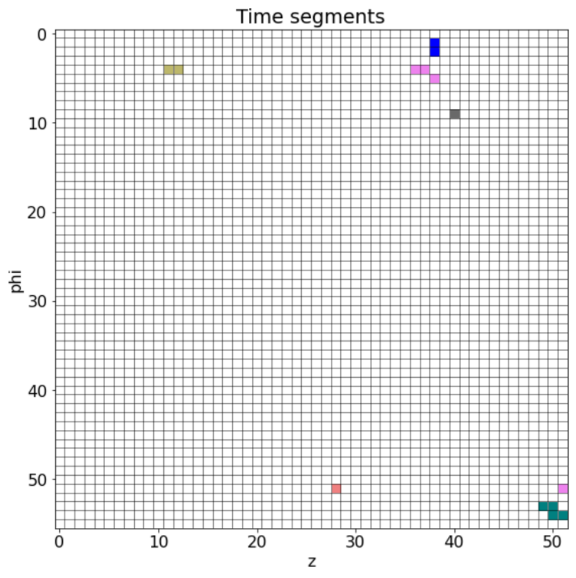
(c) Time & spatial clustering

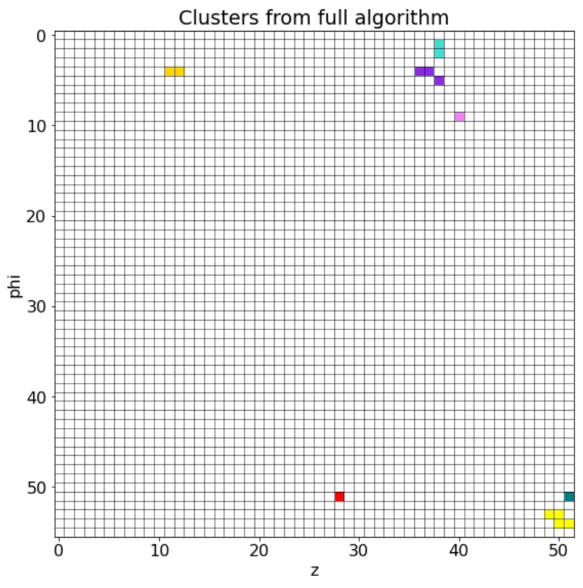
Figure 4.1: Efficiency histograms for (a) pure time segmentation, (b) pure spatial clustering, (c) combination of time and spatial clustering.

This result is quite surprising, since the algorithm combining time and spatial clustering is expected to yield much better results, and when single frames are visualized, it also seems like it is working as intended and performing well. Figure 4.2 shows a typical case where the time segmentation, as well as the full clustering, seem to work very well, since both algorithms associated all hits to the correct cluster. The horizontal axis shows the z-direction (direction of muon beam) and the vertical axis the ϕ-direction (azimuthal angle), where the unit is the number of tiles in either direction. The colored tiles are all hits in the frame and the colors indicate the individual clusters. To better understand why the algorithm does not perform better at first sight, we need to investigate in which cases it fails and whether this affects the experiment.



(a) Truth clusters





(b) Time segments (c) Clusters from full algorithm

Figure 4.2: Visualization of the clustering results for (b) time segmentation and (c) the full algorithm, compared to the truth clusters (a).

4.2 Causes for Inefficiency

There are two main cases where the algorithm fails to detect the truth clusters, as they are defined in the Monte Carlo truth data. In the first case, the algorithm merges multiple clusters into one, because they are close in time and space. An example for this case is outlined in red in Figure 4.3, where the hits are direct neighbors and only 73 ps apart in time. The horizontal axis again shows the z-direction (direction of muon beam) and the vertical axis the ϕ-direction (azimuthal angle), where the unit is the number of tiles in either direction. The colored tiles are all hits in the frame and the colors indicate the individual clusters. This case is intrinsically indistinguishable due to the finite spacial granularity and timing resolution of the tile detector, thus it can only be improved by using other information from the tracking system, as discussed in section 3.2. Fortunately, it is much more common, that the algorithm splits up truth clusters, since it has been observed that the number of clusters that are found is 9.7% higher than the number of truth clusters. One of those cases is outlined in blue in Figure 4.3. Here, the hits are far apart in space and also have large time differences between 0.5 ns and 1.7 ns.

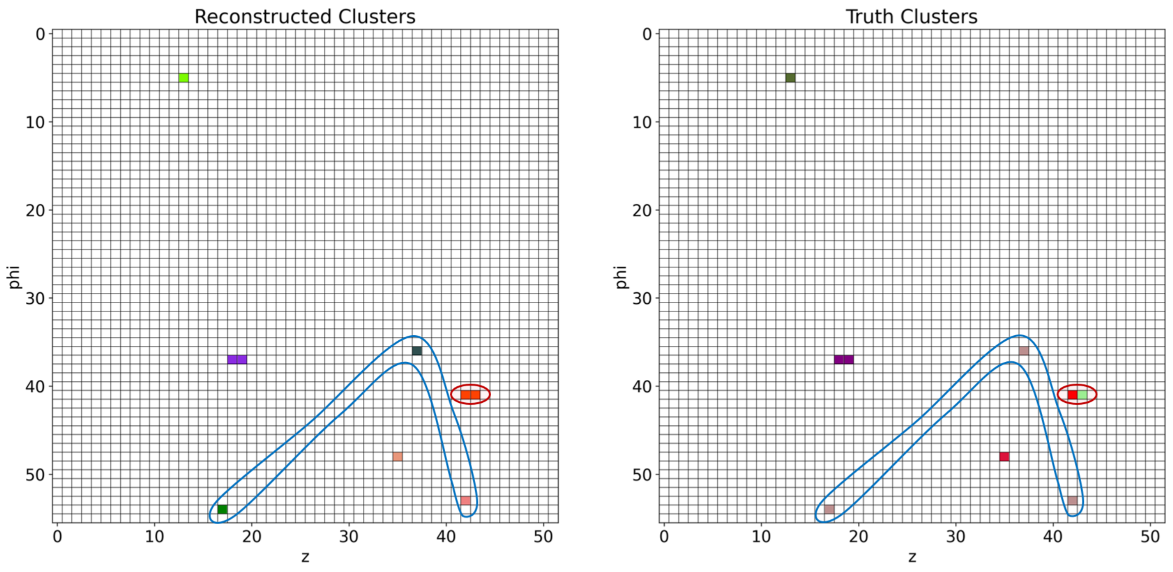


Figure 4.3: Visualization of the clustering results (left) compared to the truth clusters (right) for the cases where the algorithm fails.

This much more common case has already been briefly discussed at the beginning of this chapter. From the available information, it appears that this is most likely caused by elastic scattering processes that can lead to these big separations. Assuming there is no bug in the simulation software, the cluster shown in Figure 4.3 should be caused by the same particle. Unfortunately, this can not be proven with the simulated data currently available, but it could be verified that the three hits are caused by a positron with similarly high energy deposition in the tiles. This is exactly what is to be expected from elastic scattering processes.

Since the goal of the tile detector is to provide very precise timestamps, this type of cluster should be excluded, because there is no way to assign a proper timestamp to the track. Since there is no easy way of excluding these clusters from the efficiency calculation without a separate tracking algorithm, the effect these clusters have on the efficiency of the clustering algorithm can be approximated by limiting the size of the clusters, which are included in the calculation of the efficiency.

4.3 Analysis with Realistic Cluster Size

4.3.1 Truth Cluster Size

To properly select a realistic maximum cluster size for the calculation of the efficiency, we first take a look at the sizes of the truth clusters. To do so, Figure 4.4 shows the distribution of the maximum distance between two hits in a truth cluster. All clusters with just one hit therefore get a maximum distance of zero. The distance between two cluster hits is calculated in a three-dimensional space and is defined as the distance between the centers of the tiles. It can be seen, that the vast majority of clusters have a size of well under 50 mm. From this we can already conclude that a threshold of around 30 mm will include almost all clusters with relevant time information for the experiment. The small peak at around 120 mm is caused by clusters with hits that have very similar z-coordinates and ϕ-coordinates, which differ by around 180°, since it is the diameter of the tile detector.

This maximum distance was also plotted against the number of hits in truth clusters in Figure 4.5. The vast majority of clusters has a realistic extent and number of hits. In fact, even though the maximum distance is 1140 mm, which is the length of the diagonal of the whole detector, the mean distance is still 21.7 mm. Also, the mean number of hits per truth cluster is two, while the maximum is 32 hits. The gap in the middle is caused by the fact, that the center station is longer than the recurl stations. Therefore,

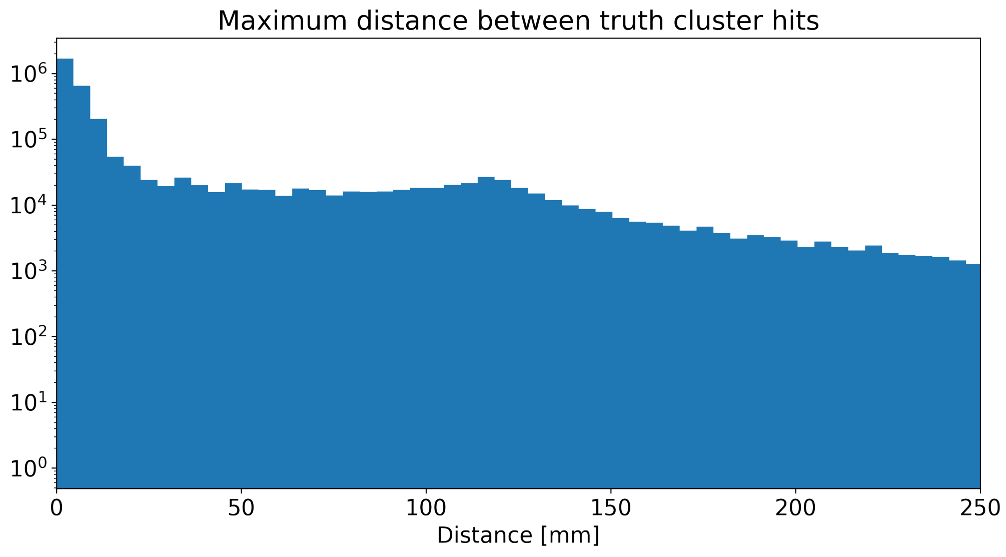


Figure 4.4: Histogram showing the maximum distance between two cluster hits in the truth clusters. Note that the horizontal axis is limited to 250 mm.

the part of the plot for shorter distances represents the clusters in a single station. This is followed by the gap, which is as large as the difference in length of the center and the recurl station. After the gap, the clusters with hits in both stations can be seen.

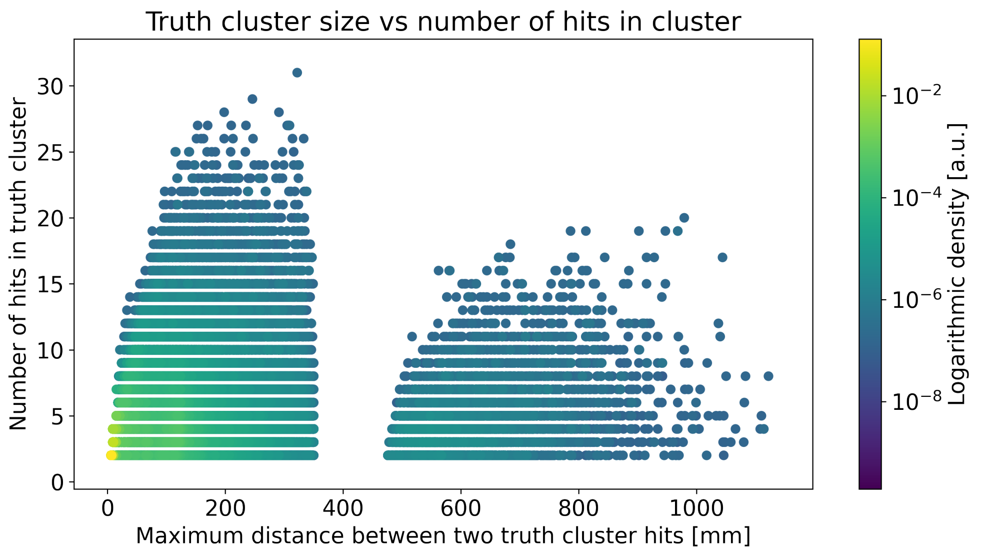


Figure 4.5: Maximum distance between two cluster hits in the truth clusters plotted against the number of hits in the clusters with logarithmic density.

With this information, we can now look at the amount of clusters that would have been left out by the different possible thresholds to check whether thresholds around 30 mm are realistic. Figure 4.6 shows the percentages of clusters that would be excluded for different thresholds.

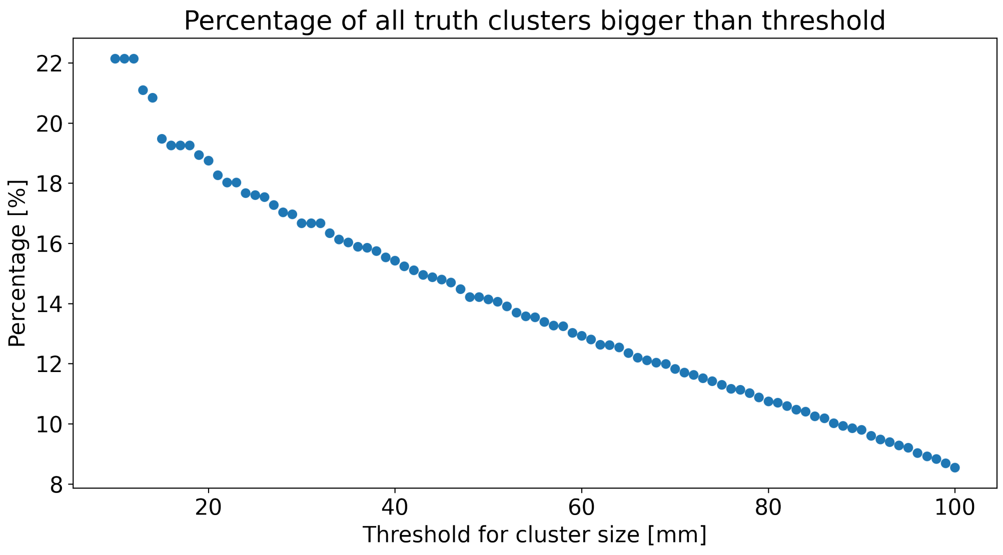


Figure 4.6: Percentage of truth clusters that have a maximum distance between hits bigger than the threshold on the horizontal axis.

To take this argument further, we can also take a look at how many hits a cluster usually has, which is also a limiting factor for the size of the clusters. The histogram in Figure 4.7 shows the amount of clusters with a specific number of hits. The mean number of hits per truth cluster is 2, and if clusters containing a single hit are not considered it is 3. From this, a threshold for the cluster size of 30 mm can finally be chosen, since it excludes the clusters that give unusable timestamps, while not excluding a significant amount of usable clusters.

A more precise efficiency calculation would just be possible if the clustering was done after a tracking matched a seed tile to every track, thus detecting most of the unusable clusters, which then could be filtered beforehand.

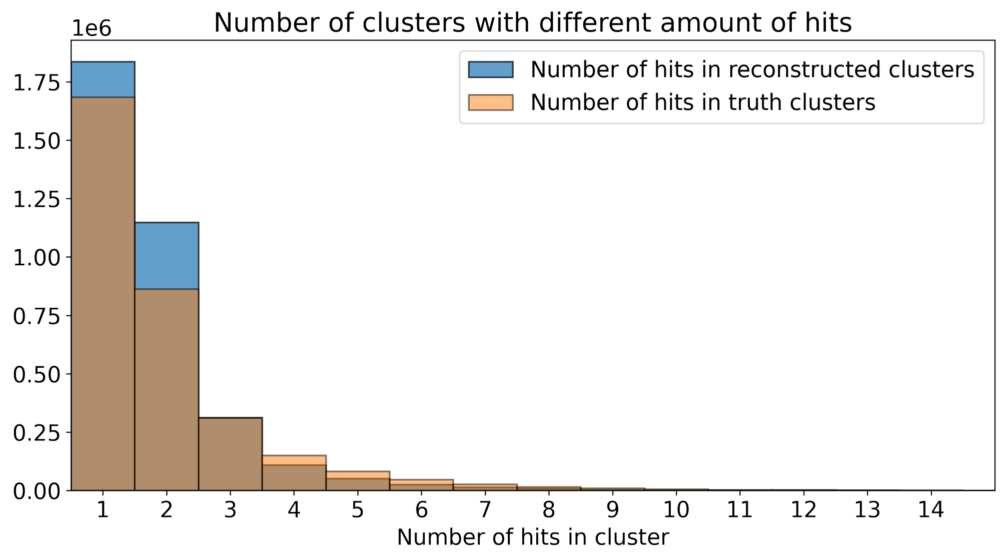


Figure 4.7: Histogram showing the number of hits in the clusters.

4.3.2 Efficiency for Limited Cluster Size

This section will discuss the efficiencies resulting from limited truth cluster sizes and compare them to the results for an unlimited truth cluster size.

For the calculation of the efficiency, all cases where a truth cluster was split up into multiple individual clusters have been checked for their maximum distance between two of the clusters. If the maximum distance exceeded a given Monte-Carlo-threshold, the clusters were excluded from the efficiency calculation. Figure 4.8 shows the efficiency plotted against the maximum cluster size. The efficiency is almost constant up to a Monte-Carlo-threshold of around 20 mm and then decreases linearly with increasing threshold.

Table 4.1 shows the efficiencies for different Monte-Carlo-thresholds of the cluster size for the algorithm combining time and spatial clustering, as well as for the two individual algorithms. All values have been calculated from around six million frames in order to provide enough statistics.

When just considering the efficiencies for an unlimited cluster size, the pure time segmentation performs far better in comparison to the other two methods. From this we can conclude that the case where the hits in a truth cluster are far apart in time and also

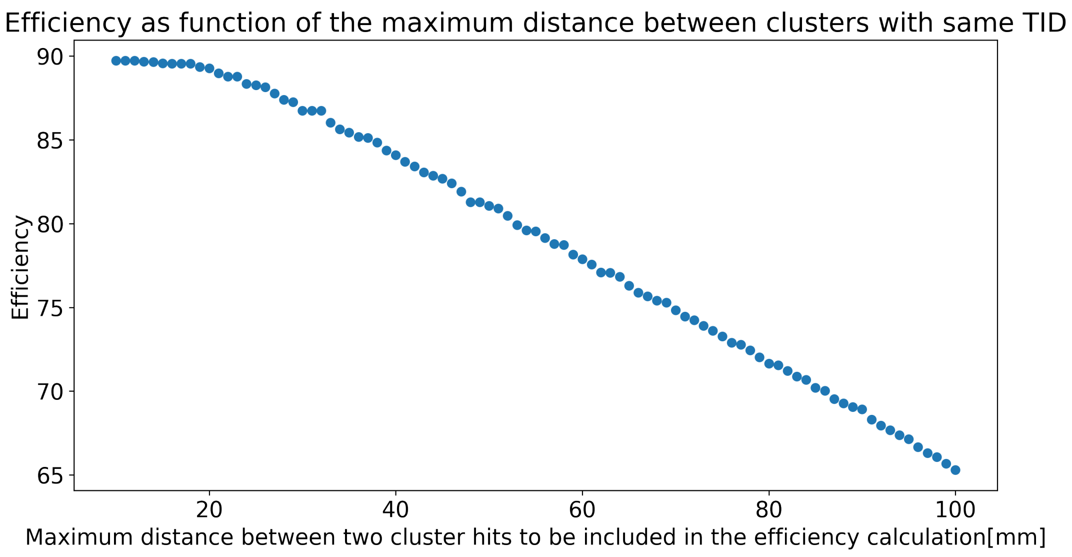


Figure 4.8: Efficiency of the clustering algorithm for different truth cluster size thresholds.

Threshold [mm]	Efficiency Time [%]	Efficiency Spatial [%]	Efficiency Combined [%]
None	56.08	45.44	43.13
20	79.33	85.53	89.24
25	79.14	84.67	88.23
30	78.90	83.38	86.74
35	78.68	82.28	85.45
40	78.43	81.11	84.10

Table 4.1: Efficiencies for different cluster size threshold for the three clustering algorithms.

far apart in space is more likely to affect the spatial clustering method. Therefore, it is very probable that a large spatial separation is more likely than a large temporal separation in these cases. By implication, the spatial clustering should exceed the performance of the time segmentation for limited cluster sizes. Exactly this can also be observed in the results shown in Table 4.1. Here it can be seen, that all clustering methods have increasing efficiencies for decreasing size thresholds, but the time segmentation never reaches 80% efficiency, while the pure spatial clustering reaches 85.5% and the combined clustering method reaches 89.2% efficiency. Also, for the previously selected threshold of

30 mm, the time segmentation performs worst with 78.9%, while the spatial clustering reaches 83.4% and the combined clustering again has the best efficiency with 86.7%. This also clearly shows the benefits of combining the time and the spatial clustering.

5 Summary & Outlook

The goal of the tile detector in the *Mu3e* experiment is to provide the most precise timing information possible. This precise timing information is needed to properly distinguish coincident signal decays from combinatorial background processes. Clusters occur naturally for shallow incident angles in the tile detector. Here one particle triggers multiple hits, which are usually in close proximity to one another. This means that multiple timestamps belong to a single particle. The goal of a clustering algorithm is therefore to associate all hits to the particle that caused them. Developing such a clustering algorithm was the goal of the work presented in this thesis. In the course of this work, multiple algorithms have been implemented to work with the data provided by the *Mu3e* simulation software. The final algorithm uses both temporal and spatial information to properly associate the hits to clusters. The biggest challenge was presented in quantifying an efficiency parameter, since the truth data from the simulation does not directly include the truth clusters. It was found that limiting the cluster size to a threshold of 30 mm ensures that almost only clusters carrying valid timing information are included and also very few of those valid clusters are discarded. For this threshold the efficiency was estimated to be 86.7%.

This work could be carried on by implementing a tracking algorithm before the clustering algorithm. A next step also would be to determine how the multiple available timestamps for one particle are best combined to get the most precise timestamp.

Bibliography

[1] K. Arndt 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). ISSN: 0168-9002. DOI: 10.1016/j.nima.2021.165679. URL: http://dx.doi.org/10.1016/j.nima.2021.165679.

[2] CERN. *The Standard Model.* URL: https://home.cern/science/physics/standard-model.

[3] P. Eckert. "The Mu3e Tile Detector". PhD thesis. Ruprecht-Karls-Universität Heidelberg, 2015.

[4] R. Eichler and C. Grab. "The SINDRUM-I Experiment". In: *SciPost Physics* (2021).

[5] A. Blondel *et al.* "Letter of Intent for an Experiment to Search for the Decay $\mu \to eee$". In: (2012).

[6] Paolo Girotti. *Status of the Fermilab Muon $g - 2$ Experiment.* 2022. arXiv: 2202.11391 [hep-ex].

[7] C. Graf. "A Calibration Scheme for the Mu3e Tile Detector". MA thesis. Ruprecht-Karls-Universität Heidelberg, 2015.

[8] Wikipedia. *Geant4*. URL: https://en.wikipedia.org/wiki/Geant4.

[9] Wikipedia. *ROOT*. URL: https://en.wikipedia.org/wiki/ROOT.

List of Figures

List of Tables

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, Friday 11th March, 2022, [illegible] ....

E. Steinkamp