Studies of gaseous detectors and tracking reconstruction using Corryvreckan

Gaseous Detectors Development, Antti Lumppio

MATR310 Laboratory Exercises

Contents

1 Introduction/Theoretical Background

I worked in the Gaseous Detectors Development lab this Summer from 3.6 till 30.8. My work there consisted of two parts: studying the basics of gaseous strip detectors with some basic gain measurements, and trying to see if one could utilize a pixel detector framework Corryvreckan to do tracking reconstruction for our strip detectors.

1.1 Gaseous radiation detectors

Gaseous radiation detectors operate by generating and collecting electron-ion pairs in a gas inside the detectors sensitive volume. Depending on the structure of the device, either the electrons or ions construct the signal. There are many different types of gaseous detectors, and gaseous detectors that measure directly the ionization resulting from the passage of ionizing radiation can be categorized in three main groups: ion chambers, proportional counters and Geiger tubes. My work focused on new-age proportional counters called micropattern gas detectors (MPGD) invented first by Anton Oed in 1998, more specifically my focus was on triple Gas Electron Multipliers (GEMs). Micropattern gas detectors implement precise techniques used in the semiconductor industry, like photolithography and selective etching, where the name micropattern stems from [2] [7].

The signal in these micropattern detectors is usually induced by the electrons drifting towards anode and the ions drifting towards the cathode, where they are respectfully collected. The electric field between these electrodes is high enough such that recombination is not likely to happen. The signal is induced by the movement of the charges and is quite small if we measure low amount of charges migrating to electrodes. The induced signal can be amplified by producing more electron-ion pairs inside the gas, this can be done by accelerating the electrons via high electric fields, which increases the kinetic energy of the electrons. On their way towards the anode, the electrons collide with neutral gas atoms, and if the electrons have accumulated an energy higher than the ionization energy specific to the gas used, more electron-ion pairs are formed in the collisions. This process is repeated by accelerating the newly generated electrons, and they too will further create more electron-ion pairs, that will also repeat this process, called the **avalanche process** [2] [3].

The measured signal from this avalanche process, usually consists of two parts the higher electron peak and a slower ion tail. The electrons construct a much faster nanosecond scale signal with a high peak, due to electrons having a much higher drift speed than ions. Ions on the other hand, generate slow signal with a much lower peak value. This is visualized in Figure 1 [3].

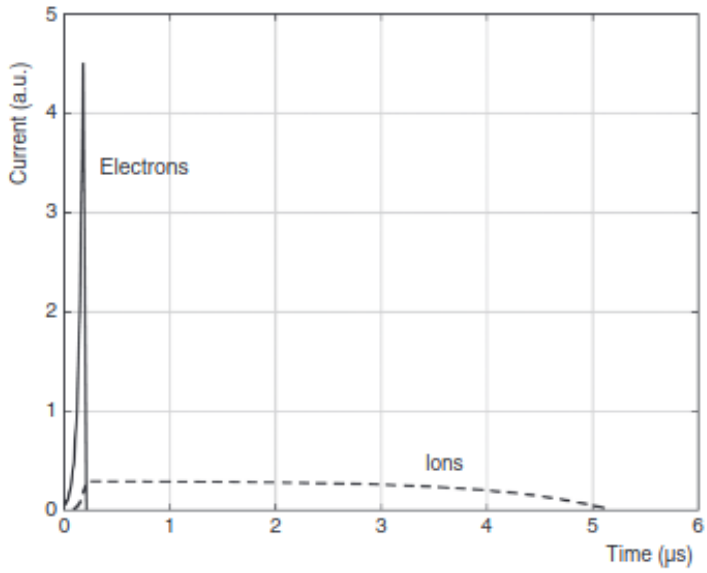

Figure 1: Illustration of the signal measured from an avalanche process. The signal consists of a high electron peak and a slow ion tail. The units on the y-axis are arbitrary and the units on the x-axis are on microseconds [7].

1.2 Gas Electron Multipliers (GEMs)

Gas electron multipliers (GEMs) are micropattern gaseous detectors created by Fabio Sauli in 1997. These detectors use pre-amplification stages which boost the gain of the detector, gain here referring to the multiplication of the electrons produced when compared to the original amount. In the GEM, between the drift and collection electrode, there is a thin polymer foil pierced with a high density of holes, typically $50 - 100mm^{-2}$, seen in Figure 2. The polymer foil is metal coated on both sides and when a large difference of potential is applied between the two sides, a high electric field is produced in the holes in the foil structure, called the GEM electrode, as seen in Figure 3. The electrons released by ionizing radiation in the sensitive volume above the GEM electrode drift towards the holes and acquire high enough energy to produce ionizing collisions with neutral gas atoms and cause an avalanche process. Substantial fraction of these electrons generated in the avalanche processes leave the multiplication region of the GEM electrode and move towards the anode or towards another multiplication layer. The signal in GEMs is collected with a 2D structure of charge detection anode strips, seen in Figure 4, and unlike other gaseous counters the signal is only from the electrons and has no contribution from the slow ions [3] [7].

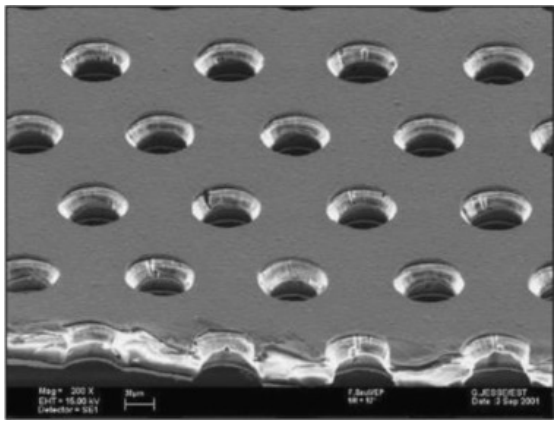

Figure 2: Electron microscope picture of the typical GEM electrode foil structure. The GEM electrode is $50\mu m$ thick and the holes pitch and diameter are $70\mu m$ and $140\mu m$ [3].

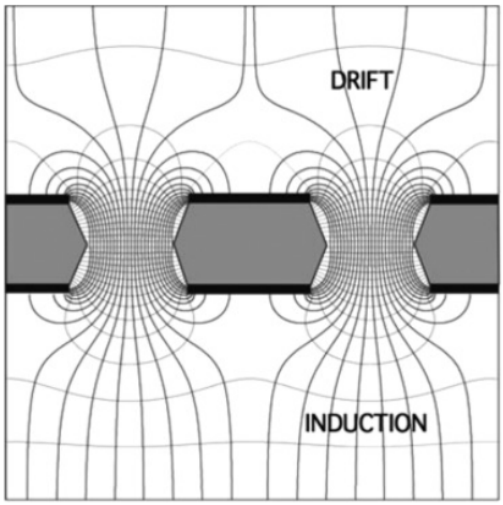

Figure 3: Illustration of the high electric field inside the holes in a GEM electode [3].

1.3 Corryvreckan

Corryvreckan is a pixel detector framework developed and maintained by the Beam Telescopes and Test Beams (BTTB) community. Corryvreckan is written in C++ and is an object-oriented, fast, flexible and lightweight test beam data reconstruction framework that uses a modular reconstruction chain. The framework has an adjustable algorithm for event building which can combine data

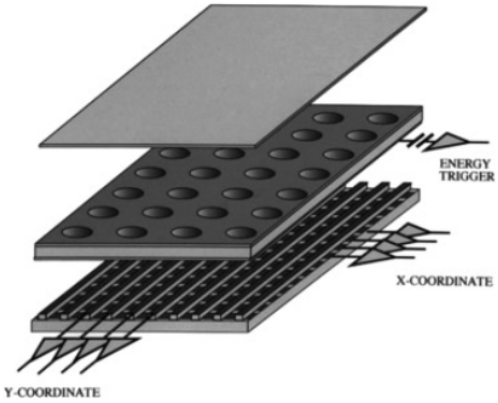

Figure 4: Schematic of a GEM detector with only a single amplification layer.

from different devices with different readout schemes. The framework strongly profits from the Allpix Squared Project, which is a generic pixel detector simulation framework [4].

In Corryvreckan there are four main objects: Event, Pixel, Cluster and Track. All of these objects are created using modules which use algorithms for specific tasks, for example the module can read in data from a specific readout system and create Corryvreckan pixel objects out of the data. These objects are stored in a clipboard, which acts as a momentary storage space for the objects between the modules. The Corryvreckan objects use ROOT TTree structure and the data used in Corryvreckan is read in and written using these objects. ROOT is a high-performance software written also mainly in C++.

In Corryvreckan the run is defined by the combination of the configuration file and the detectors file. The configuration file defines all of the modules and the possible variables used in the run. The modules are executed in linear order as they are written in the configuration file. The detectors file, like the name indicates, contains the information of all of the detectors used in the run. In a Corryvreckan run, one first has to create an Event object and store it in the clipboard. During the event, the data of that event is read in and from the data Pixel objects are created. This is typically done by the same module that does the reading. From these pixel objects Cluster objects are created and further from these clusters, Track objects are created. After the event has ended the clipboard is cleared for the next event. This process is illustrated in Figure 5.

My job was to see if we could use Corryvreckan to do particle track reconstruction for our gaseous strip detectors. What this means is that the software fits a function to probable path, which the ionizing particle took in our detector system, that can be called the telescope system. The fitting of the tracks can

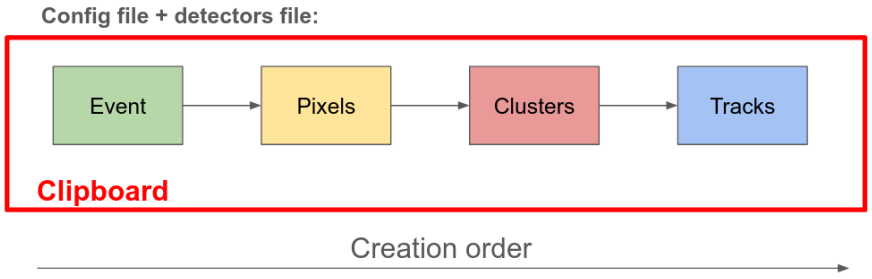

Figure 5: Illustration of a typical Corryvreckan run. The run is defined by the combination of the configuration and detectors file. Inside the run first an event is created and stored temporarily inside the clipboard. During the event the data is read in, and from this first pixels are created followed by clusters and lastly from clusters tracks are created.

be done using different functions, but by default Corryvreckan creates linear fits between the clusters in the first detector and the clusters in last detector. Different cuts can be applied to these track fits, and the tracks are made if the residuals, meaning the distance between the track fit intercept in the detector and the associated cluster, are within the cuts provided. The cuts used to make the tracks are usually the position and the time information of clusters used to make the tracks, but can also be some other constraints, like requiring each detector to have accepted cluster, a cluster within the cuts, on the track. Illustration of tracking can be seen in Figure 6.

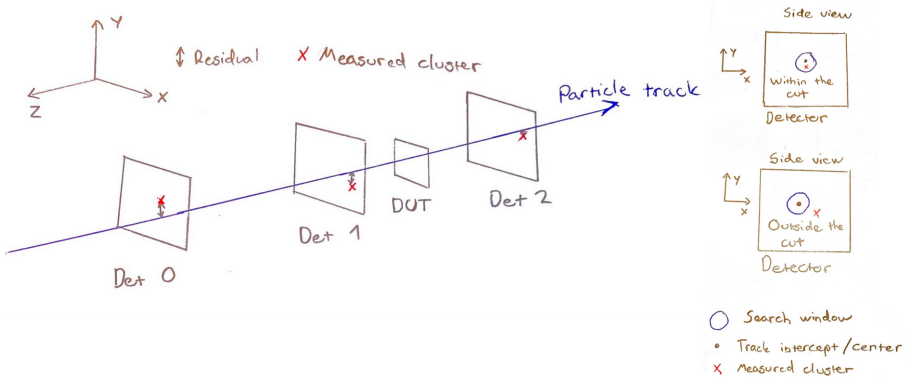

Figure 6: An illustration of particle track reconstruction inside a detector telescope. The track is made only if the clusters in the detectors are inside the spatial and time cuts provided. On the right, there is a rough sketch on first a accepted cluster and then a rejected cluster.

1.3.1 Pixel detectors vs. Gaseous strip detectors

Typical pixel detectors get the position information of the ionization process straight away with each "hit", as usually there are individual readout channels for each pad in the detector matrix. However, with gaseous strip detectors the signal comes from readout strips in different planes, meaning that the clusters has to be manually reconstructed using charge, timing or other information of the measured signals with strip detectors, and can this clustering can be done using different algorithms. The differences of these processes are illustrated in Figure 7.

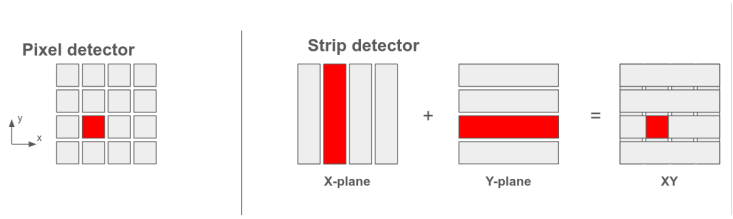

Figure 7: Illustration of the differences on how we obtain the position information with pixel detectors vs. gaseous strip detectors.

2 GEM gain measurements

To get familiar with the architecture that I would be working with, I did some basic coarse gain measurements with a triple gain layer GEM. In this detector we have 3 GEM electrodes and 3 readout strips, where the last strip layer, called the u-layer, is in a 45 degree angle respective to the X- and Y-plane readout strips. With this layer one can solve the ghost tracks that can happen with strip detectors. A schematic of this detector can be seen in Figure 8. I did three separate measurements of to determine the gain of the detector.

In the first measurement, I measured current from the bottom of the third gain layer GEM electrode with a Keithley 6487 picoammeter (Figure 10) and also the current from the readout strips was fed into a multichannel analyzer AmpTek MCA8000D (Figure 11). I used a Fe-55 source with an activity of $67.25MBq$. I varied the high voltage applied to the detector from $3900V$ to $4400V$ with $50V$ steps, and between each step, I recorded the current measured from the picoammeter and the MCA data with and without a radiation source. The high voltage device I used was a four channel CAEN Mod. N470. I did measurements with and without the source, so I could subtract the measured baseline current of the picoammeter. The MCA data was resolved using AmpTek's software DPPMCA and the picoammeter current data was analyzed using LabVIEW.

The gain of the detector can be obtained by dividing the measured current with the primary current, that is the current produced by the initial electrons which the ionizing radiation released in the gas. This means that the gain can be calculated from Equation 1, where f is the rate of the interactions in the gas, n is the number of primary electrons and q is the charge of the electron. I assumed the number of primary electrons to be approximately 200, for the gas mixture I was using Ar-CO2 (70/30). I approximated the rate of the interactions by diving the total counts from the MCA with the measurement time, $f = \frac{totalcounts}{measurementtime}$. From this calculation, I used the interaction rate for the clearest measurement which was at the highest voltage $4400V$, for which the rate was $f = \frac{totalcounts}{measurementtime} = \frac{125898}{30s} \approx 4.2kHz$. Using all of these we get the primary current to be $I_{primary} \approx 1.345 \cdot 10^{-13}A$.

$$G = \frac{I_{measured}}{I_{primary}} = \frac{I_{measured}}{fnq} = \frac{I_{measured}}{1.345 \cdot 10^{-13}A} \tag{1}$$

In the second measurement, I repeated the process of varying the high voltage applied to the detector, but now with finer steps going from $4150V$ to $4500V$, in each step increasing the voltage by $25V$. I again took measurements at each step with and without the $67.25MBq$ Fe-55 source, expect now I used three three Keithley 6487 picoammeters for the current measurements, reading the current from each of the readout planes, while at the same time the signals from these planes were also fed into the MCA8000D. The goal of this measurement was to

verify if the gain curve is similar to the curve we get from the current measurements from bottom of the last gain layer. In this the gain is calculated with the same equation, but the measured current is split into three parts (Equation 2).

$$G = \frac{I_{measured}}{I_{primary}} = \frac{I_{measured}}{fnq} = \frac{I_{x_{plane}} + I_{y_{plane}} + I_{u_{plane}}}{1.345 \cdot 10^{-13} A} \tag{2}$$

The third measurement I did, was done to find out when the readout planes have approximately equal charge sharing, while biasing the lowest plane, the u-layer. In this measurement I used the most stable high voltage setting I found in the previous measurement, which was at $4400V$. In this high voltage setting, I measured the current from each readout plane using Keithley 6487 picoammeters, while applying a bias voltage from $0V$ to $20V$ to the u-layer. The bias voltage was changed $1V$ at a time and at each step the currents were measured with and without the Fe-55 source. The currents were also fed into the MCA8000D.

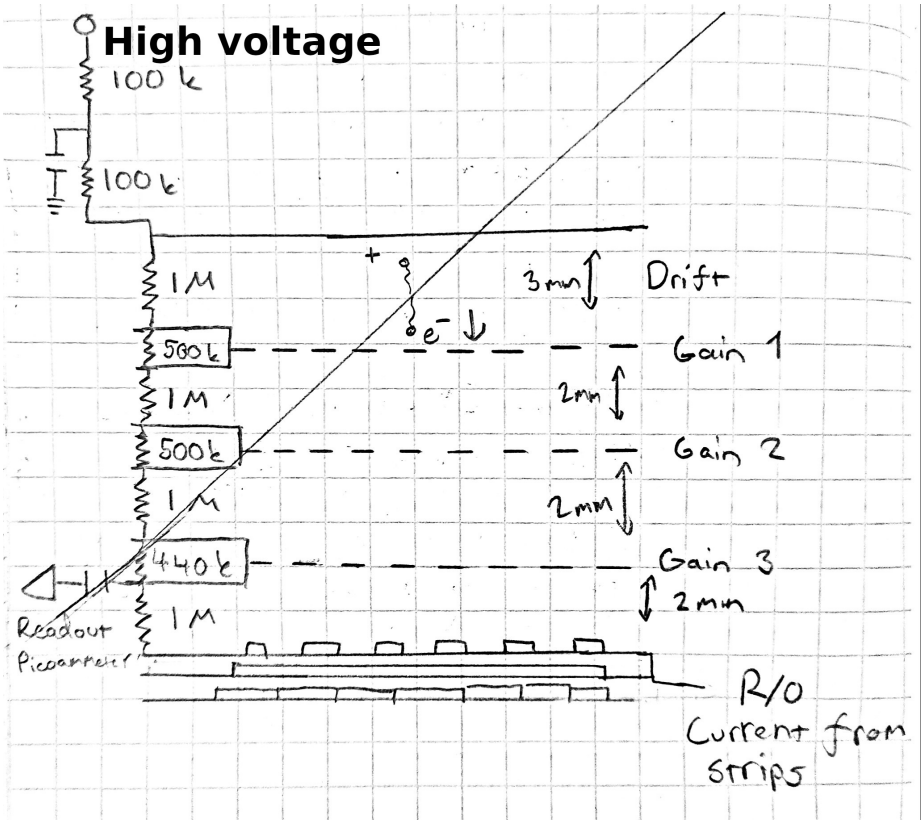

Figure 8: Schematic of the GEM detector I used to do gain measurements with a particle track illustration.

Figure 9: The GEM detector and the source stand can be seen in the red circle. The high voltage system can be seen in the green circle and the oscilloscope I used to verify that the signals were present can be seen in the purple circle.

Figure 10: The picoammeter I used for the current measurements, Keithely 6487.

Figure 11: The MCA I used to read the currents from the readout channels, MCA8000D.

3 Tracking reconstruction approaches in Corryvreckan

After getting somewhat familiar with the GEM detectors, I moved on to try to understand how this pixel detector framework Corryvreckan works. I did some reading and also did tutorials I could find. When I had an idea how to use Corryvreckan, I tried to use our own data in the software. There were no default software's that could our read in, so I had to make custom modules for them. Diego Figueiredo from the PPS-Test beam team helped me greatly to get started with this software.

The tracks in corryvreckan are usually done in a cycle, where we first create pixels, then clusters out of these pixels, and further tracks out of these clusters. I tried different variations of this reconstruction chain in Corryvreckan. This reconstruction chain is illustrated in Figure 12. Track reconstruction in Corryvreckan is done using the frameworks objects. Firstly the run is defined with the combination of the configuration file and the detectors file. In the beginning of the reconstruction chain, an Event object is made, and during this Event the data inside the event is read in. Pixel objects are created from the data read in, and out of these pixels, Cluster objects are made, and further from the Clusters, Track objects are made. All objects are stored in the clipboard, which acts as a storage space for objects, and this space can be accessed by the different modules. This reconstruction chain is continued until the run has ended, for example, when all of the data has been read in.

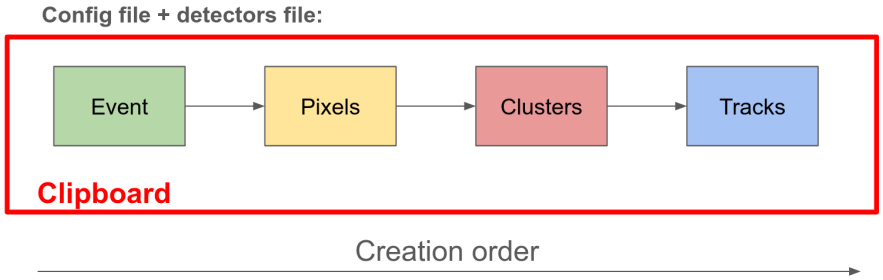

Figure 12: The typical reconstruction chain inside Corryvreckan. First an Event object is created and during this event the data used in the event is read in. Out of this data then Pixel objects are made. Continuing from the pixels, Cluster objects are made, and further then Track objects are made out of these clusters. This chain is continued until the run comes to an end.

To get accurate tracking data, the telescope has to be aligned first. This was done using default modules of Corryvreckan. The z positions of the detectors were calculated by hand, meaning orientational corrections were made only to the x- and y-coordinates of the detectors alonside the rotational corrections made by the software. For the alignment, we first do the prealignment for the telescope system using default [Prealignment] module. After this we start to

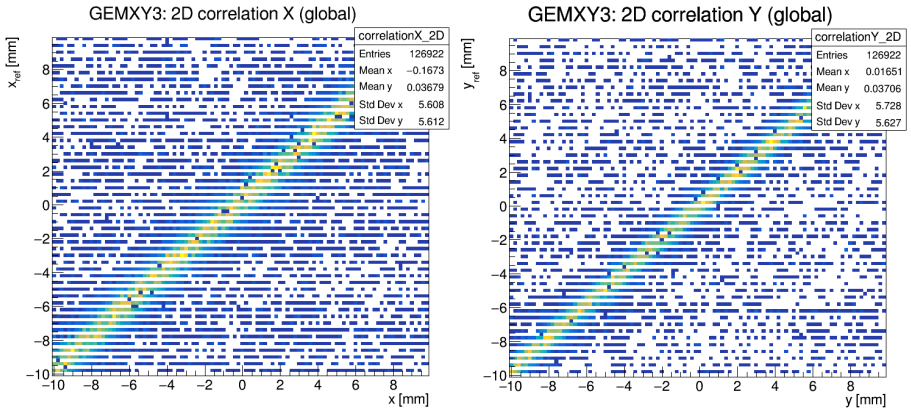

Figure 13: Example correlations gotten with the second approach tried in Corryvreckan (section 3.3). The alignment is good as we see a straight line from the origin.

do tracking using [Tracking4D] module and using this track data the detectors are aligned using [AlignmentTrackChi2] module. To align the detector under test (DUT), we need to first get some track data data for the DUT and this is done using [DUTAssociation] module. By default the detector under test is excluded from the track reconstruction telescope, as otherwise the tracks made would be biased towards the DUT. When the DUT has some track data from the [DUTAssociation] module, the DUT it has to be aligned, and this is done with [AlignmentDUT] module.

After all alignment procedures are done, the alignment can be checked by looking into the correlation plots. If the correlation plots show a linear line from origin, as seen in Figure 13, given by default module [Correlations], it means that the telescope is properly aligned and one can start taking tracking data.

3.1 APV25 vs. VMM3a

In the different approaches in Corryvreckan, I also used data from different readout systems, both used in the RD51 Scalable readout system (SRS). RD51 is a collaboration for development of micropattern gaseous detectors.

3.1.1 APV25

APV25 128-channel analogue application specific integrated circuit (ASIC) that uses waveform sampling to get the signal peak information [1]. The data we get from this readout chip goes first through DATE software to get the binary file. After this the file from DATE is analyzed in AMORE software which produces a ROOT file I will use in Corryvreckan. The ROOT file contains the reconstructed plane clusters and the reconstructed strip hits that are created by

AMORE. As the reconstruction in this readout system is done with waveform sampling, meaning no precise timing information, the data we get from AMORE is associated by eventIDs. Event in this context means a time slice, where the hits in the detector inside the telecsope system happened, and these are associated by number called eventID.

3.1.2 VMM3a

VMM3a is a 64 channel application specfic integrated circuit (ASIC), in which each front end channel combines the analogue part and the digital part into the signal processing [5] [6]. The data we get front this readout system is associated by time in nanoseconds. Readout chip data is first processed and a PCAP file is produced, and then from this file the vmm-sdat software creates the strip hits, individual plane clusters and matched XY-clusters [8]. Vmm-sdat produces a ROOT file that contains all of this information and this is further processed in Corryvreckan.

3.2 Approach 1: constructing clusters from cluster data

In this approach, I skipped the creation of Pixel objects in Corryvreckan, and used the reconstructed strip cluster data to create Cluster objects, as illustrated in Figure 14. First the Event object was created using default module [Metronome]. To read in data, I used a new module [ClusteringGeneric], which loads in APV25 AMORE cluster data. The AMORE clusters are not matched, so [ClusteringGeneric] only allows events in which there are only one hit per telescope, meaning a singular cluster in x and another cluster in y per detector. Out of these clusters, Track objects were made using default module [Tracking4D]. The detector in this approach was determined to have $256x256$ pixels with the pitches corresponding to the strip lengths we have in GEMs, so $400\mu m, 400\mu m$. The spatial resolution of the detector was determined to be one strip length in both dimensions, meaning $400\mu m$. However for the tracking absolute spatial cuts were set. The tracking cuts were 5 strip lengths $5 \cdot 400\mu m = 2mm$ in both dimensions.

3.3 Approach 2: constructing pixels from cluster data

In the second approach, I used again the default reconstruction chain in Corryvreckan, illustrated in Figure 5. First the Event object was created using the [Metronome], and the event that the [Metronome] creates has a fixed time duration, which is determined in the configuration file. For each event, the matched XY strip cluster data that was in the time interval of the Event object, was read in using a new [ClusterLoaderVMM3a] module. This time interval is also determined in the configuration file. From this external XY cluster data, Corryvreckan Pixel objects were made. After this, the "clustering" was again done using default [Clustering4D] module. The clusters that are made by the [Clustering4D] module will always be size 1, as we always have only one pixel. From

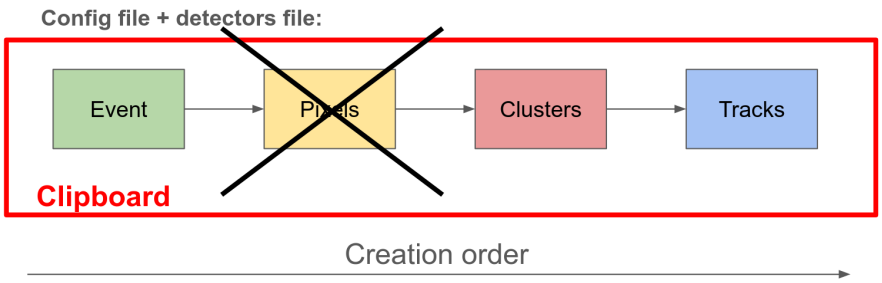

Figure 14: In this approach, I skipped the creation of pixels inside Corryvreckan. I first created the Event object, and then made Cluster objects out of this cluster data. Then from these clusters Track objects were made.

the clusters, Track objects were made using [Tracking4D] module. In this approach, I used the same detector configuration, as I had in the first approach. The detector had $256x256$ pixels with $400\mu m$ pitches in both dimensions and the spatial resolution also being $400\mu m$ (one strip length) in both dimensions. Again absolute tracking cuts were applied, with the cuts being 5 strip lengths $5 \cdot 400\mu m = 2mm$ in both dimensions.

With VMM3a data, the strip hit data could also be recovered using a new [VMM3aStripDataPreserver] module. This module uses the same data file we used to create tracks in the run. The module takes the timestamp of the Track object and reads in the strip hit and strip charge data of the clusters, which were used to make the tracks.

3.3.1 Matching APV25 clusters

With APV25 data, the clusters that we get from the AMORE ROOT file are not matched. Because the data is associated with eventIDs, I used only the charge information of the individual plane clusters to match them. I did this such that I make all possible XY cluster pairs and put them in a list, then from this list I choose the cluster pairs for which the sums of YX charge ratio between the planes is the lowest, so $\min(\sum \frac{charge_y}{charge_x})$. To prevent noise hits being paired with actual hits (because this would have the lowest charge ratio), I put a constraint that the mean YX charge ratio for the pairs made has to be in between 0.5 and 1.5. I try to clarify this method with the following example: Let's say I have two lists of clusters $X = x_1, x_2, x_3$ and $Y = y_1, y_2, y_3$. Now I fix the elements from the X list to stay in the same place, and generate all the different pairs by making pairs with Y list, while looping over all of the permutations of the Y list. This means that from the lists:

$$X = x_1, x_2, x_3 \quad \text{and} \quad Y = y_1, y_2, y_3 \tag{3}$$
$$\hookrightarrow (x_1, y_1),\ (x_2, y_2),\ (x_3, y_3), \quad \text{or} \tag{4}$$
$$(x_1, y_1),\ (x_2, y_3),\ (x_3, y_2) \quad \text{or} \tag{5}$$
$$(x_1, y_2),\ (x_2, y_1),\ (x_3, y_3) \quad \text{or} \tag{6}$$
$$(x_1, y_2),\ (x_2, y_3),\ (x_3, y_1) \quad \text{or} \tag{7}$$
$$(x_1, y_3),\ (x_2, y_1),\ (x_3, y_2) \quad \text{or} \tag{8}$$
$$(x_1, y_3),\ (x_2, y_2),\ (x_3, y_1) \tag{9}$$

From these example cluster pairs, we say that the equation (9) is chosen, because the sum of the YX charge ratios $\sum_{n=0}^{2} \frac{charge_{y_n}}{charge_{x_n}}$ is the lowest, with the constraint $0.5 < \sum_{n=0}^{2} \frac{charge_{y_n}}{charge_{x_n}} < 1.5$. This method of matching the clusters seems to work at least for one APV25 dataset I used, but I suspect it might have some trouble with noise hits.

3.4 Approach 3: constructing pixels from strip data

The third approach was an attempt to try to make and use strip detectors inside the Corryvreckan framework. In this approach, I split the previously used XY detectors into separate X- and Y-plane detectors. This was done by creating two plane detectors for each XY detector in the Corryvreckan detectors file. In the detectors file the X-plane detectors will have 256 columns and 1 row, and vice versa the Y-plane detectors will have 1 column and 256 rows. To make these plane detectors strip detectors, the pixel pitches had to be changed, meaning that one length of the pixel will be the size of the detector, $100mm$, and the other pixel length will the size of the strip, $400\mu m$. Again the Event object was created using [Metronome] module and VMM3a strip hit data was read in within the event time duration for each plane detector. From these strip hits Pixel objects were made and this time I let Corryvreckan do the clustering using again default module [Clustering4D]. And from these clusters, I use [Tracking4D] to create Track objects.

For the tracking to work, the spatial resolutions for the detectors had to be changed. We use a rough estimation that the spatial resolution will be the standard deviation for the a continuous distribution along a interval $[0, pitch]$. The variance for this continuous uniform distribution is $Var(X) = \frac{pitch^2 - 0^2}{12}$ and the standard deviation will be the square root of the variance $\sigma = \sqrt{\frac{pitch^2}{12}} = \frac{pitch}{\sqrt{12}}$, which is our spatial resolution estimate. This calculation assumes that there is little or no charge sharing between the pixels, that really doesn't apply to our case as we have multiple strip hits from which the clusters are created. This resolution was just used as a starting point to see if the tracking would work with this approach.

Because in this approach the XY detectors are separated into individual plane detectors mimicking strip detectors, the alignment in these runs is done a bit differently. In Corryvreckan, the reference detector has to be in the telescope, meaning that the reference detector will be either a X- or a Y-plane detector. Since we are referencing either the X or the Y plane, it doesn't make sense to rotational corrections, as using this will over correct the other plane detectors opposite to the reference plane. Thus only orientational corrections are applied in the alignment runs, and to check the quality of the alignment, I had to do two different runs, one with a X-plane detector as a reference and the other with a Y-plane detector as a reference. It doesn't matter which plane you use as a reference, as the orientational alignment of the whole telescope seems to work both planes, but it has to be checked with two different runs after the alignment.

4 Results

4.1 Gain measurements

I plotted all of the graphs using ROOT, except the MCA graphs.

For the first measurement, I got the expected exponential gain curve, seen in Figure 15. When this gain curve is plotted in log-scale in Figure 16, we see that the fit is linear, which confirms that the curve is exponential as $\log e^x = x$. These current measurements are from the bottom of the third gain layer and mean value of 10 picoammeter current points were used at each step. The errors for each of the points is the standard deviations of these mean picoammeter currents.

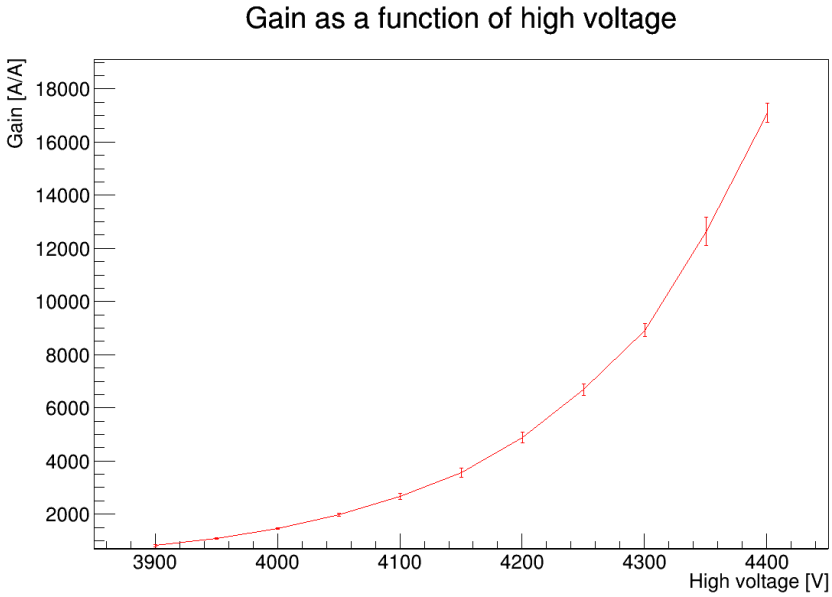

Figure 15: The gain curve for the first measurement as a function of applied high voltage. The voltage was varied between $3900V$ and $4400V$ with $50V$ steps. At each step the measurements were done with and without the source.

I also measured the current from the readout strips using a MCA. From this MCA data I produced a plot in the DPPMCA software. In the plot Figure 17, we see the characteristic energy curve for Fe-55 in argon, where we can spot the primary photopeak in which all of the energy that the passing photon ionized in the gas. There is also this smaller peak, which is only showing with $4300V$ and $4400V$. The smaller peak corresponds to the events, where the X-ray photon from Fe-55 is absorbed via the photoelectric absorption, that causes the gas atom to be in an excited state. The de-excitation process can go through two effects,

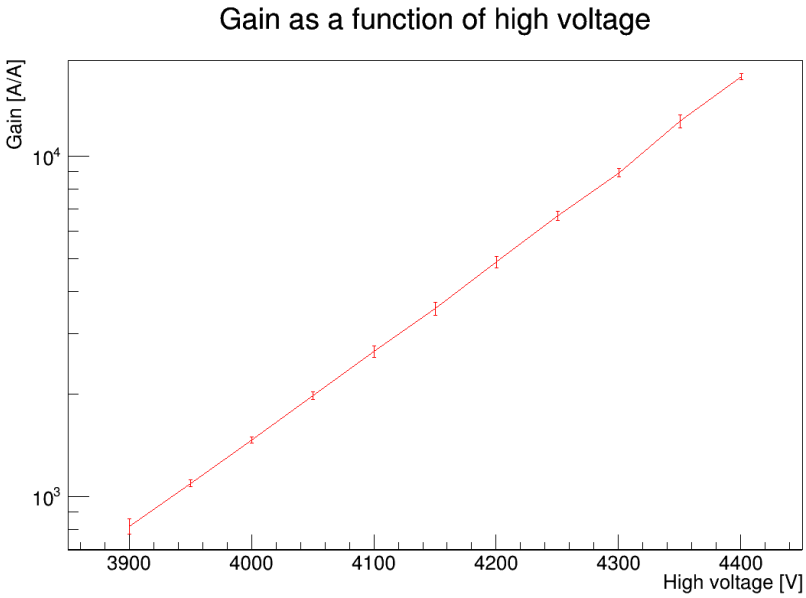

Figure 16: The first measurement curve plotted on a log-scale. This causes the fit to be linear as we have an exponential function.

either by the emission of an auger electron or through a emission fluorescence photon. This fluorescence photon can be converted far away from the primary interaction or escape the detection volume. If this photon escapes the detection volume, we are left with a energy deficit compared to the primary peak, which causes us to have a second peak, the escape peak. In argon the probability to that a fluorescence photon is produced after photoelectric absorption is around 8%

In the second measurement, I used finer step in between the high voltages from $4150V$ to $4500V$ with $25V$ steps. The currents in this measurement were measured from the readout planes with picoammeters. The error for each of the points is calculated using the standard deviation of the current measurements. The sum of the gains from each current measurement should correspond to the plot we get using the current data from the bottom of the last gain layer. When we compare the plots Figure 15 and Figure 18, we see that the gain curves are quite similar. The gain curves for the X- and Y-readout planes are quite akin, but the gain curve from the lowest u-plane is significantly smaller. This motivated the third measurement, where we try to find the bias voltage applied to the u-layer, that causes the charges to be equally shared between the readout planes.

In the third measurement the high voltage applied to the detector was fixed

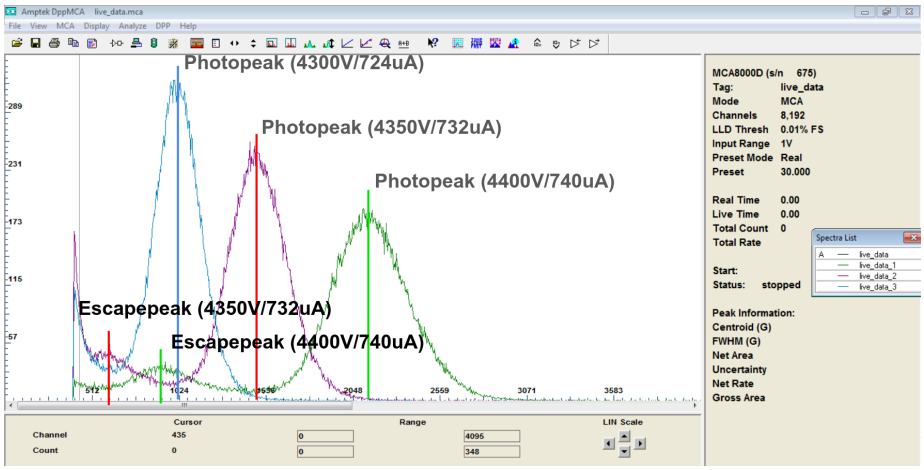

Figure 17: In this MCA graph, we can see three MCA measurement each at a different voltage. For all of the distributions, we can spot the main photopeak, but the escape peak can only be spotted with $4350V$ and $4400V$. The divider current are also marked in the peaks.

at $4400V$. In this measurement the bias voltage applied to the u-layer was varied from $20V$ to $0V$, with $1V$ steps. The currents from all of the readout planes was measured and plotted as a function of the applied bias voltage to the u-layer. ROOT automatically plots the points with the voltage ascending. I did two plots, one plot was with the raw signal currents that I got from the picoammeters, and the other plot was with the fractional signal currents, meaning that the signal current from the plane was divided by the total current from all the planes. In the fractional current plot, I had to use error propagation to get the error for each of the measurement points. The error was calculated with equation 11. From both of the plots Figure 19 and Figure 20, we can see that the charge sharing is almost equal, as the currents are within the error bars, when we apply a $19V$ bias voltage to the u-layer.

$$f = \frac{x}{x+y+u} \quad \longrightarrow \quad \Delta f = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 (\Delta x)^2 + \left(\frac{\partial f}{\partial y}\right)^2 (\Delta y)^2 + \left(\frac{\partial f}{\partial u}\right)^2 (\Delta u)^2} \tag{10}$$

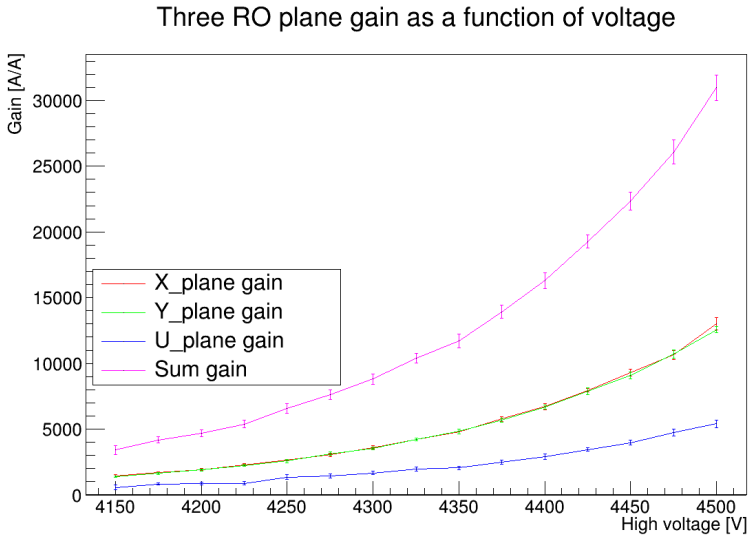

Figure 18: The gain measurements taken from each readout plane separately. We see that the gain curve is almost identical for the x and the y layer, but is significantly lower for the u-layer.

$$\hookrightarrow \Delta f = \sqrt{\left(\frac{1}{x+y+u} - \frac{x}{(x+y+u)^2}\right)^2 (\Delta x)^2 - \left(\frac{1}{x+y+u} - \frac{y}{(x+y+u)^2}\right)^2 (\Delta y)^2 - \left(\frac{1}{x+y+u} - \frac{x}{(x+y+u)^2}\right)^2 (\Delta x)^2} \quad (11)$$

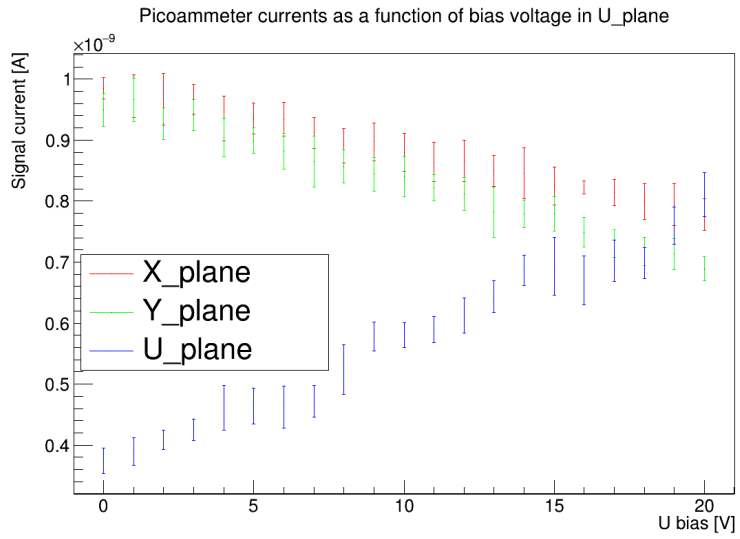

Figure 19: The picoammeter currents from each readout plane plotted against the bias voltage applied to the u-layer. The voltage applied was from $20V$ to $0V$ with $1V$ steps. We see that the charge sharing is almost equal at $19V$.

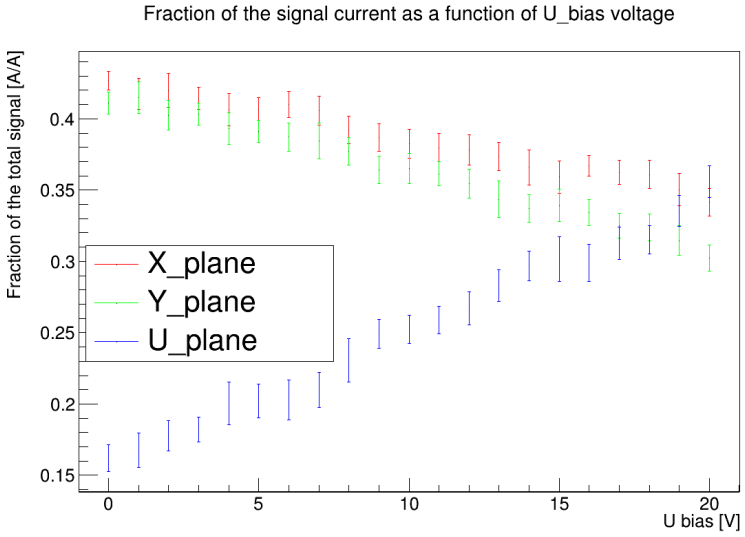

Figure 20: The fractional currents from each readout plane plotted against the bias voltage applied to the u-layer. The fractional current means the measured current from one plane divided by the total current measured over all planes. The voltage applied was from $20V$ to $0V$ with $1V$ steps. We see that the charge sharing is almost equal at $19V$.

4.2 Corryvreckan tracking results

With all three approaches in Corryvreckan, I was able to produce tracking for our detector systems. I used both APV25 and VMM3a data on all of the approaches, but in the first approach I used mostly utilized APV25 data. In the latter approaches I mainly worked with VMM3a data.

4.2.1 Approach 1

In the first approach, I used a module [ClusteringGeneric] that Diego Figueiredo had created. This module was a starting point, a test module, to see if Corryvreckan would be able to reconstruct particle tracks with our gaseous detector data. In this approach I used APV25 data, and I read in already reconstructed clusters into Corryvreckan. These clusters were created with AMORE, but the individual X- and Y-plane clusters were not matched in the AMORE ROOT file from which I read in the data. Because the clusters were not matched, [ClusteringGeneric] reads in only the events, where there is a single cluster on both planes, as then the matching of the clusters is not a issue. From this external cluster data Clusters were created in Corryvreckan, and further reconstructed tracks were made from these clusters. No pixels were created in this approach.

Using [ClusteringGeneric] and the reconstruction chain, seen in Figure 14, we are able to get tracking data. In Figure 21 we can see the clusters that are made in Corryvreckan. From these clusters tracks were made using absolute spatial cuts of 2000um in both dimensions, which corresponds to five strip lengths of the detector. The residuals of the tracks for the first detector in the telescope can be seen in Figure 22 and the intersects of the tracks for the same detector can be seen in Figure 23.

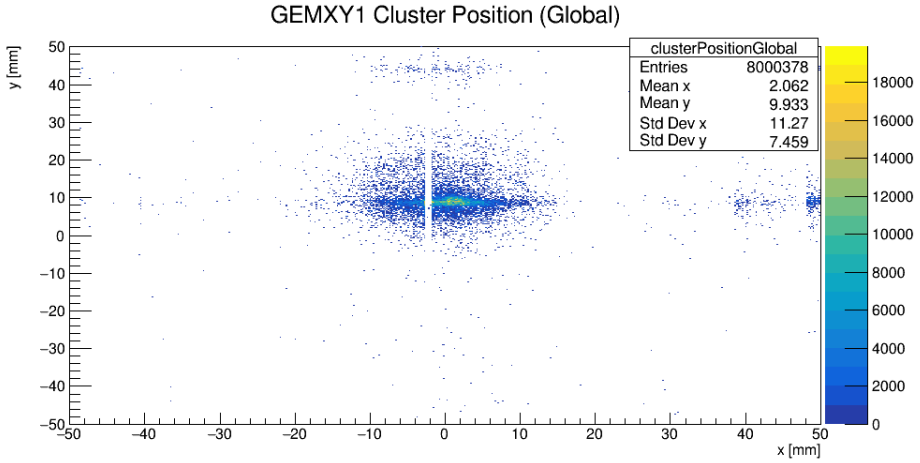

Figure 21: The position of the clusters in global coordinate system (the telescope coordinate system). Here I was using H8 test beam data with pions.

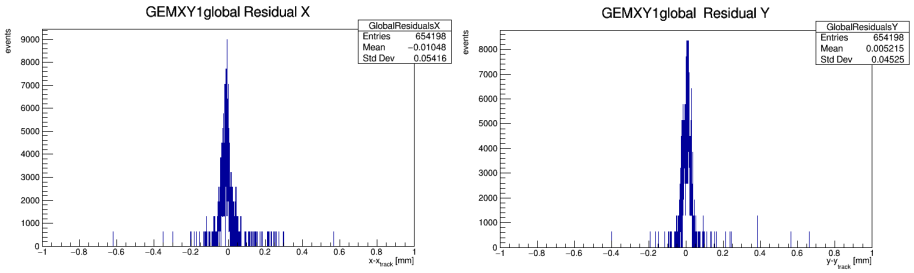

Figure 22: Residuals of the tracks for the first GEM detector in the telescope system using H8 test beam data with pions.

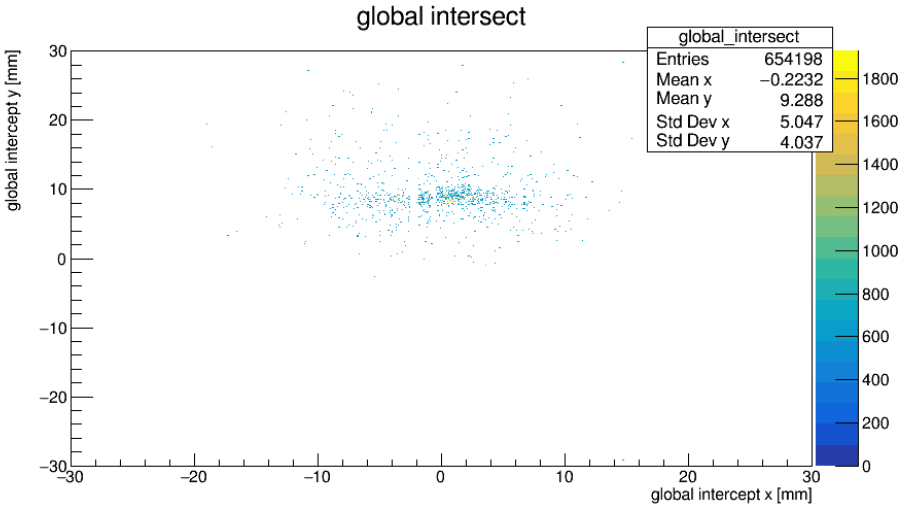

Figure 23: The global intersects of the first GEM detector in the telescope system using H8 test beam data with pions.

The tracking data from Corryvreckan can be extracted in a ROOT file in TTree format with a new [TreeWriter] module. Diego Figueiredo also created this module, but I modified it to give us the tracking data in a different format. From the ROOT TTree that the module creates, one can construct DUT residuals. The module also accounts for the events, where [ClusteringGeneric] won't read in any data, by making empty entries in these events to the TTree. The TTree branches and the possible DUT track residuals you could create with this data, can be seen in Figure 24. I didn't have a APV25 data with 4 detectors, so I couldn't do a proper DUT track residual plot.

We can get tracking data from Corryvreckan, by using only cluster objects, but also comes with its consequences. With this approach we lose the individual strip data of the clusters, like the strip charges or positions of the hits, which are part

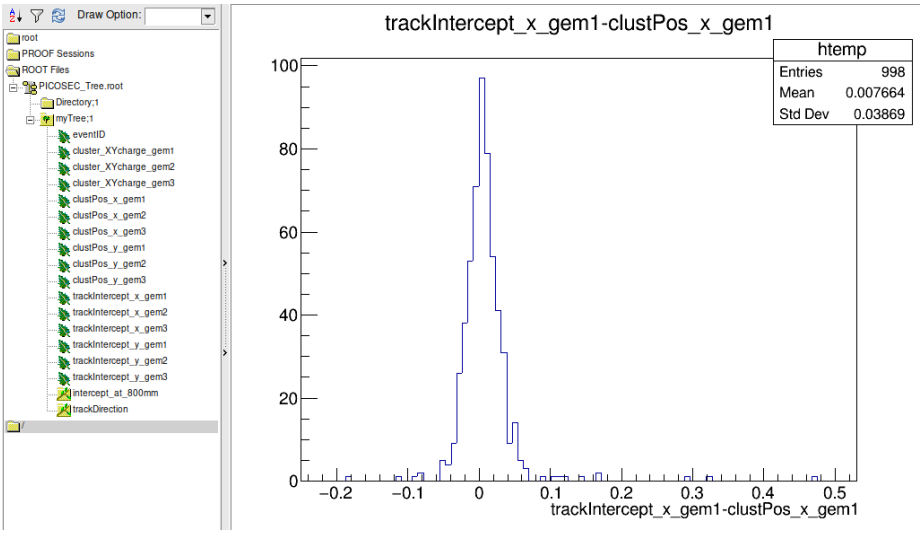

Figure 24: In this figure, we can see how one could possibly construct the DUT track residuals by drawing the track intersects in a detector from which the associated cluster positions have been substracted. In this plot however, the residuals are biased as the clusters are also used in the tracking.

of the clusters used to make Cluster objects in Corryvreckan. Only the overall charge, position and time of the cluster is preseved inside the software. Also by skipping the creation of the Pixel objects we lose some of the default analysis, that Corryvreckan would otherwise give, for example the [AnalysisDUT] module doesn't work without pixels. If we use this module without Pixel objects, we get a segmentation fault, as it is most likely trying to look for pixel data that doesn't exist.

This motivated the second approach, in which I try to create Pixel objects in Corryvreckan from cluster data.

4.2.2 Approach 2

The loss of some of the default analysis motivated to try to use the default reconstruction chain in Corryvreckan. In this approach, however, the Pixel objects were made using already matched strip detector clusters. With VMM3a the matching of the clusters, was done with vmm-sdat [8]. With APV25 data, I used an algorithm I explained in section 3.3.1 to match the plane clusters. From these Pixel objects, I do the clustering again using default [Clustering4D] module. This reclustering might cause a loss in the spatial resolution of the cluster position, as the cluster made in Corryvreckan in two dimensional space is assigned to be in the center of the pixel, given by strip positions of the cluster. These clusters will always have size 1 in Corryvreckan as only one pixel is used

to create them. The distribution of these clusters in one detector is seen in Figure 25. From these Cluster objects, tracks were made using [Tracking4D] module. As VMM3a data is associated with time, absolute time cut of $200ns$ was assigned for the tracking.

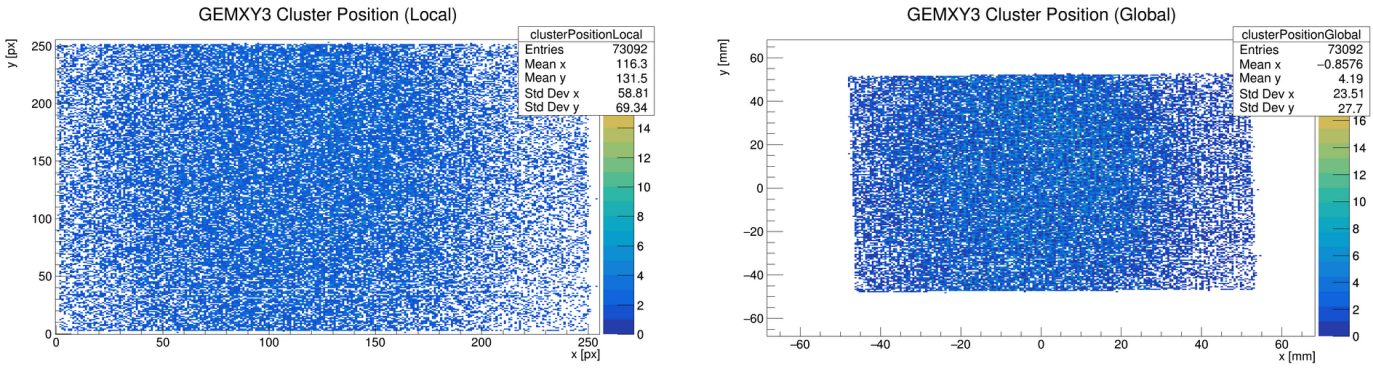

Figure 25: Distribution of the clusters that Corryvreckan creates from the pixels. Each of these clusters has size one, as the clusters are done from pixels, which in reality are reconstructed strip detector clusters.

The tracks we get with this approach give gaussian shapes. For the residuals, I used a double gaussian fit, as this accounts for the tails of the residual distributions, this is explained in more detail in the thesis of Lucian Scharenberg [6]. The residuals for one detector on a run done using VMM3a data can be seen in Figure 26. Also with this cluster data pixel approach, we are able to get the default analysis that was lost in the first approach. The [AnalysisDUT] module doesn't crash in this case, and we are able to get DUT residuals from this residuals, seen in Figure 27.

4.2.3 Approach 3

As the second approach of making pixels out of cluster data and reconstructing tracks out these seems to work, I decided to go even further and try to create strip detectors in Corryvreckan, splitting the plane of each detector into long pixel detectors. With VMM3a data, I used a strip hit data I get from vmm-sdat [8]. Using this data I do the clustering for each plane with the [Clustering4D] module with an absolute time cut of $500ns$ between the strip hits and allowing 1 missing strip in between the clusters, as this was also used in the matched cluster file I had for the same run. Comparing the clustering that Corryvreckan does with that of the vmm-sdat, we see that the distributions are quite similar Figure 28. There is around one percent error in between the clustering, which might be due to vmm-sdat having an extra constraint with the clustering, as in this software the maximum time difference between two consecutive hits was set to $150ns$, and this is not present in Corryvreckan. From these clusters, tracks are done using spatial resolutions of $\frac{pitch}{\sqrt{12}}$, meaning that the spatial resolution

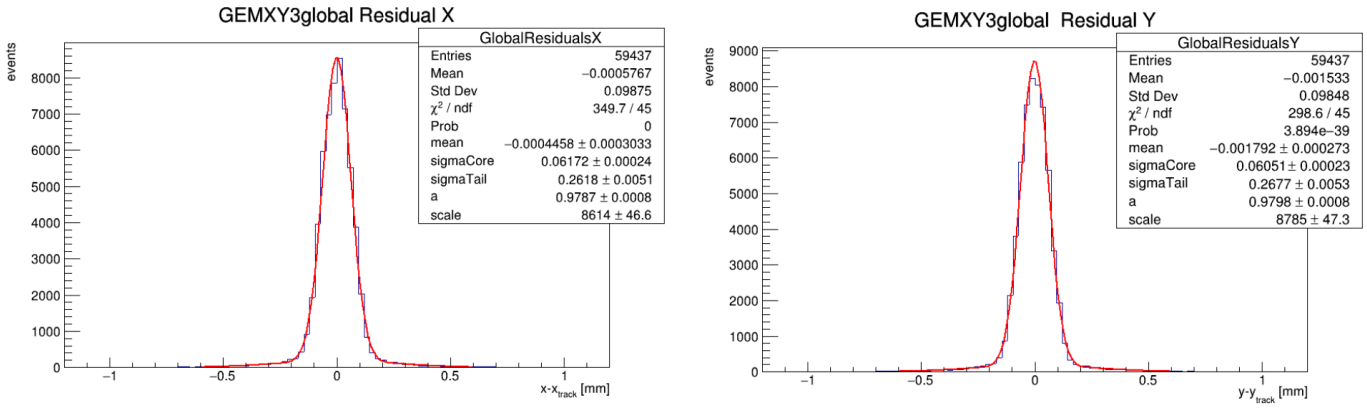

Figure 26: Track residuals for the third detector in the telescope. Double gaussian fits are plotted over the distributions as this accounts the tails better than a normal gaussian fit [6].

for the long coordinate will be $\frac{100mm}{\sqrt{12}} \approx 28.868mm$ and for the normal strip length $\frac{400\mu m}{\sqrt{12}} \approx 115.470\mu m$.

With this approach, much more tracks are made than in the previous approach. When comparing the tracks we get for the same run with the second approach and this third long pixel approach, we see that this long pixel approach gives around 28% more tracks for this particular run. Both residual distributions were plotted in Figure 29. The alignment for the tracks in the Corryvreckan correlation plots also works, but to check this, one has to do two different runs, one using a X-plane detector as a reference and the other run using a Y-plane detector as a reference. The correlation plots can be seen in Figure 30 from which we can see that the alignment is good for this particular run.

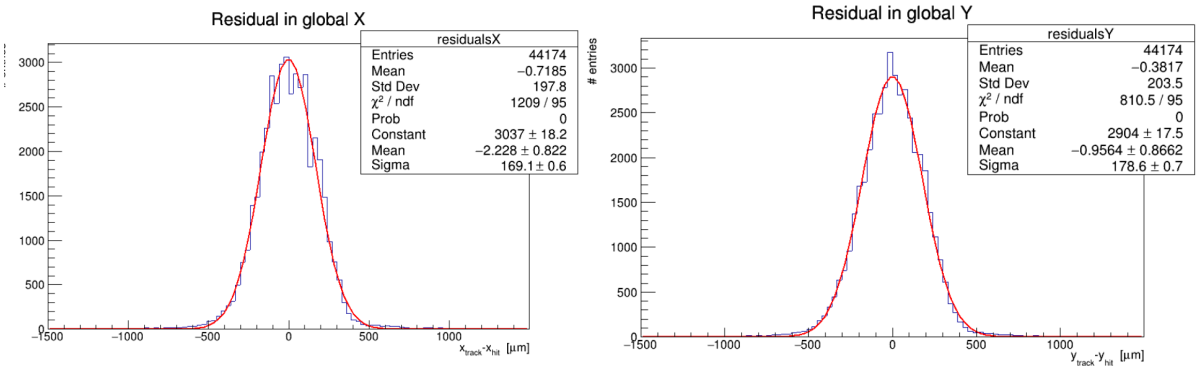

Figure 27: Detector under test residuals with VMM3a data. Double gaussian fits are plotted over the distributions as this accounts the tails better than a normal gaussian fit [6].

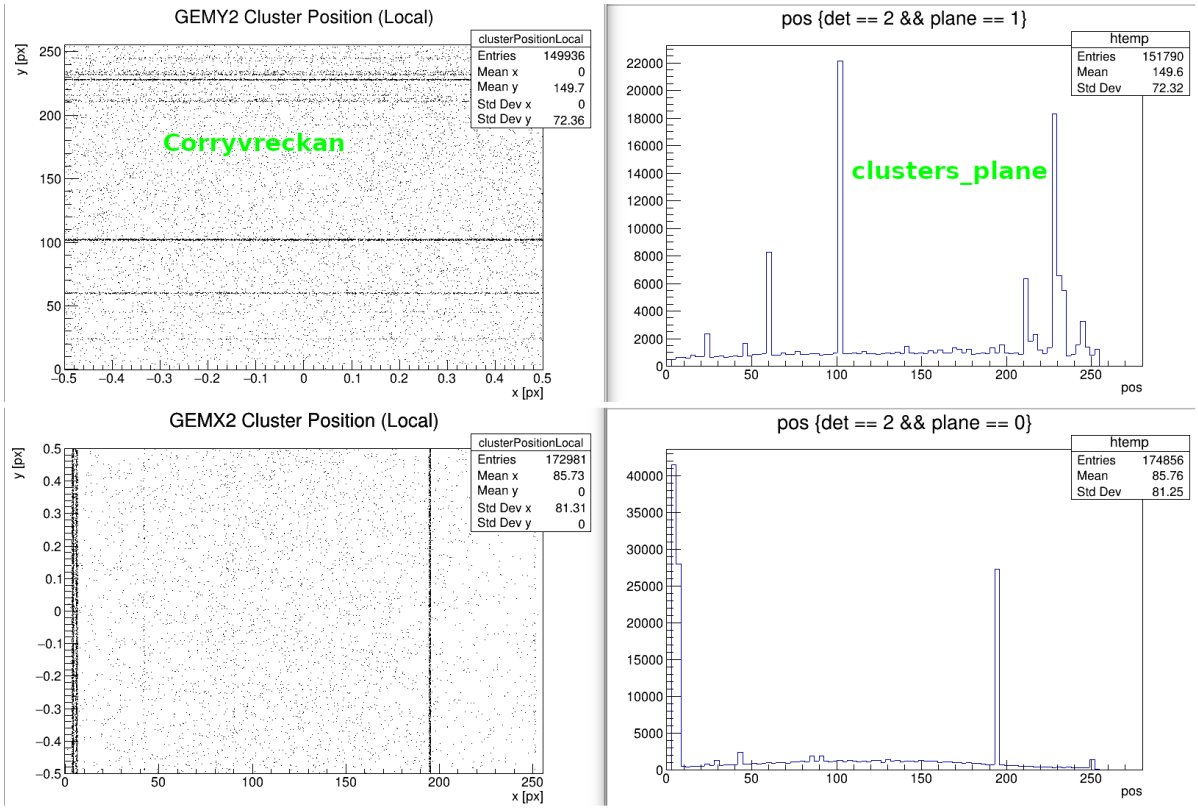

Figure 28: Comparison between the cluster distributions we get from Corryvreckan (on the left) and vmm-sdat (on the right). The distributions are quite similar, but there is a 1% difference in the amount of clusters created.

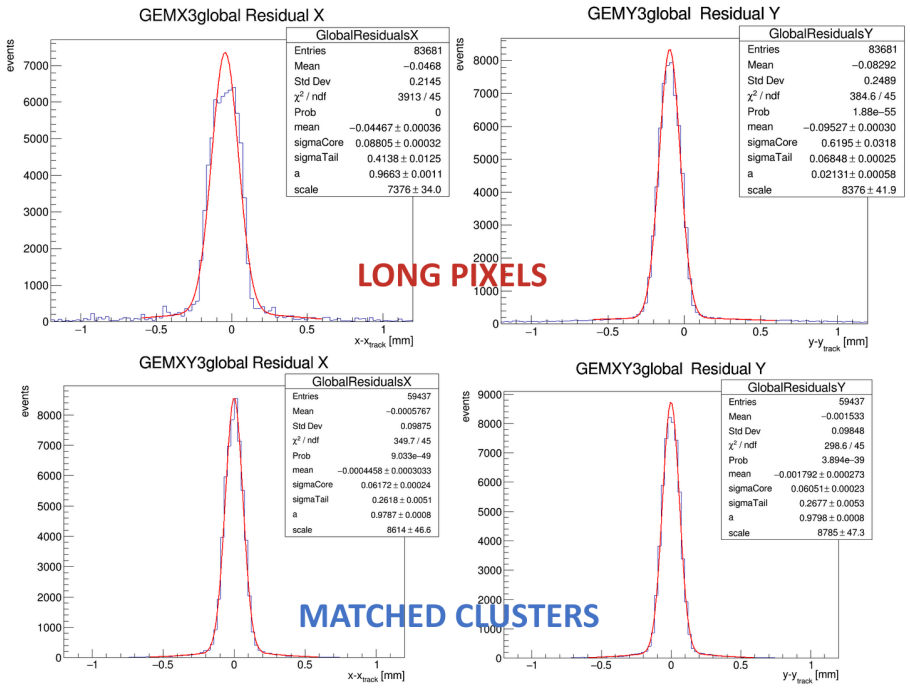

Figure 29: The comparison between the track residuals we get with the long pixel approach and with the cluster data pixel approach. The long pixel residuals are the top plots and the matched cluster plots are the bottom plots. We see approximately a 28% difference between the amount of tracks created. For the long pixel approach the distributions are also not aligned at zero.

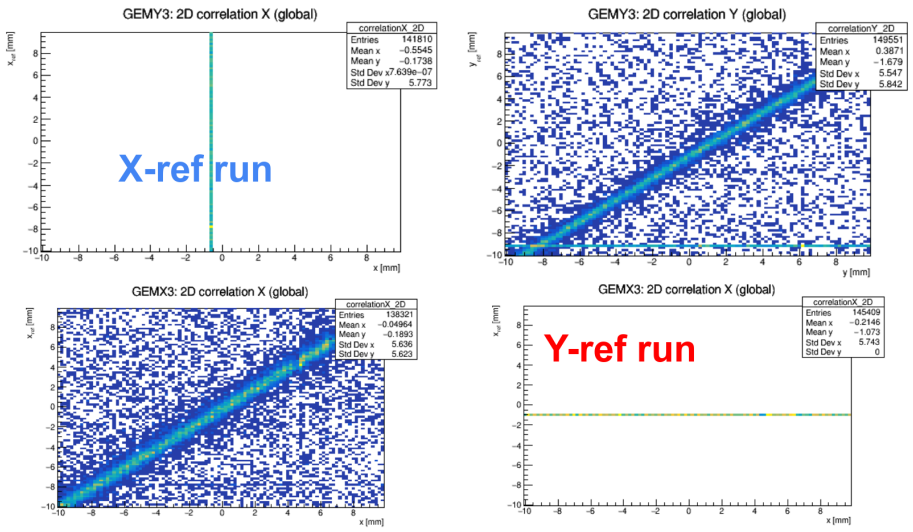

Figure 30: The correlations for the long pixel approach. To check that the alignment is done properly, one has to do two runs to check this. The left plots are from a run with a X-plane reference detector and on the right the plots are from a run with a Y-plane reference detector. Comparing these plots we see that the alignment is good for this run.

5 Conclusion

I got more familiar with the GEM detector through the gain measurements that I did with them. The plots I got from the measurements were as expected, the gain increases exponentially and is linear in log-scale Figure 16. Also the gain curves from the different readout methods were similar as comparing the plots (Figure 18 and Figure 15), we see that the data points for both measurements are within the error bars of each other. In the u-layer bias measurement, we saw that the charge sharing is almost equal for the X- and the Y-plane, but the U-plane has a much lower induced signal (Figure 19). This is also expected as it is the last layer and some of the electron will be already collected by the X- or the Y-plane, and thus there are less electrons moving to induce a signal. When a $19V$ bias voltage was applied to the u-layer, more electron were collected in this layer, and at this voltage the charge sharing between the planes was equal, meaning within the error bars of each other.

Tracking reconstruction in Corryvreckan is possible with gaseous strip detectors. I tried three different approaches inside the software, each with their own advantages and disadvantages. I was also able to create tracks with two different types of data, from the APV25 readout system, where the data is associated by eventIDs, and from the VMM3a readout data which is associated by time. Somewhat accurate tracking data can be acquired from Corryvreckan with two different approaches: using reconstructed cluster data and making Cluster objects out of these, or by using reconstructed matched XY clusters and making Pixel objects out of these. In third approach, where I tried to create strip detectors in Corryvreckan, the tracks that we got were not as good as with the earlier approaches.

The tracks that we get with the second approach using VMM3a data seem to be the best. The amount of tracks produced with this method is what one would expect for the runs I used, and the position resolution (sigma of the gaussian fits) from the residuals (Figure 26) are similar to the ones Lucian had produced in his thesis work [6]. Also with this approach one is able to get DUT analysis from the default [AnalysisDUT] module.

References

[1] M.J. French et al. "Design and results from the APV25, a deep sub-micron CMOS front-end chip for the CMS tracker". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 466.2 (2001). 4th Int. Symp. on Development and Application of Semiconductor Tracking Detectors, pp. 359–365. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(01)00589-7. URL: https://www.sciencedirect.com/science/article/pii/S0168900201005897.

[2] Glenn F Knoll. *Radiation detection and measurement*. John Wiley & Sons, 2010.

[3] Fabio Sauli. "The gas electron multiplier (GEM): Operating principles and applications". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 805 (2016). Special Issue in memory of Glenn F. Knoll, pp. 2–24. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2015.07.060. URL: https://www.sciencedirect.com/science/article/pii/S0168900215008980.

[4] D. Dannheim et al. "Corryvreckan: a modular 4D track reconstruction and analysis software for test beam data". In: *Journal of Instrumentation* 16.03 (Mar. 2021), P03008. ISSN: 1748-0221. DOI: 10.1088/1748-0221/16/03/p03008. URL: http://dx.doi.org/10.1088/1748-0221/16/03/P03008.

[5] Gianluigi de Geronimo et al. "The VMM3a ASIC". In: *IEEE Transactions on Nuclear Science* 69.4 (2022), pp. 976–985. DOI: 10.1109/TNS.2022.3155818.

[6] Lucian Scharenberg. "Next-Generation Electronics for the Read-Out of Micro-Pattern Gaseous Detectors". Presented 24 Jan 2023. Bonn U., 2022. URL: https://cds.cern.ch/record/2860765.

[7] Fabio Sauli. *Gaseous Radiation Detectors: Fundamentals and Applications*. Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology. Cambridge University Press, 2023.

[8] Dorothea Pfeiffer et al. *vmm-sdat — VMM3a/SRS Data Analysis Tool*. 2024. URL: https://github.com/ess-dmsc/vmm-sdat.