Studies of gaseous detectors and tracking reconstruction using Corryvreckan

Gaseous Detectors Development, Antti Lumppio

DRD1 Note draft

Contents

1 Introduction

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 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 from different devices with different readout schemes. The framework strongly profits from the Allpix Squared Project, which is a generic pixel detector simulation framework [2].

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 1.

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 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.

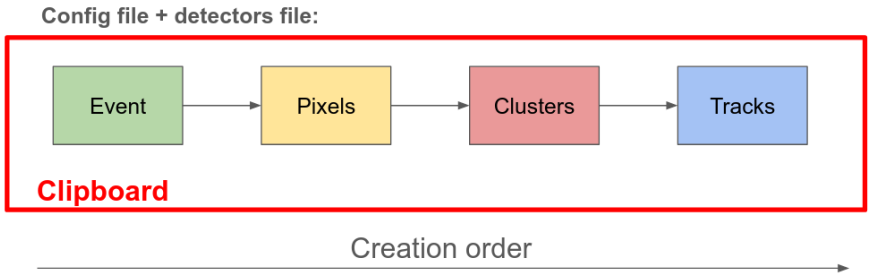

Figure 1: Illustration of a typical Corryvreckan run. The run is defined by the combination of the configuration and detectors file. Inside the run first an event is created and stored temporarily inside the clipboard. During the event the data is read in, and from this first pixels are created followed by clusters and lastly from clusters tracks are created.

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 2.

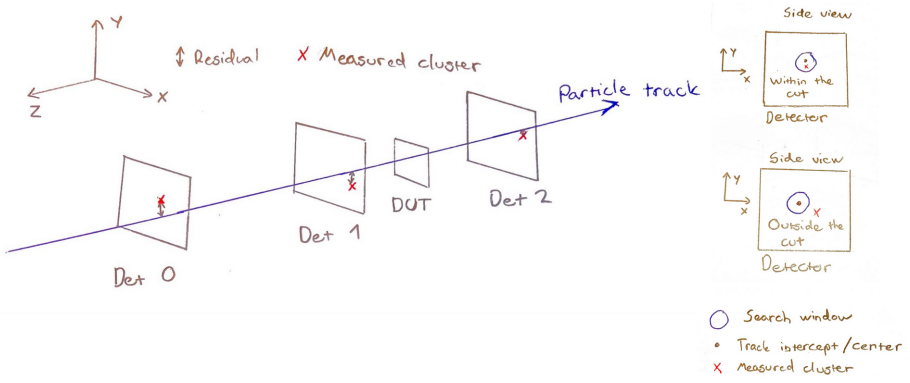

Figure 2: An illustration of particle track reconstruction inside a detector telescope. The track is made only if the clusters in the detectors are inside the spatial and time cuts provided. On the right, there is a rough sketch on first a accepted cluster and then a rejected cluster.

1.1.1 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 3.

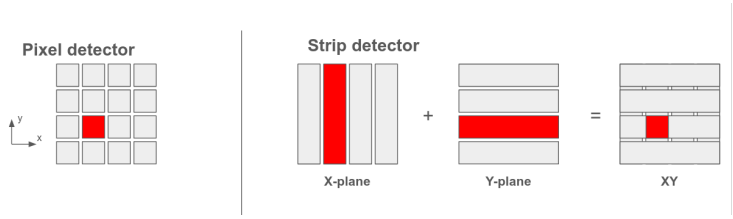

Figure 3: Illustration of the differences on how we obtain the position information with pixel detectors vs. gaseous strip detectors.

2 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 4. 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.

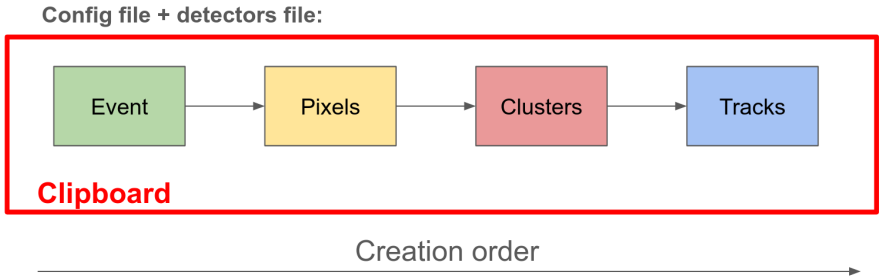

Figure 4: The typical reconstruction chain inside Corryvreckan. First an Event object is created and during this event the data used in the event is read in. Out of this data then Pixel objects are made. Continuing from the pixels, Cluster objects are made, and further then Track objects are made out 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

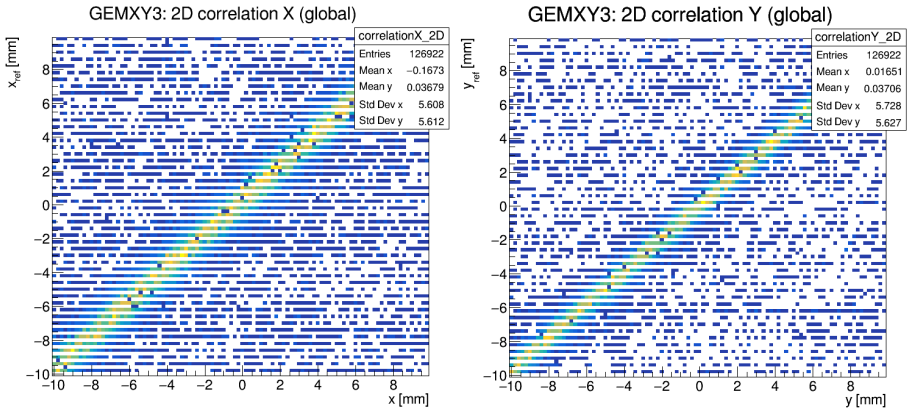

Figure 5: Example correlations gotten with the second approach tried in Corryvreckan (section 2.3). The alignment is good as we see a straight line from the origin.

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 5, given by default module [Correlations], it means that the telescope is properly aligned and one can start taking tracking data.

2.1 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.

2.1.1 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.

2.1.2 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 [3] [4]. 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 [5]. Vmm-sdat produces a ROOT file that contains all of this information and this is further processed in Corryvreckan.

2.2 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 6. 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.

2.3 Approach 2: constructing pixels from cluster data

In the second approach, I used again the default reconstruction chain in Corryvreckan, illustrated in Figure 1. 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

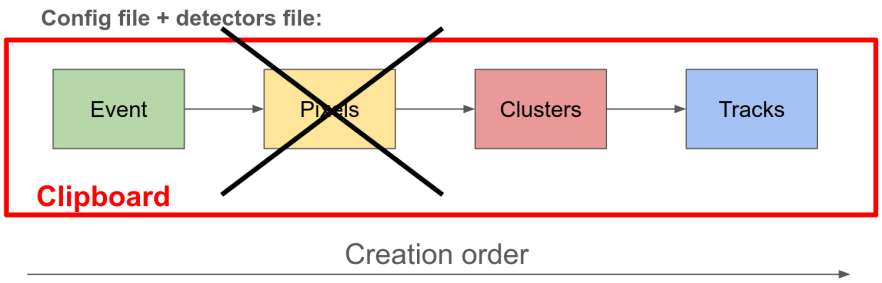

Figure 6: In this approach, I skipped the creation of pixels inside Corryvreckan. I first created the Event object, and then made Cluster objects out of this cluster data. Then from these clusters Track objects were made.

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.

2.3.1 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:

$$X = x_1, x_2, x_3 \quad \text{and} \quad Y = y_1, y_2, y_3 \tag{1}$$
$$\hookrightarrow (x_1, y_1),\ (x_2, y_2),\ (x_3, y_3), \quad \text{or} \tag{2}$$
$$(x_1, y_1),\ (x_2, y_3),\ (x_3, y_2) \quad \text{or} \tag{3}$$
$$(x_1, y_2),\ (x_2, y_1),\ (x_3, y_3) \quad \text{or} \tag{4}$$
$$(x_1, y_2),\ (x_2, y_3),\ (x_3, y_1) \quad \text{or} \tag{5}$$
$$(x_1, y_3),\ (x_2, y_1),\ (x_3, y_2) \quad \text{or} \tag{6}$$
$$(x_1, y_3),\ (x_2, y_2),\ (x_3, y_1) \tag{7}$$

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.

2.4 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.

3 Results

3.1 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.

3.1.1 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 6, we are able to get tracking data. In Figure 7 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 8 and the intersects of the tracks for the same detector can be seen in Figure 9.

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 10. 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 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

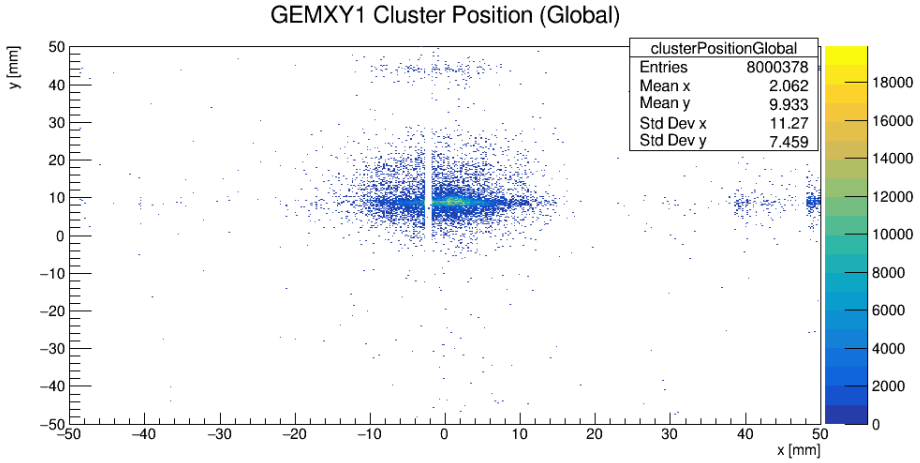

Figure 7: The position of the clusters in global coordinate system (the telescope coordinate system). Here I was using H8 test beam data with pions.

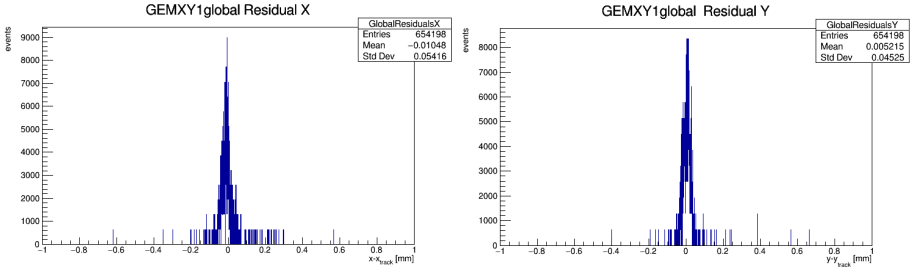

Figure 8: Residuals of the tracks for the first GEM detector in the telescope system using H8 test beam data with pions.

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.

3.1.2 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

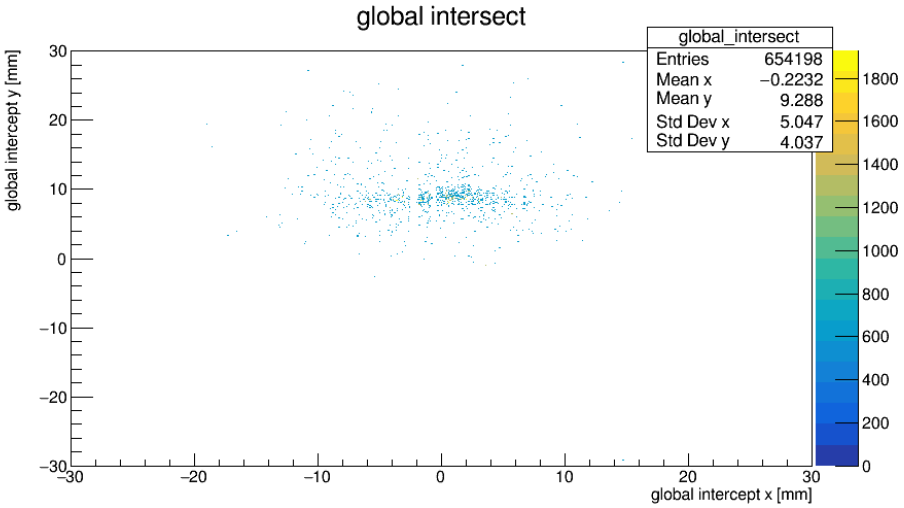

Figure 9: The global intersects of the first GEM detector in the telescope system using H8 test beam data with pions.

the matching of the clusters, was done with vmm-sdat [5]. With APV25 data, I used an algorithm I explained in section 2.3.1 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 11. 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.

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 [4]. The residuals for one detector on a run done using VMM3a data can be seen in Figure 12. 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 13.

3.1.3 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

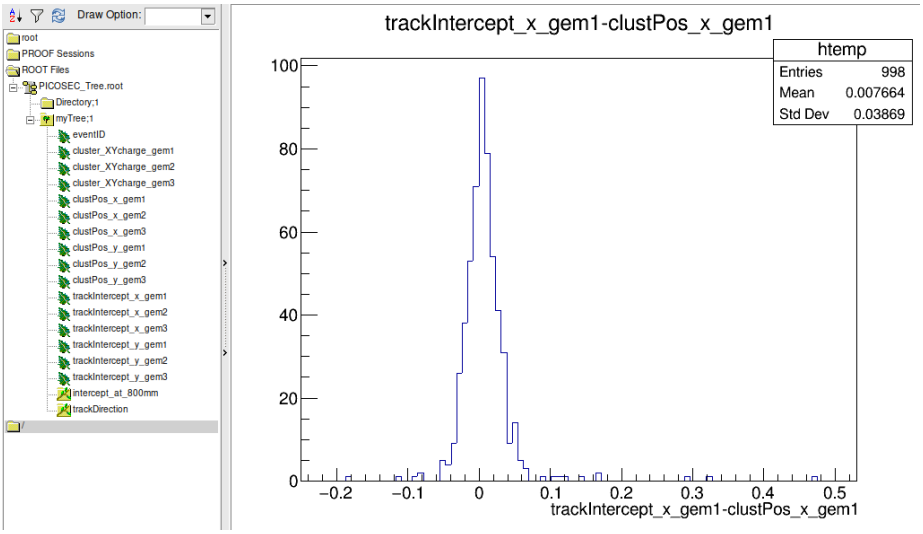

Figure 10: In this figure, we can see how one could possibly construct the DUT track residuals by drawing the track intersects in a detector from which the associated cluster positions have been substracted. In this plot however, the residuals are biased as the clusters are also used in the tracking.

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 [5]. 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 14. 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 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 15. 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

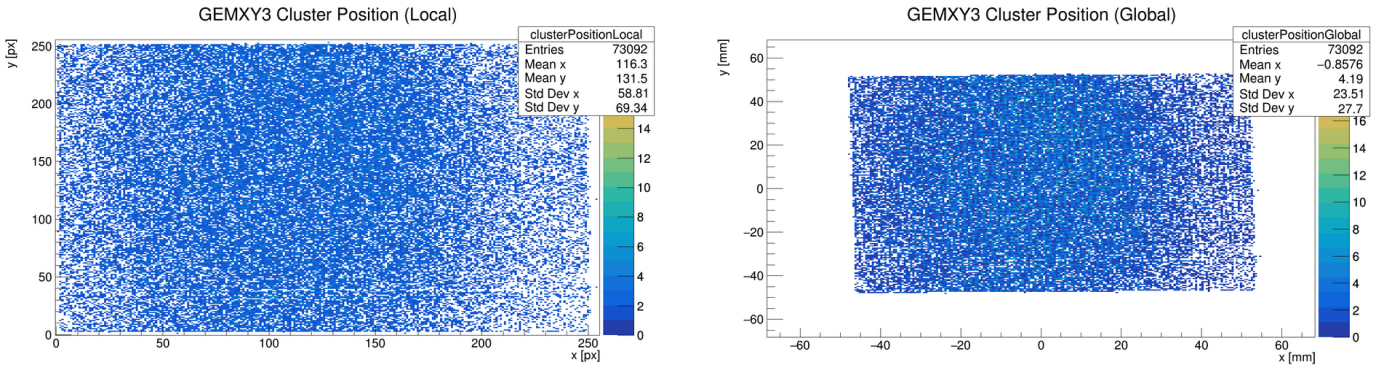

Figure 11: Distribution of the clusters that Corryvreckan creates from the pixels. Each of these clusters has size one, as the clusters are done from pixels, which in reality are reconstructed strip detector clusters.

detector as a reference. The correlation plots can be seen in Figure 16 from which we can see that the alignment is good for this particular run.

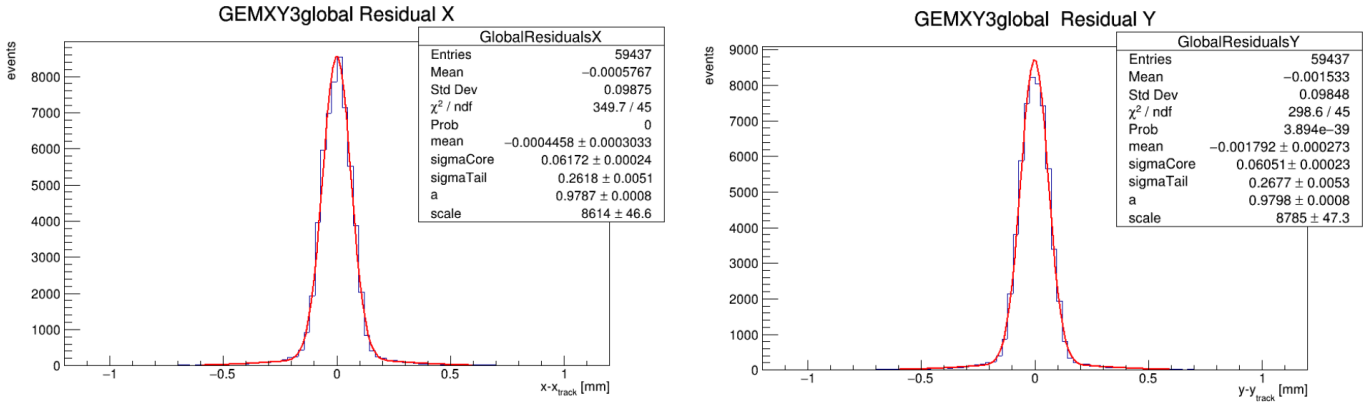

Figure 12: Track residuals for the third detector in the telescope. Double gaussian fits are plotted over the distributions as this accounts the tails better than a normal gaussian fit [4].

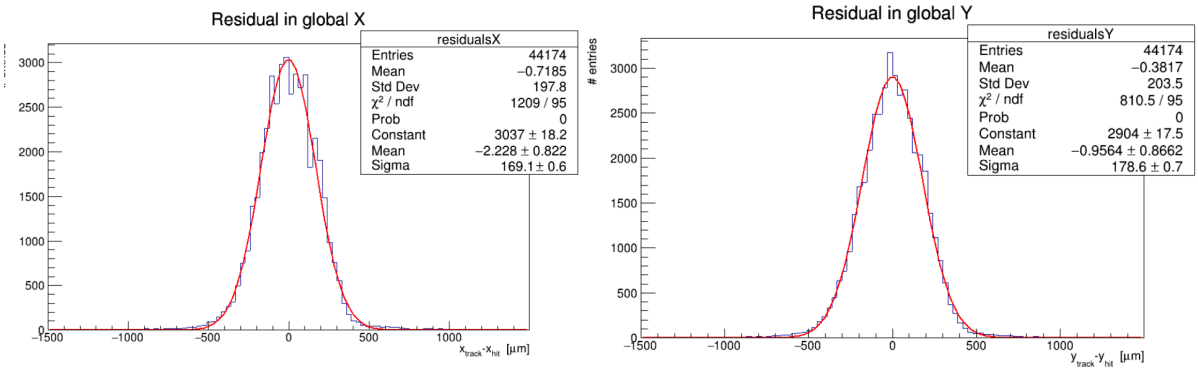

Figure 13: Detector under test residuals with VMM3a data. Double gaussian fits are plotted over the distributions as this accounts the tails better than a normal gaussian fit [4].

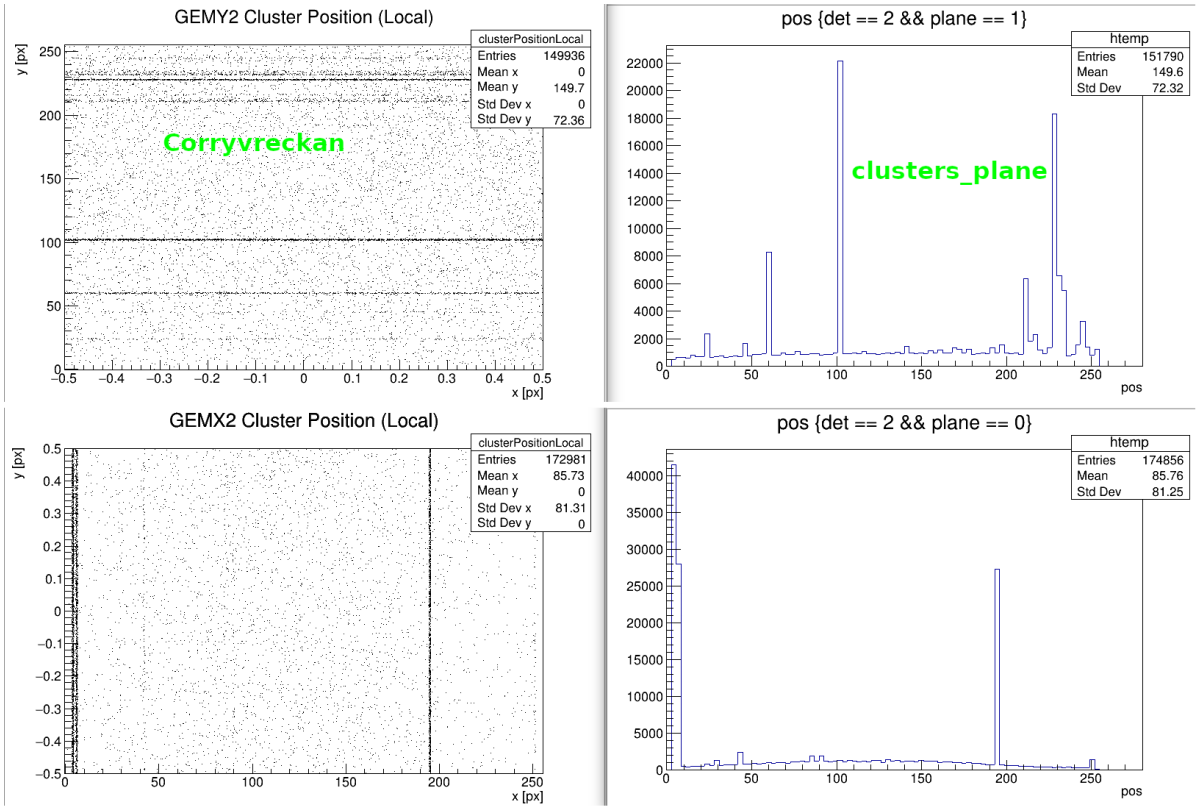

Figure 14: Comparison between the cluster distributions we get from Corryvreckan (on the left) and vmm-sdat (on the right). The distributions are quite similar, but there is a 1% difference in the amount of clusters created.

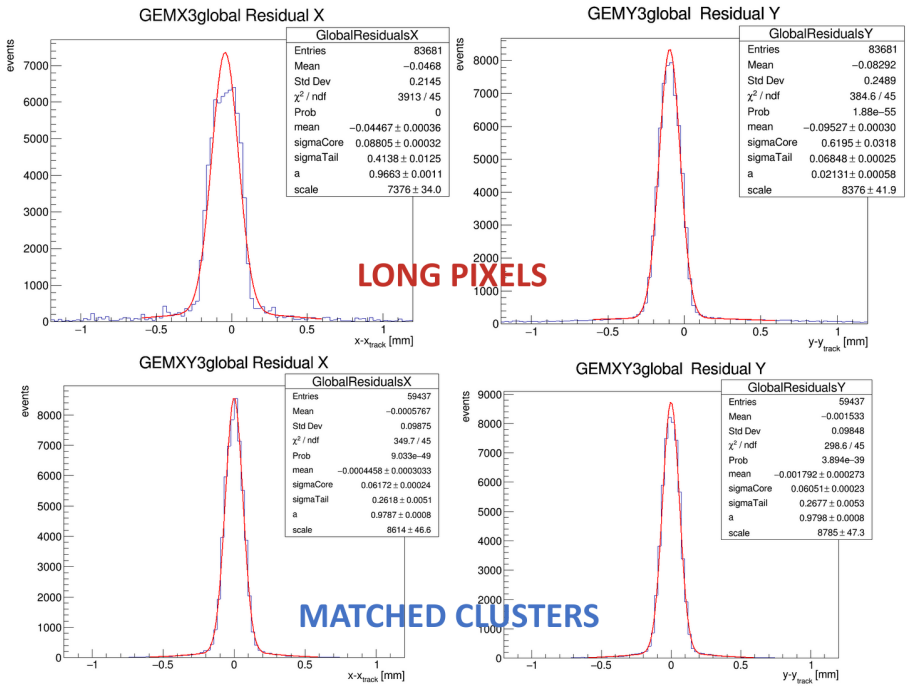

Figure 15: The comparison between the track residuals we get with the long pixel approach and with the cluster data pixel approach. The long pixel residuals are the top plots and the matched cluster plots are the bottom plots. We see approximately a 28% difference between the amount of tracks created. For the long pixel approach the distributions are also not aligned at zero.

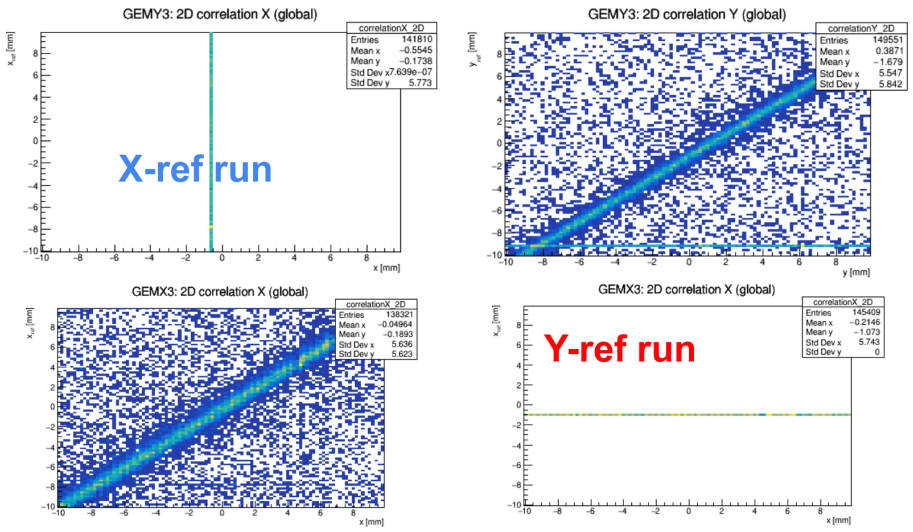

Figure 16: The correlations for the long pixel approach. To check that the alignment is done properly, one has to do two runs to check this. The left plots are from a run with a X-plane reference detector and on the right the plots are from a run with a Y-plane reference detector. Comparing these plots we see that the alignment is good for this run.

4 Conclusion

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 12) are similar to the ones Lucian had produced in his thesis work [4]. 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] 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.

[3] 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.

[4] 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.

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