

# Alignment of the MINOS FD

B. Becker and D. Boehnlein

November 5, 2004

*The results and procedure of the alignment of the MINOS Far Detector are presented. The far detector has independent alignments of SM1 and SM2. The misalignments have an estimated uncertainty of  $\sim 850 \mu\text{m}$  for SM1 and  $\sim 750 \mu\text{m}$  for SM2. The alignment has as inputs the average rotations of U and V as determined by optical survey and strip positions within modules measured from the module mapper. The output of this is a module-module correction for transverse mis-alignments. These results were verified by examining an independent set of data. These alignment constants on average contribute much less than 1 % to the total uncertainty in the transverse strip position.*

## Introduction

The alignment of the MINOS Far Detector has been considered before for SM1 in terms of a plane-plane alignment<sup>1</sup> and a module by module alignment<sup>2</sup>. These previous studies were used only for study of the alignment algorithm and procedure and not used to actually align the detector. The alignment of SM2 has never before been considered.

This note contains results from initial alignment of SM1 and a newer alignment of the entire FD. This contains both a detailed account of the procedure and the actual results. In addition, an estimate of both the resource requirements and justification of a more complex alignment procedure for future work are considered.

## Initial alignment of SuperModule 1

In June 2003, an alignment for SM1 was carried out. The alignment used the nominal positions of the scintillator in the module (no strip-strip corrections) and used an estimate of average rotations in the U and V view<sup>3</sup>. These values of the average rotation in the U and V view for SM1 are in UgliDbiScintPln in the 200007nnn series of SEQNO. The data that was used for this was the ‘alignment data’ that was taken in the summer of 2002 before the magnetic field was turned on. The alignment data was broken up into separate data sets. A set was used to generate the alignment constants and a separate set used to ‘verify’ the constants. Each of these data runs was processed on the FNAL cluster using R0.18.0. The alignment used SR track reconstruction for tracking and the alignment package for the actual alignment<sup>2</sup>. The alignment to generate the constants used 2 iterations and the verification used just the ‘zeroeth’ (no) iteration. The results are shown in figure 1 and figure 2. During the time between the first and second run the z-pitch model of the detector changed slightly, although, the alignment clearly worked. The runs used during this alignment are listed in Table 1.

Pass	Runs
Generation	5969,5994,6002,6011,6033,6079,6109,6122
Verification	5966.5991,6008,6017,6030,6077,6105,6107,6120

**Table 1:** This shows the runs used for both the generation and verification phase of alignment.

The results of this first alignment are presented in table 2. Pass here refers to pass 1 (generation) or pass 2 (verification). The number presented here are the numbers off the gaussian fits done in figure 1 and figure 2.

SM1 Number	View	Iteration	Mean (mm)	Sigma (mm)
1	U	1	-0.07±0.12	3.20±0.13
1	V	1	-0.06±0.13	3.79±0.13
1	U	2	-0.08±0.05	1.78±0.06
1	V	2	-0.05±0.05	1.85±0.05

**Table 2:** This shows the same results that are shown in figure 1 and figure 2. The mean is consistent with zero, although it is always negative. The sigma value improved from pass 1 to pass 2.

The results in table 1 suggest that the alignment of modules after alignment for this pass was about 1.8 mm. These constants were placed in the database in June 2003. The alignment constants were put in the UgliDbiScintMdl table in 200006nnn series SEQNO. These constants are only for SM1. The modified UgliDbiScintMdl and UgliDbiScintPln tables were used for the R1.0.0a batch processing of far detector data.

### Improved alignment

After the completion of the first alignment described above, it was realized that a new and improved alignment would have to be done. Besides aligning SM2 it was also realized that it was possible to attempt to generate an improved alignment for SM1.

#### *Improved alignment: Geometry overview for alignment*

Three database tables are of particular relevance for alignment, UgliDbiStrip, UgliDbiScintPln and UgliDbiScintMdl. UgliDbiStrip contains information of the strip position within a given module. UgliDbiScintPln has information relevant to an entire scintillator plane, in particular it contains information about the rotation of the plane. UgliDbiScintMdl holds information about the modules relative position on the scintillator plane. Many other database tables are relevant for reconstruction, however, these three tables are the only three tables the alignment process modifies, so they will be examined in detail. Every scintillator plane is made up of modules and every module consists of strips. This means that rotations in UgliDbiScintPln are applied to every module and every strip in that plane. In the same way, when a module has a transverse

offset, every strip in the given module also has a transverse offset.

#### *Improved alignment: UgliDbiStrip*

In the first alignment the nominal strip positions (inside a module) were used. However, the module mapper provided an improved estimate of strip position inside a module. The module mapper data was used to estimate the transverse strip position by averaging over the entire length of the strip (both leading and trailing edges). The new strip positions were estimated by taking data from the module mapper database (processed by Leon Mualem) and using the data to determine the position of all strips in a module relative to the first strip in module, which is assumed to be in the correct position. Even if the first strip is not in the correct position this is not a problem as it is only an absolute offset to the position of all the strips in that module and should be corrected by the module to module alignment.

However, this method of correcting for strip to strip variations has several problems. Some modules did not have reliable positions data available (this was not the mapper purpose), these modules were filled in with nominal positions. Some modules had single strips that were clearly unphysical because of some problem with the mapper. An unphysical strip position is defined as a 1mm difference between the measure position and estimated position of that strip based off the linear fit for that particular module. These strips were replaced by using a linear fit to fit all strips in the particular module and then replace the strip position with the extrapolated position from the linear fit. However, these problems were rare ~ 1% level. The strip-strip correction attempted to make no correction for strip-strip rotation inside a module, however, there was no evidence of this in the data<sup>4</sup>. The overall precision of the strip transverse position from the module mapper was estimated to be better than 1 mm (~few 0.1 mm). Figure 3 shows the difference between the measured and nominal slope (41.1 mm/strip) for a linear fit of strip versus transverse offset. The plot shows almost the total data set in addition to modules used in the vetoshield. The linear fits are done after the bad strip positions are cut and replaced. Figure 3 shows that for the most part the modules are very close to the nominal width. However, some tend to be slightly wider, while other modules are slightly more narrow. In particular figure 3 shows that the mean difference from nominal slope is 50  $\mu$ m/strip and the RMS of the distribution is 100  $\mu$ m/strip. The cause of this variation and of the distribution shape has not been studied and is outside the scope of this note. The resulting tables were placed in UgliDbiStrip database table with SEQNO from the 200011nnn series.

#### *Improved alignment: UgliDbiScintPln*

The rotations for the improved alignment were handled in the same way they were in first alignment. An average rotations measured from optical survey are used for the U and V planes. The only difference is the rotation now includes rotations for SM2. This means no attempt to correct for module rotation within a plane or rotation within a module are done. The rotation values used for this alignment are stored in UgliDbiScintPln with SEQNO in the 200011nnn series.

#### *Improved alignment: UgliDbiScintMdl*

The module by module alignment explained in this note, is accomplished by moving the module in the transverse position. This is the information stored in the UgliDbiScintMdl table. This is

the only table which values changes after alignment. The final alignment values are in UgliDbiScintPIn database with SEQNO 200012nnn. The series of SEQNO 200011nnn is based off code with an error and should never be used.

*Improved alignment: Overview*

The alignment was carried out with the idea that verification is very important. Verification here means that the alignment should show evidences that the alignment worked when the alignment is applied to an independent set of data. To do this, the alignment data sets were broken into two parts. The alignment data set was taken before the magnet was turned in a given SM. For SM1 this was from the summer of 2002 and for SM2 this was from the summer of 2003 (the particular files are listed in Run and Data Selection). The first part of the data set is used to generate the new alignment constants. Each iteration runs on the data from this data set. The second part of the data is the verification data set. This data set is not used until the final verification run. Each of the two data sets have approximately the same amount of data. The data sets were split up by alternating which run goes into which data set, this was done to try and limit any effect that drifted over the several days the data was taken. Even though this system reduces the statistical power of the alignment, it gives more confidence that the alignment results observed are robust.

*Improved alignment: Procedure*

The detailed procedure for alignment is given in Appendix A.

*Improved alignment: Run and Data selection*

The runs selected for the improved alignment are given in table 3.

Detector	Pass	Run Number
SM1	Generation	5966,5972,5991,6008,6011,6017,6020,6030,6033,6077,6079,6105,6107,6120
SM1	Verification	5963,5980,6005,6014,6027,6036,6081,6118,5969,5994,6002,6109,6497
SM2	Generation	16561,16557,16549,16545,16541,16533,16529,16525,16517,16513,16509,16501,16497,16493,16485,16481,16477,16588,16597,16601,16605,16613
SM2	Verification	16563,16559,16555,16547,16543,16539,16531,16527,16523,16515,16511,16507,16495,16491,16483,16479,16475,16595,16599,16603,16611

**Table 3:** This shows which runs were used for the generation and verification of the improved alignment for both SM1 and SM2.

Data Selection: Not all the data in these runs were selected. Only hits that were either in SM1 or SM2 were reconstructed (depending on which SM was being aligned). All tracks used for alignment had to be 20 or more tracklike planes long. Only single track events were considered.

To eliminate tracks that underwent a hard scatter, a requirement that direction cosine determined from the vertex must agree with the direction cosine determined over the entire track to 1.5 milliradian in the x, y and z direction was imposed.

*Improved alignment: Results*

The results of the alignment are presented in table 4 and in figures 4-21. The results in table 4 are a summary of the alignment.

SM Number	View	Iteration ( <i>pass</i> )	Mean (mm)	Sigma (mm)
1	U	0	0.256±0.104	2.565±0.141
1	V	0	0.025±0.132	3.142±0.177
2	U	0	0.209±0.112	2.663±0.176
2	V	0	-0.141±0.126	2.993±0.178
1	U	1	0.009±0.029	0.783±0.027
1	V	1	-0.027±0.027	0.832±0.027
2	U	1	0.000±0.027	0.737±0.033
2	V	1	-0.005±0.025	0.705±0.026
1	U	2	-0.033±0.017	0.469±0.015
1	V	2	0.016±0.016	0.431±0.014
2	U	2	-0.023±0.013	0.364±0.014
2	V	2	-0.024±0.012	0.345±0.013
1	U	Verification	-0.026±0.030	0.859±0.029
1	V	Verification	0.016±0.029	0.827±0.024
2	U	Verification	0.013±0.026	0.755±0.024
2	V	Verification	-0.037±0.027	0.765±0.024

*Table 4: This shows a summary of the alignment. The data shown are the results of gaussian fits done on the data shown in figure 4 to figure 11. All of the means are consistent with zero which is what you expect for random errors. The lower sigma values on the aligned data (Iteration 1,2 and verification) is evidence that the alignment worked. The two views agree well for a given SM and given iteration. The verification iteration suggests that the misalignment in SM1 is about ~ 850 m and about ~ 750 m in SM2.*

*Histogram Explanation and analysis:*

There are two types of histograms shown in the results section. There are the type of histograms shown in figure 4-11. These histograms show the mean residual of all U/V plane modules in either SM1/SM2. A gaussian fit is done to the distribution in order to quantify the quality of the alignment. All of the distributions have tails, but for the most part the gaussian model seems to fit the data well. The values in table 2 and table 4 are the results of the parameters determined from the gaussian fit. The histograms in figure 12-21 show the exact same data as the histograms in figure 4-11. The only difference is these figures use a color code to show the information. The histogram at the top of each figure is the mean residual for all (Not just U view) modules in SM1/SM2 as a function of plane number versus module number (position). The color scale is set to  $\pm 15$  mm for most histograms. The verification is shown both at  $\pm 15$  mm and  $\pm 3$  mm. The bottom histogram in each figure is the RMS value of the residuals as a function of plane number and module number. In these figures the RMS is much smaller than the mean. The fact that residual and RMS distribution get worse at the edges involves poor acceptances in these sections of the detector.

#### *Improved alignment: Secondary verification*

A possible secondary way to verify the alignment is to use part of the magnet on data set. Although this is independent data set with a much larger sample of data, it is not clear that this is the right thing to do. At the time of this study the magnetic field has not undergone final calibration, this could in principle introduce bias that might be very difficult to understand. On the other hand, the data with a magnet on is what the vast majority of the analyses will use. The data used for analysis was runs 25499,25502,25508,25511,25514,25517,15520,25523 (all sub-run 3) processed with R1.7. These 8 runs were ran with three different set of alignment constants. First these were run on the 'Nominal' geometry, before alignment. Second they were run using the constants produced from the first iteration of alignment. Thirdly, they were run using the alignment constants from the second pass. Several quantities were examined with these with these three runs. The reduced  $\chi^2$  distribution is shown for all tracks, short tracks ( $\leq 20$  tracklike planes), long tracks are shown in figure 22-24. Figure 25 shows a profile histogram of reduced  $\chi^2$  versus number of tracklike planes. All the figures have the same format, SM1 is on the left side, SM2 is on the right side. The top plot is the nominal geometry, the middle plot is based off the first iteration and the bottom plot is based off the second iteration. The middle plot has less data because run 25499 was not used because of technical issues. However, the shape of these distributions are very similar so cannot change the overall result. The first thing that is observed is that each step of alignment improves the  $\chi^2$  distribution, both the mean and width decrease. This is strong independent evidence the alignment worked. The  $\chi^2$  distribution is still not fully understood, the short events have an unusually small value. There is small difference between SM1 and SM2. However, it is unclear that the difference can be explained by the  $\chi^2$  distribution alone. The longer the track, the higher average  $\chi^2$ . The cause of this is not fully understood, one possible suggestion is multiple scattering, although there is no independent evidence for this. This is a topic for future study.

#### *Alignment and additional iteration*

The alignment only used two iterations. The use of more iterations was considered but it was not used. The reason is that the alignment was already very good. Since the verification is consistent with the first pass and not the second pass, any improvement from a third or higher iteration

would seem to be non-physical. This also suggests that the alignment is limited by systematic and not statistics uncertainty. Given the way rotations are modeled, this is not a surprise.

#### *Acceptance of tracks used for alignment*

Figure 26 and figure 27 show the number of tracks per module and tracks per strips used in the alignment for SM1 and SM2 (U and V view just not U view). The tracks per strip show that the detector acceptance drops off at the corners and the front and back of the detectors. The plots are from iteration zero of SM1 and SM2.

#### *Improved alignment: Conclusions and Open Questions*

The alignment of the FD was successful. The detector alignment was improvement in a verifiable manner and the final results suggest that misalignments are known to  $\sim 850$  m for SM1 and  $\sim 750$  m for SM2 . The results might be systematically limited with the current set of assumptions. A way to get around this limitation would be a true strip-strip alignment, however, that is very large undertaking for very small gains (See Appendix B and C).

### **Conclusions**

The alignment of the MINOS FD was a success. The post verification estimates of misalignments in SM1  $\sim 850$  m and SM2  $\sim 750$  m are improved over the nominal values in the database. However, there are some lingering questions which future studies should consider addressing. First, there is the question of whether SM1 and SM2 are systematically limited? Secondly, there is the question of what improvements (if any) can be made. Since there is not enough data in the alignment data set (even without splitting the data set in two) to do a magnet off ‘true’ strip-strip alignment, a possibility is to do a magnet on strip-strip alignment. This is in principle possible, if very straight tracks are chosen. How well this works in reality remains to be seen. One possible approach is a much more complex alignment, which has rotation of strips (modules) in addition to transverse offsets of strips (modules) as an additional degree of freedom. This is in principle the correct way to do things. However, given the much larger data set that would have to be processed (this is not trivial), this would also require a change to the data model used for the alignment. Given the small amount the current misalignment, efforts might be better spent on other reconstruction related tasks.

### **Acknowledgments:**

This alignment could not have been possible without input and help from many MINOS collaborators. In particular the efforts of Brett Viren, Leon Mualem, Robert Hatcher, Nick West, Jim Musser, Jon Urheim, Julia Thompson, Jeff Nelson and Wes Smart.

### **Endnotes:**

1. NuMI-Note-856
2. NuMI-Note-876
3. NuMI-Note-913

## Appendices

### Appendix A: Detailed procedure for alignment

The procedure for alignment will be broken down into the following stages:

1) The first part of alignment is event reconstruction. This was done using R1.7 on the FNAL cluster. The reconstruction was done using the alignTracker.C script (see Appendix D). There was a separate script for SM1 and SM2. The scripts only differed in regard to whether they reconstructed tracks in SM1 or SM2. Both scripts assumed no magnetic field for tracking. The first run was on generation data. Table 3 has a list of all data used in the improved alignment. The UgliDbiScintMdl table was set back to the nominal values for both SM1 and SM2 to try and make the starting point as similar as possible. This generates two files, a candidate and ntuple file.

2) The candidate and ntuple files are used as an input to align.C in order to carry out the actual alignment. The alignment algorithm (described in NuMI-NOTE-876) is carried out on members of CandTrackFitList that pass the data selection cuts. This is done by cutting on the ntuple file and only selecting on candidate events from the corresponding run and snarl that pass the requirement. The align.C is run on all files in a given iteration. The database must be rolled back to the most recent constants. This is important as the alignment algorithm should be run over all the data in a given iteration. The result of align.C is a root file (SM1PassOne.root for example). This root file has all the important information in a given alignment (mean residual and number of tracks per iteration for example) as a function of iteration. Iteration in this context does not mean the same thing as iteration means in the rest of this document. The alignment algorithm allows for iteration. However, at each step of the iteration, the tracks are not reconstructed. It is simply assumed that shifting the tracks a small amount does not change the track. It should be pointed out that although this seems to be a perfectly reasonable statement, it is not necessarily true, as hits near the edge of two strips carries ‘extra information’ compared to one which crosses at the center of a strip. This ‘extra information’ is location of the strip boundary and therefore the strip and because we know where the strip is within a module, the location of the module. But if the assumption that a small shift does not change the track fails, it fails this situation. It was decided to do the ‘iteration by hand’. This means that only iteration zero is used for alignment.

3) The zero iteration is extracted from the root file (SM1PassOne.root for example) by using the GetInfo.C macro (see Appendix D). The extraction is simply a text dump. These constants then are added to the nominal (UgliDbiScintMdl6nnn series) values in the UgliDbiScintMdl table. This table and the modified UgliDbiStrip and modified UgliDbiScintPln tables are in a development release. The run script must be told to first go to this development database and then go to the normal offline database with all the other tables.

4) The next iteration is ran. The only difference change to any script involves the rollback feature. The rollback must be changed so it will read in the new modified UgliDbiScintMdl table

corrected for the previous iteration. Only three iterations were ran in the actual alignment although this could be different in a future alignment.

5) Each iteration works in the exact same way as the zero iteration (as explained in steps 1-4). Run the jobs, find the new corrections, modify the development version of UgliDbiScintMdl. Repeat.

6) After the alignment either stops improving or is so small no improvements are deemed necessary, the validation iteration must be ran. The problem is a set of UgliDbiScintMdl SEQNO from a particular iteration must be chosen to validate. Then the validation data set is ran with the UgliDbiScintMdl set to run on the data with rollback date to run on the validated data,

7) After the validation run is complete, the validation either works (the alignment is consistent with the improvements to the mean residual distribution expected from generation stage) or it does not. If it works the new database tables are made ready for distribution to the collaboration. If the validation fails, the process could very well have to start from the beginning with some improvement/modification.

## Appendix B: Misalignment effect on strip uncertainty

The uncertainty in the position of the strip of width  $s$  that has a misalignment error  $m$  and physical strip width of  $w$  can be represented as:

$$\sigma_s^2 = \sigma_w^2 + \sigma_m^2 + \dots$$

where given the fact the strip are uniform width  $w$

$$\sigma_w = \frac{w}{\sqrt{12}}$$

Ignoring the additional terms (which are not zero) and using the binomial theorem you can write:

$$\sigma_s \approx \sqrt{\sigma_w^2 + \sigma_m^2} \approx \sigma_w \left[ 1 + \frac{1}{2} \left( \frac{\sigma_m}{\sigma_w} \right)^2 \right]$$

Given the fact that the strip width is about 40 mm, this gives a  $\sigma_w$  mm. Also given the fact the misalignments are 2 mm or less the binomial approximation is valid. For SM2 this suggests that a  $\sim 750$  m misalignment contributes only about 0.2 % to the total strip uncertainty, while for SM1 the contribution is slightly higher. Thus reducing the misalignments to for example 500

m, does not effectively reduce the total strip uncertainty which is the important physical quantity.

### **Appendix C: Estimate of number of events needed to carry out strip-strip alignment**

An estimate of the number of events (and live time) needed to do a strip-strip alignment are presented. First the back of the envelope calculation:

The MINOS FD has 484 plane with 192 strips per plane or 92,928 strips. The rate of cosmic ray muons is about 0.5 Hz. Suppose that every muon is straight (this is in a magnetic field on data) and that all triggered muons are used for alignment with conditions similar to those used for the actual alignment (20+ planes for simplicity). For purposes of this calculation let the average plane length be 30 planes and assume that every plane has only one hit. Furthermore assume that the detector has 100 % on time. The final (and not that realistic) assumption is that the acceptance is such that every strip is hit at the same rate. With these numbers you expect every strip is hit about 13.9 times a day. This number seems very positive. The alignment resolution is  $\sim (11.6/\sqrt{N})$  mm where N is the number of tracks per strip. Thus to get a 2 mm alignment resolution takes only  $\sim 2.5$  days of data and to get 800  $\mu$ m takes  $\sim 16$  days. If you want to verify this data (which is probably a good idea), you still need only  $\sim 32$  days of data to equal SM1 alignment over the entire detector.

This number seems like an encouraging number, however, this back of the envelope calculation is an underestimate. The most serious flaw in the calculation above is that every strip is the same. This is not true. The problem is that the acceptance of the detector is different at the edges of the detector. This is an advantage of module-module alignment, in that some of these variations are averaged out (although you can still clearly see the effects in module-module alignment). For example when looking at  $\sim 50$  hours of data (magnet off), strips at the edge of the detector had order of a couple ( $\sim 5$  hits). In order to do a strip-strip alignment to 1 mm would need something like 60 days (1440 live hours) of data without verification. This data would have to be processed with special runs using the initial nominal geometry. This would be a very computationally intensive process. All of this is done assuming the same cuts are applied to magnet off data will get the same results for the magnet on data. With verification this is 4 months of data. The result would be an alignment comparable to the current one. Given these estimates, it seems that this is just not possible.

### **Appendix D: Module by Module alignment with magnet on data**

The idea of using 'magnet on' data to do a module-module alignment is more plausible than a strip-strip alignment, however, it is probably not trivial. If a magnet on module-module alignment was attempted, two issues would have to be addressed. Firstly, how much (if any?) would the current data quality cuts would have to be changed to assure the long and straight tracks which are needed for alignment. Second (depending on the first), how much computing power is needed to make an improvement. This is less than a strip-strip alignment but still might take a good deal effort. However, this is probably worth future study.

### **Appendix E: List of macros used for alignment**

All of the macros here are listed at the end of the document.

*AlignTrackerSM1.C*: This just a slightly modified version of a standard macro to run a reconstruction job.

Run with: loon -bq AlignTrackerSM1.C filename.root

*align.C*: This is an alignment script. Its inputs are a candidate file and ntuple file. It outputs a file with alignment information (“AlignmentOutput.root”).

Run with: loon -bq ‘align.C(“AlignmentOutput.root”)’ cand1.root cand2.root .....

*GetInfo.C*: This is used to extract alignment constants from align.C file

Run with: root GetInfo.C

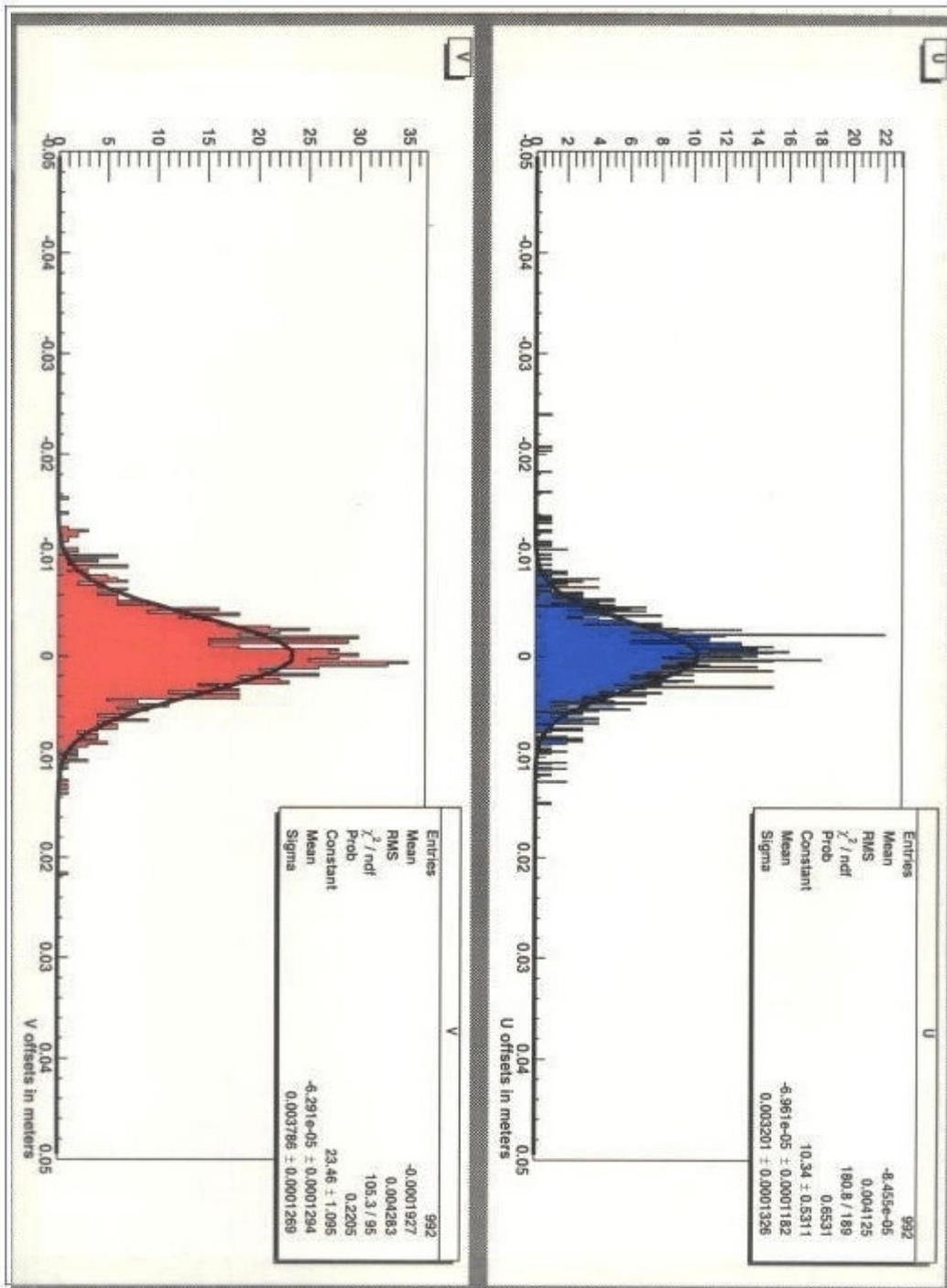


Figure 1: This shows the distribution of module offsets for SM1 for the first alignment. Each histogram point is a single module. This is before any alignment correction has been applied.

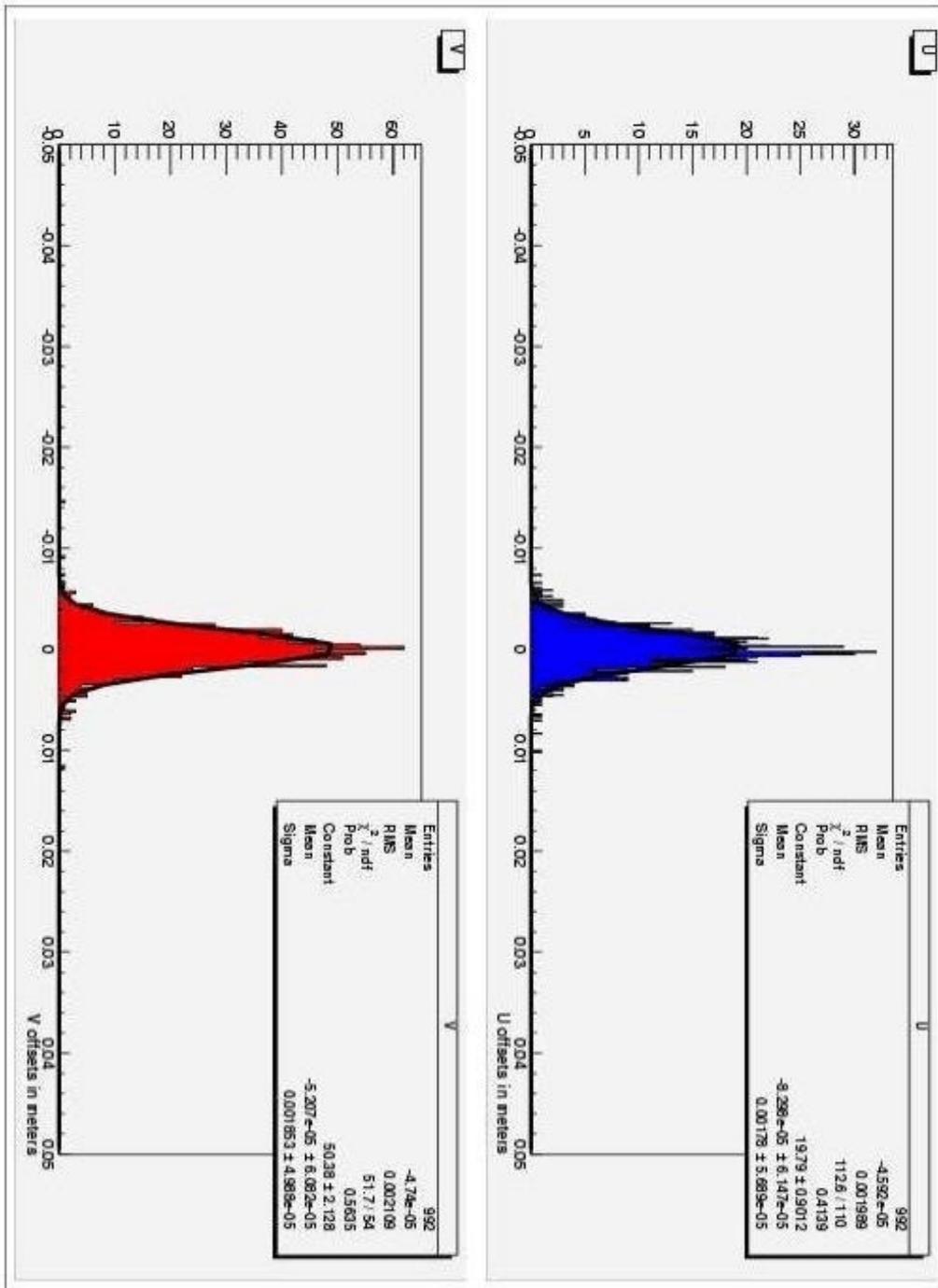


Figure 2: This shows the distribution of module offsets for SM1 in the first alignment. Each histogram point is a single module. Notice how this distribution appears to be much tighter than the same distributions in figure 1. This is evidence that the alignment worked.

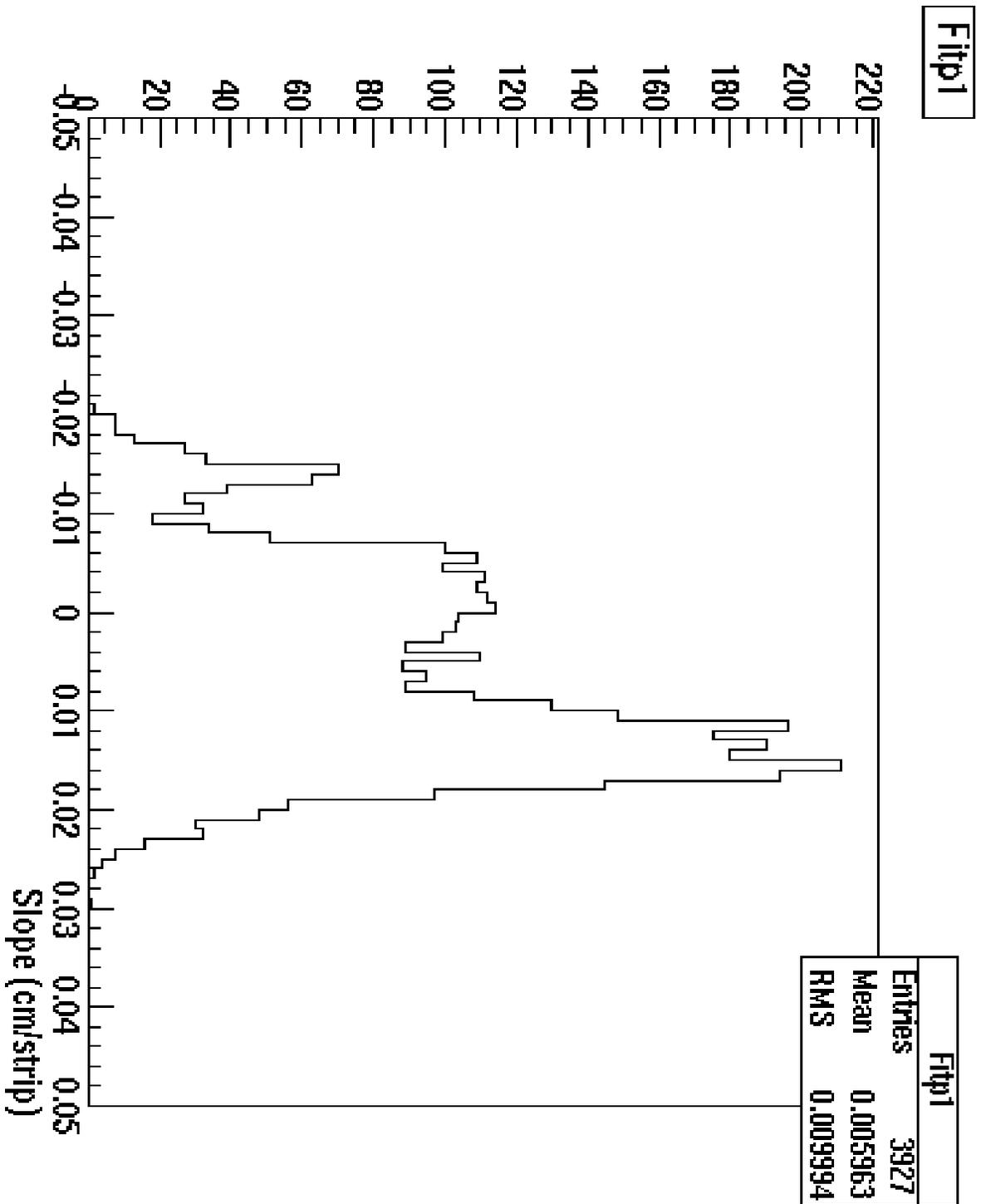


Figure 3: This shows the difference between the average slope of a linear fit to all strip positions in a module and the strips nominal spacing (41.1 mm/strip). This plot includes veto shield modules and is missing a small fraction of the data. This is the result after bad single strips were removed (1mm difference with linear fit for module). This shows that the strips are very close to the nominal value.

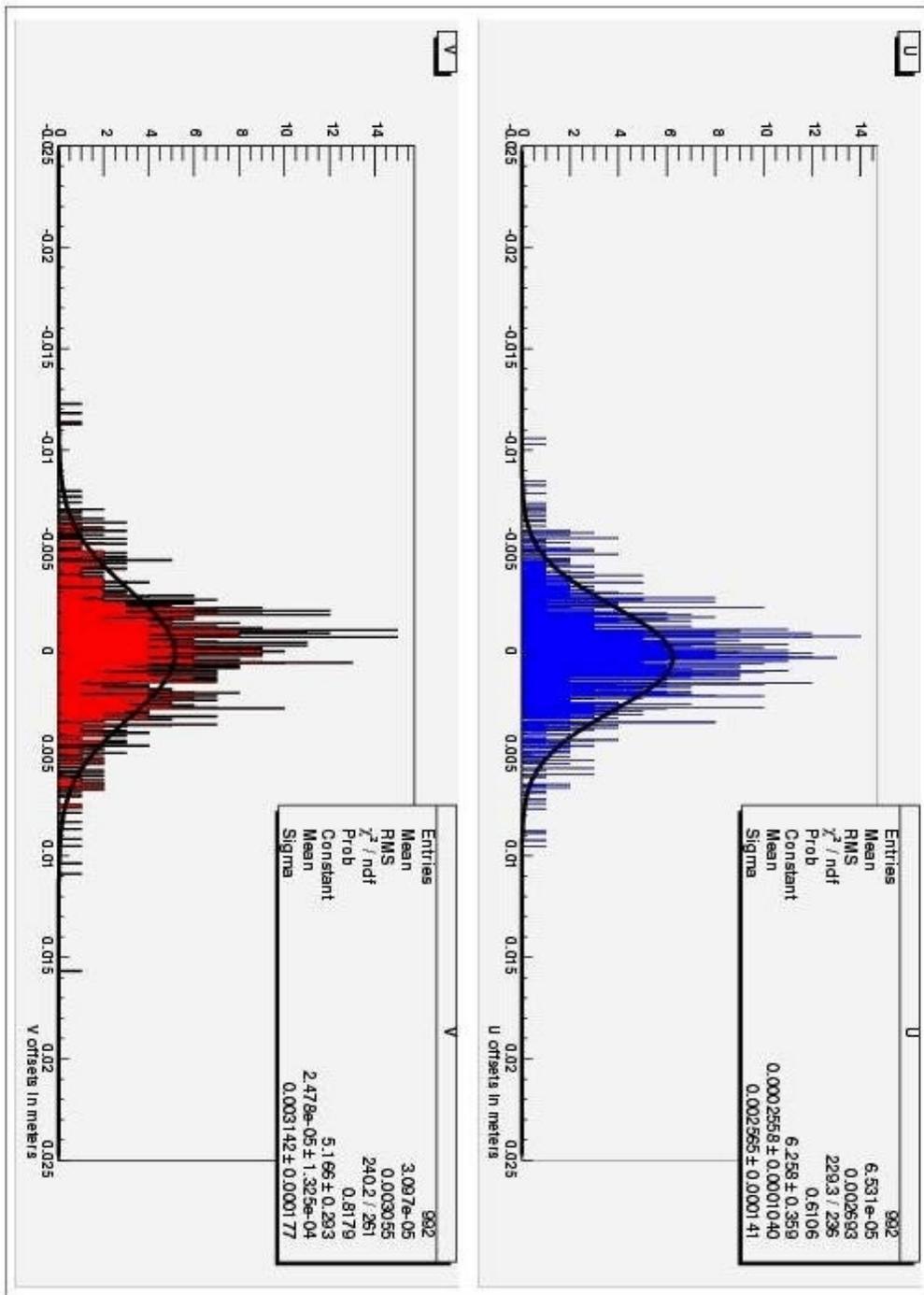


Figure 4: This shows the distribution of module offsets for SM1 after iteration zero. Each histogram point is a single module. This is before any alignment correction has been made.

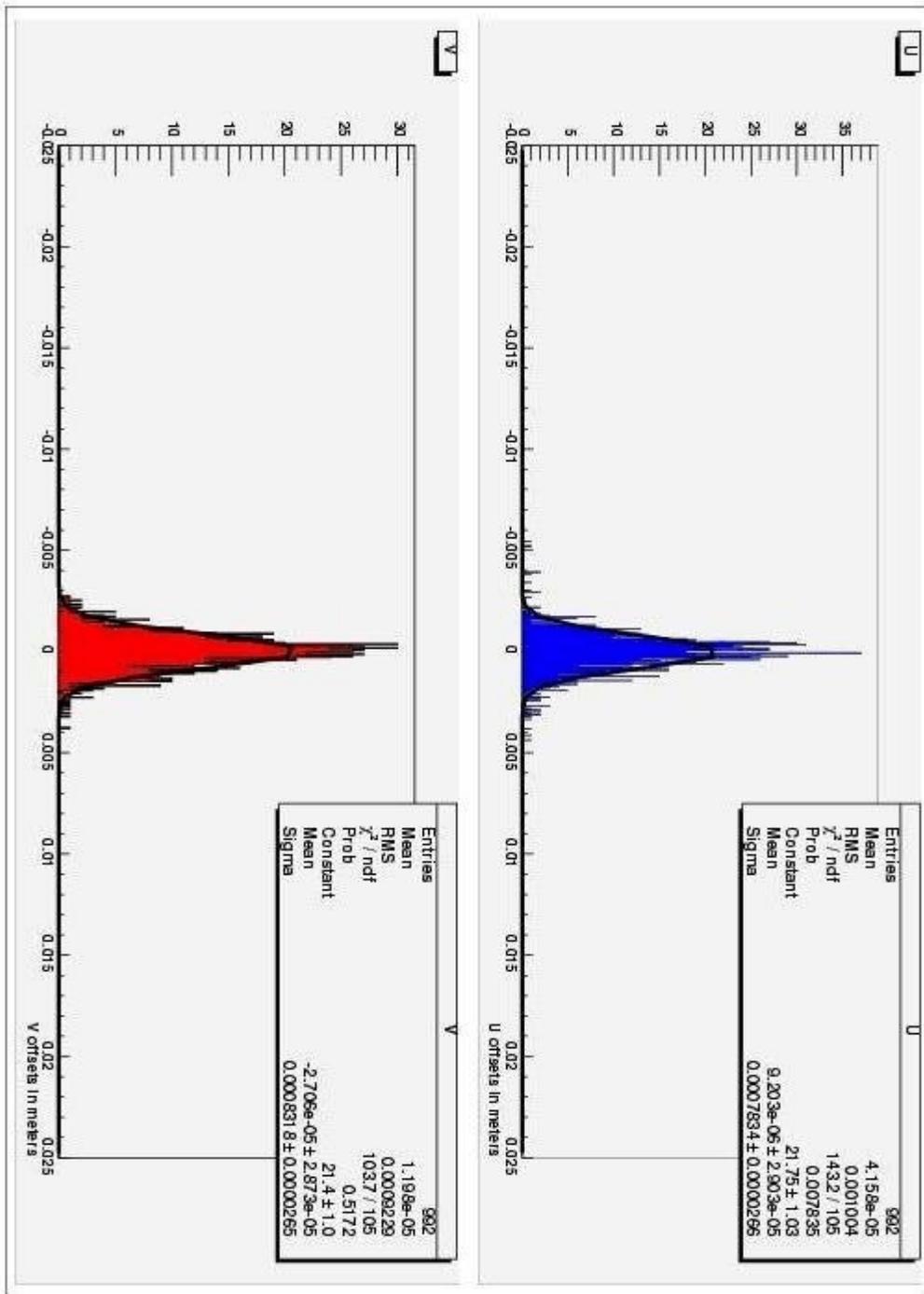


Figure 5: This shows the distribution of module offsets for SM1 after iteration one. Each histogram point is a single module. Notice how this distribution appears to be much tighter than the same distributions in figure 4. This is evidence that the alignment worked.

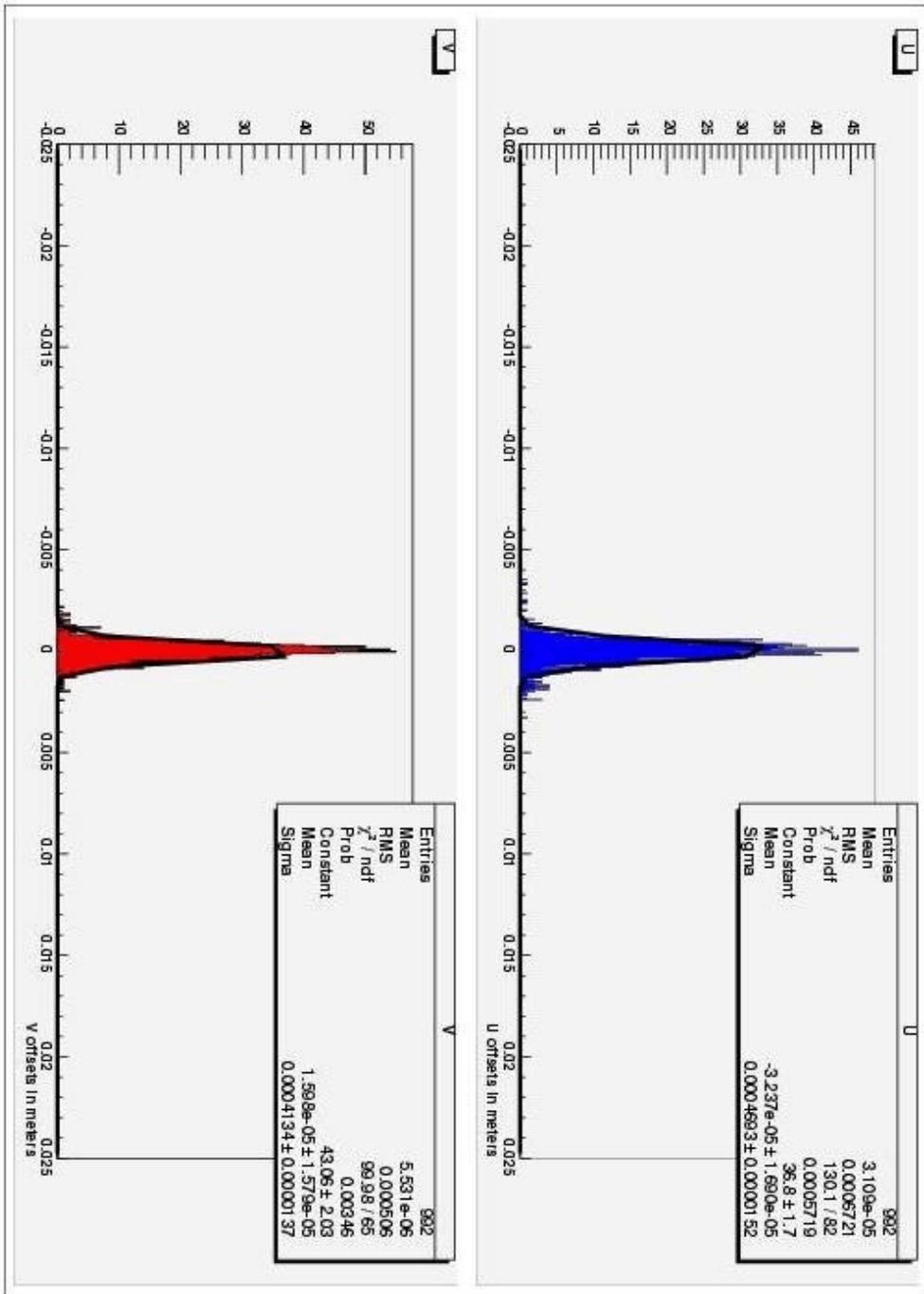


Figure 6: This shows the distribution of module offsets for SM1 after two iterations. Each histogram point is a single module.

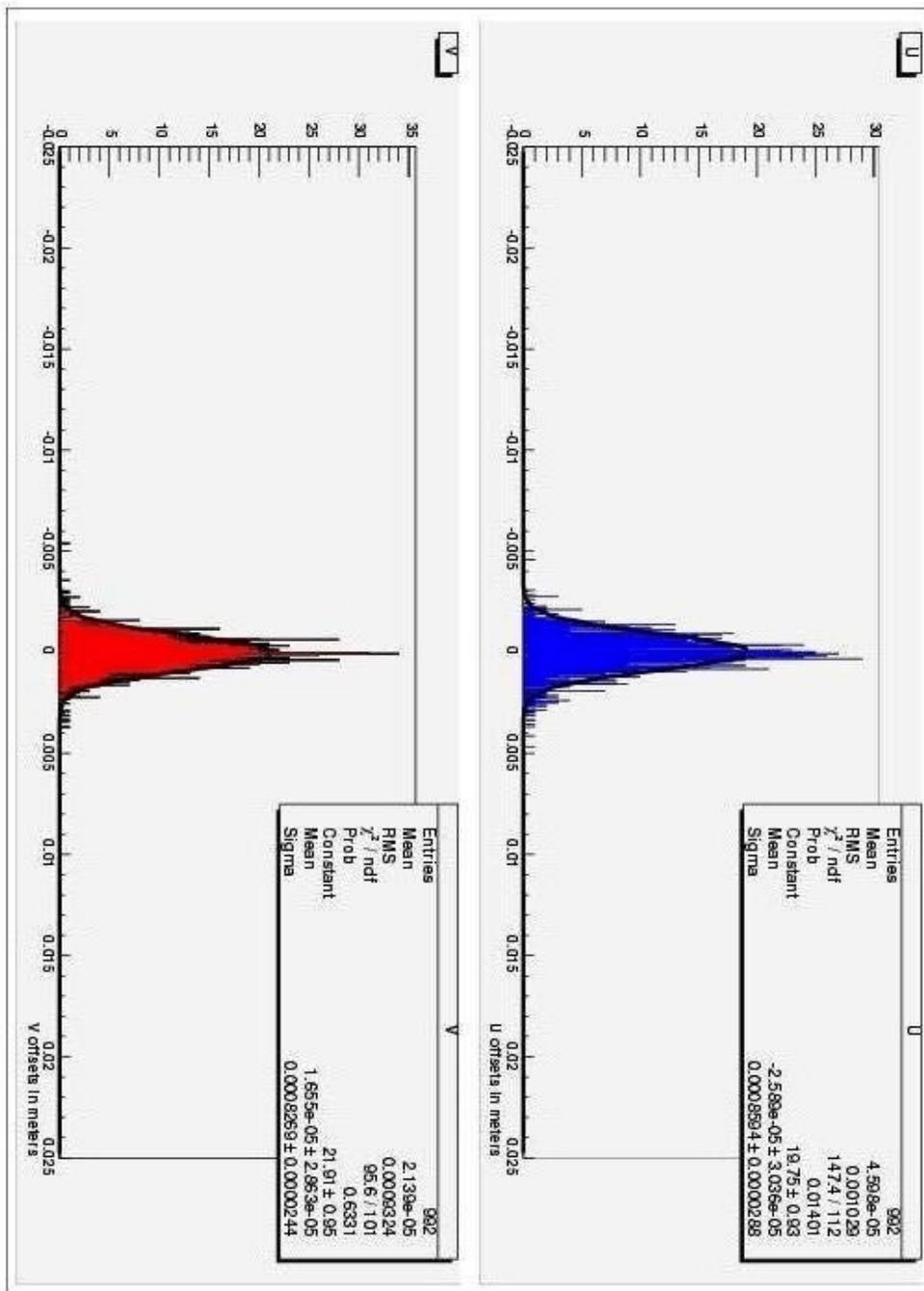


Figure 7: This shows the distribution of module offsets for SM1 after verification. Each histogram point is a single module.

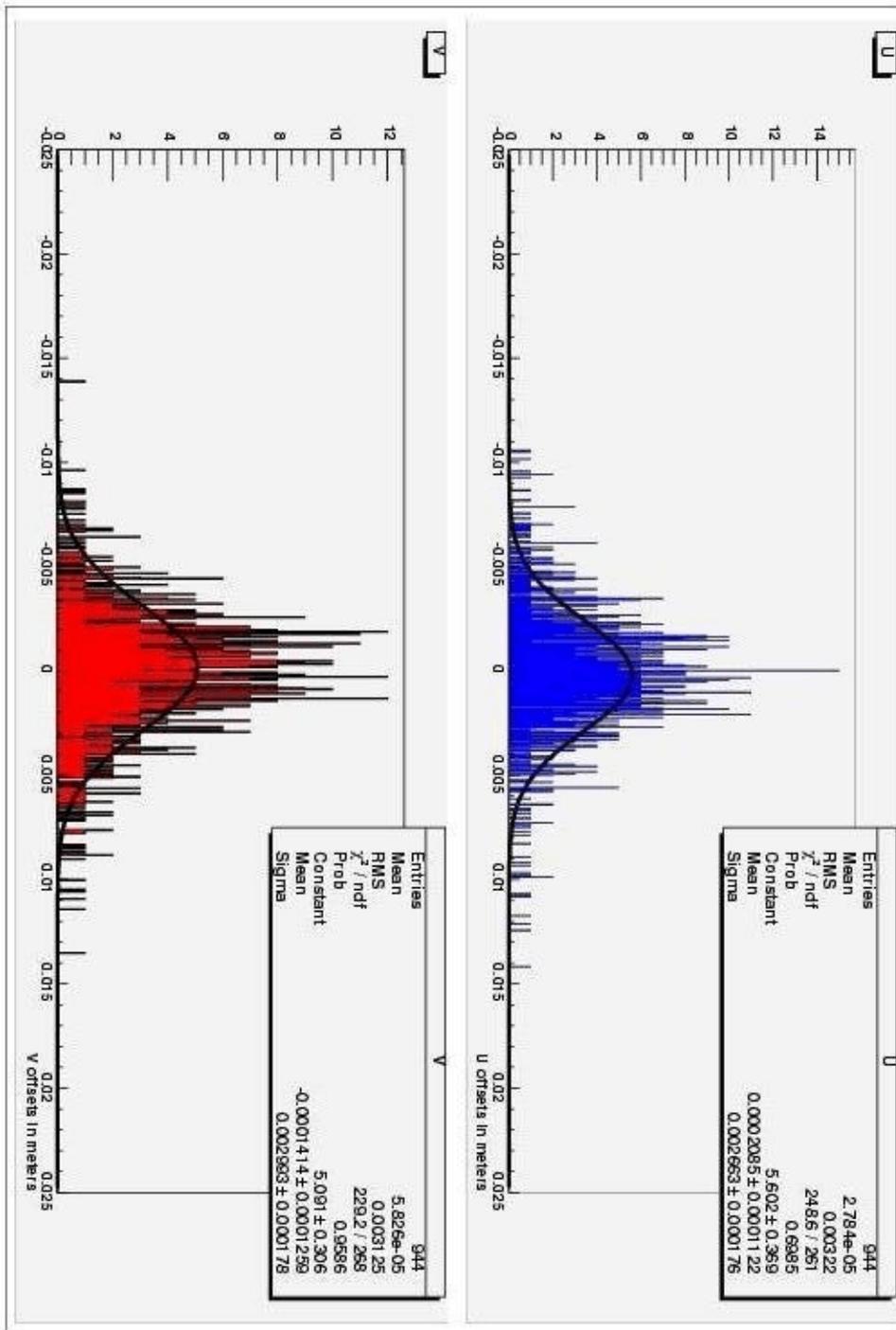


Figure 8: This shows the distribution of module offsets for SM2 after iteration zero. Each histogram point is a single module. This is before any alignment correction.

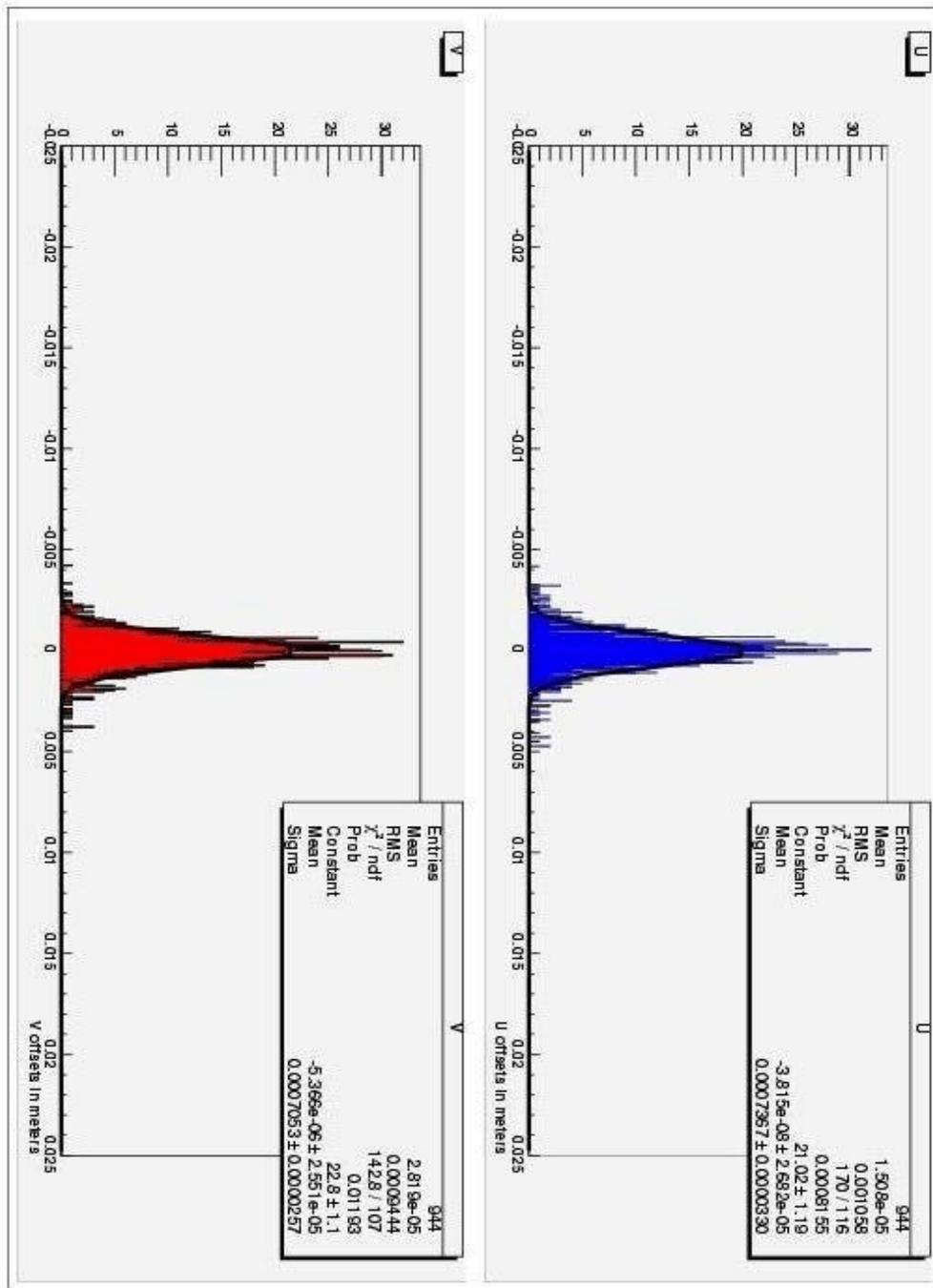


Figure 9: This shows the distribution of module offsets for SM2 after two iterations. Each histogram point is a single module. This is also evidence that the alignment worked.

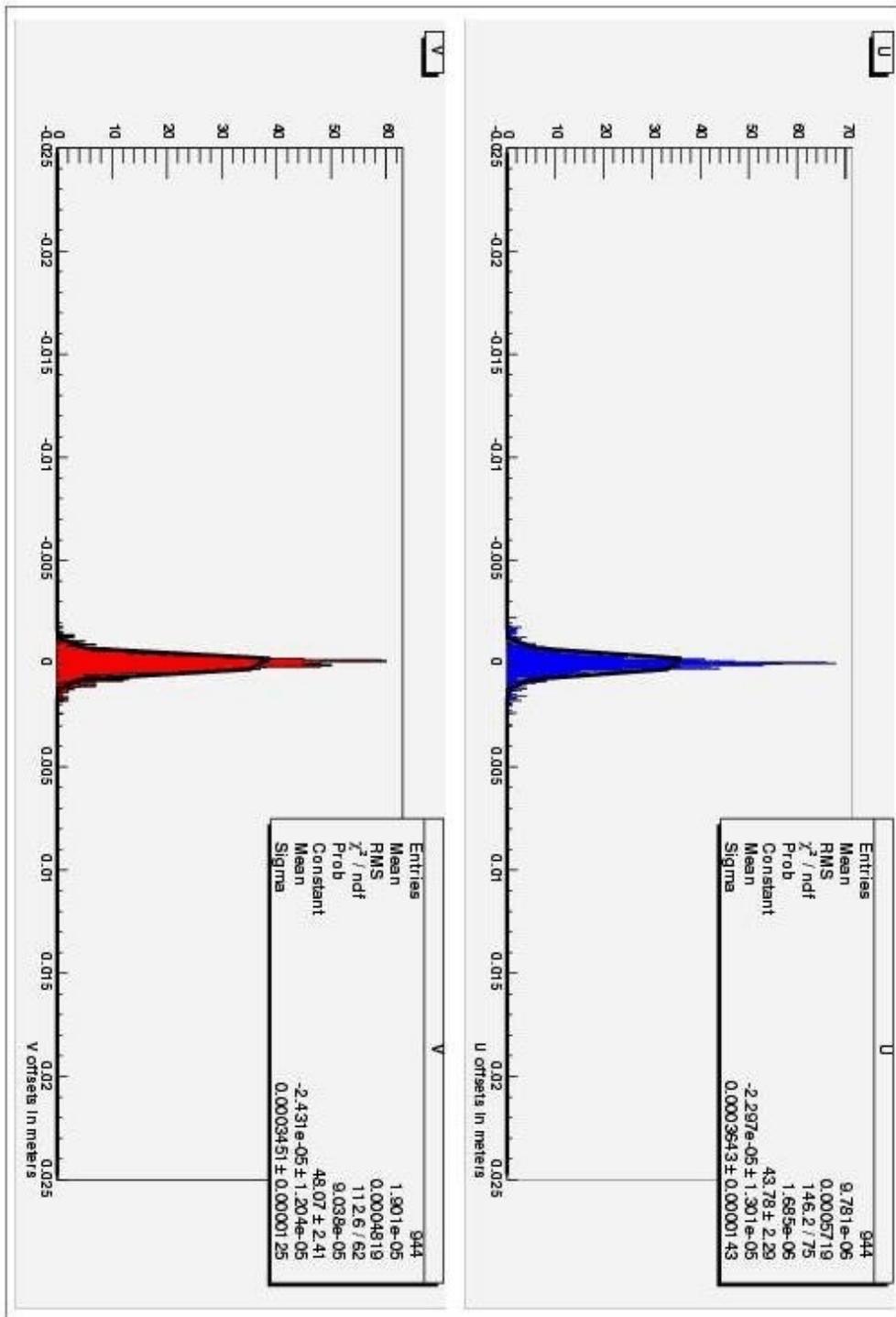


Figure 10: This shows the distribution of module offsets for SM2 after three iterations. Each histogram point is a single module.

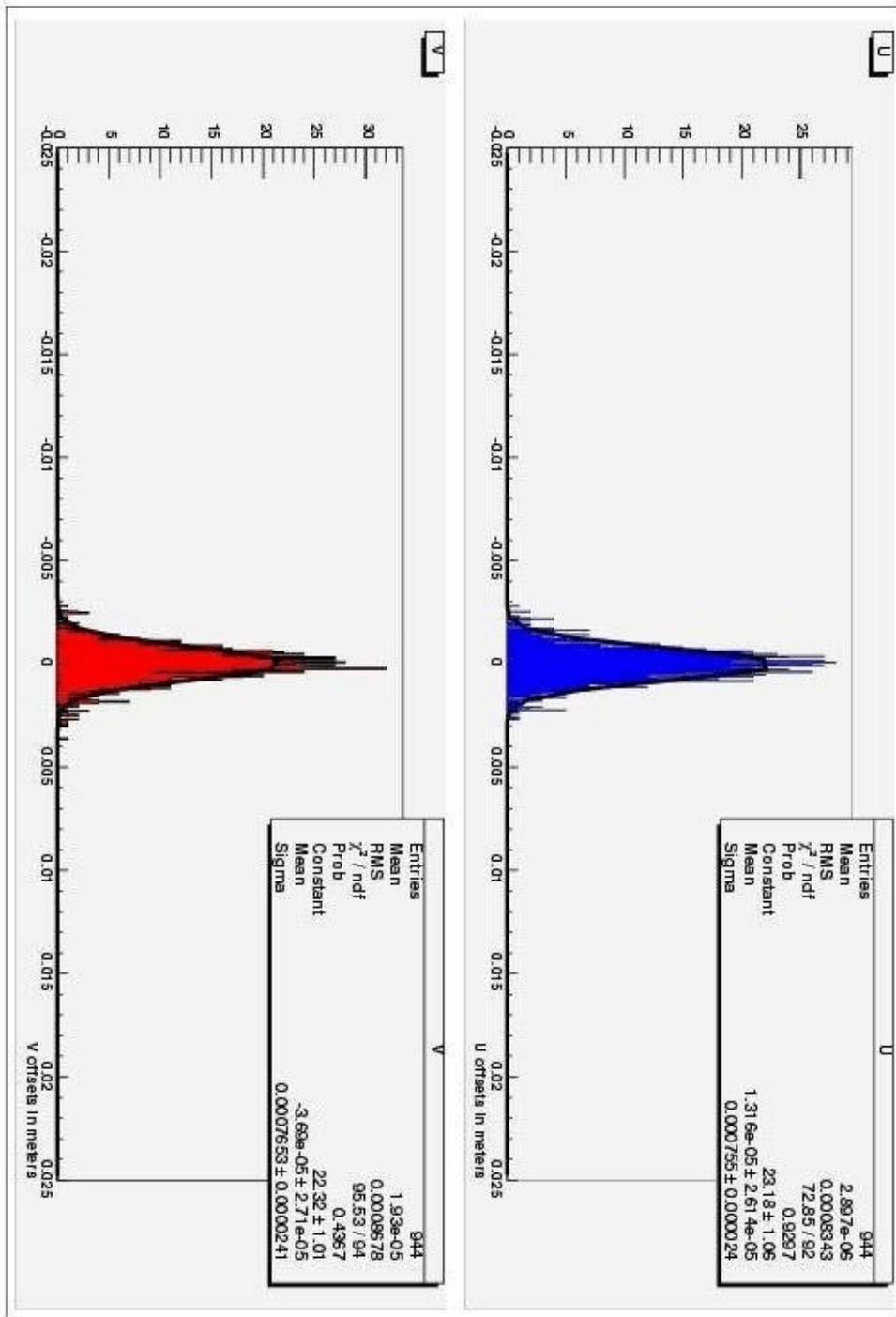


Figure 11: This shows the distribution of module offsets for SM2 after verification. Each histogram point is a single module. This is evidence that the alignment worked.

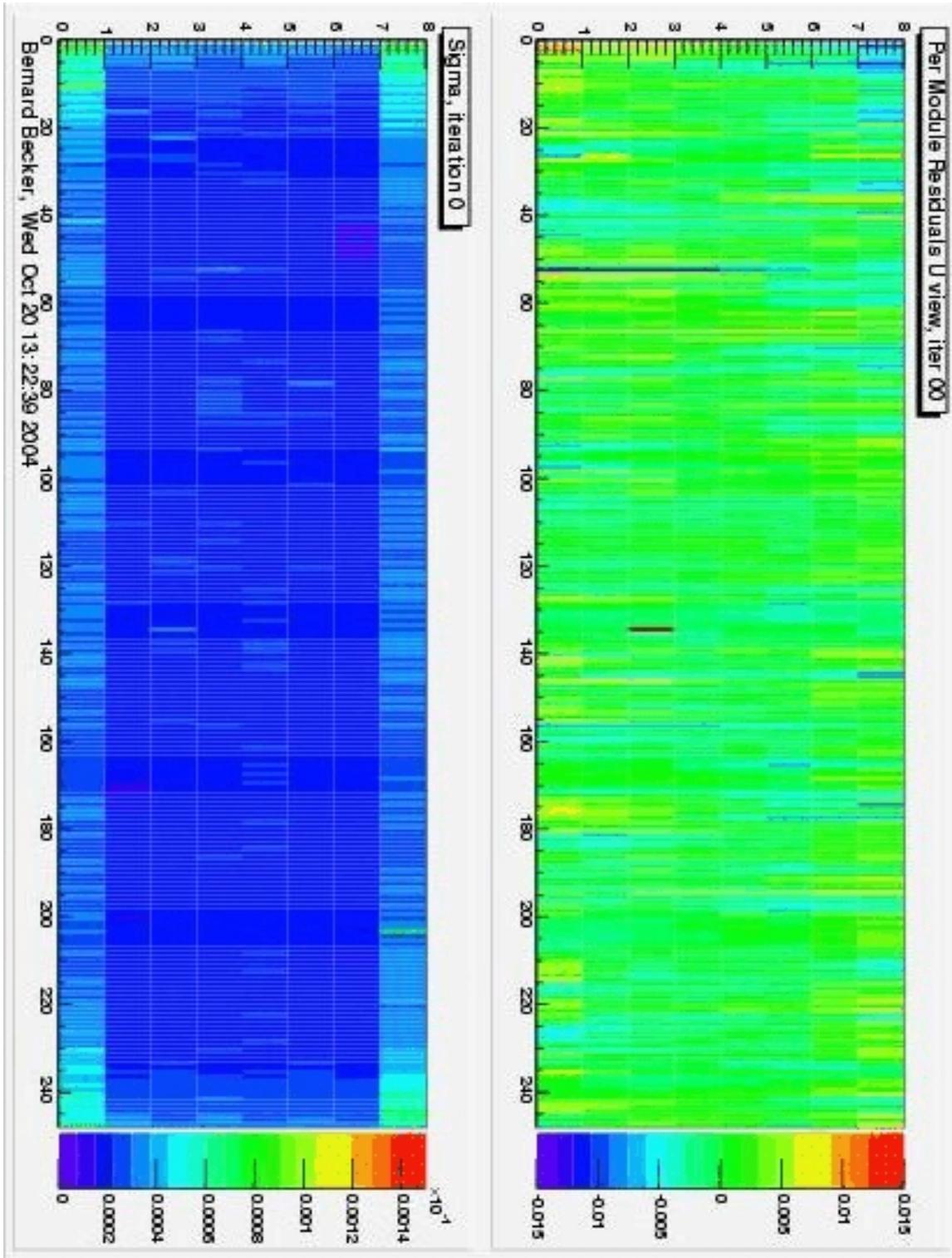


Figure 12: This shows the distribution of module offsets for SM1 after iteration zero. The histograms show misalignment versus plane and module position. This is before any alignment correction has been made. The same information is shown in figure 4.

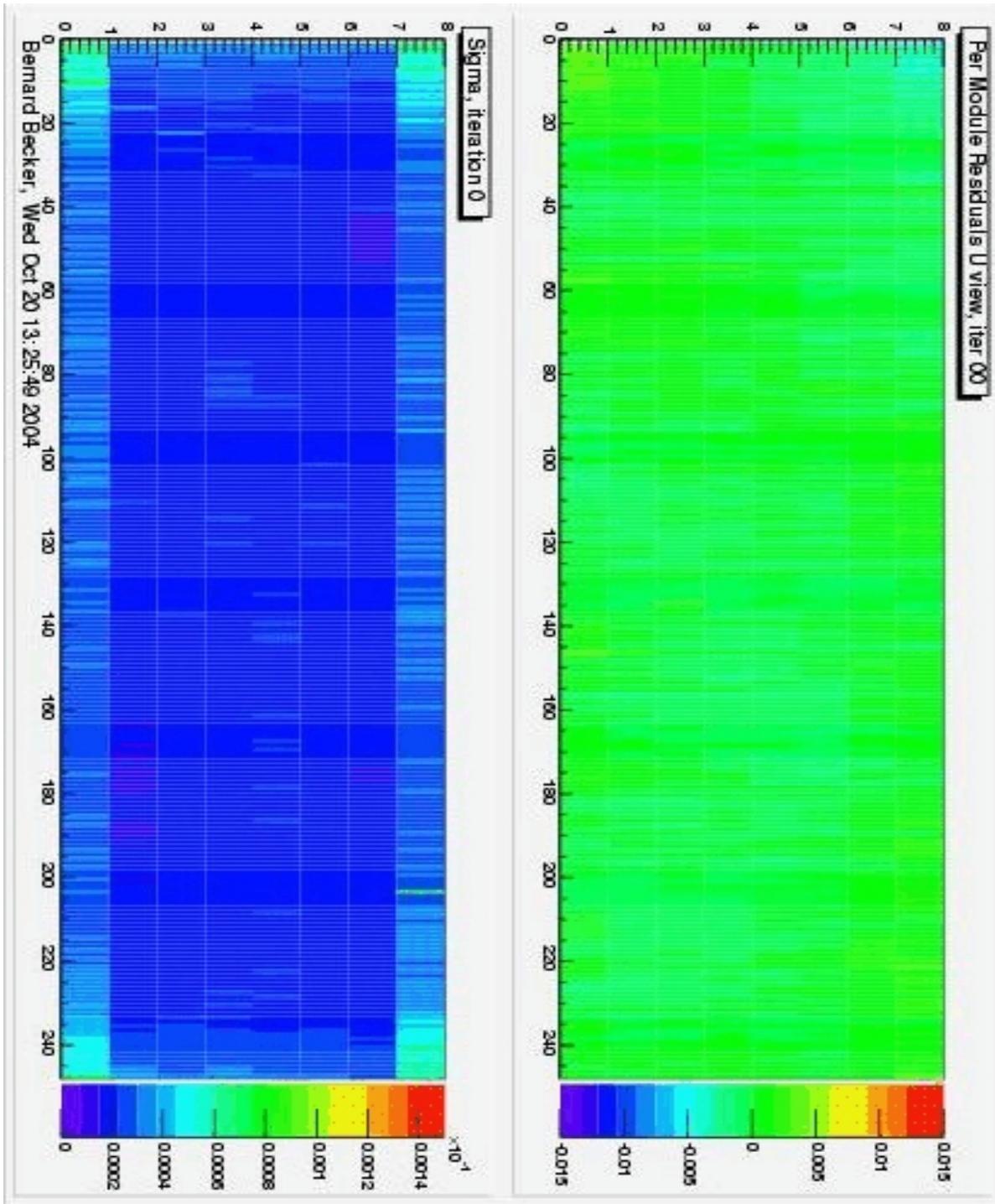


Figure 13: This shows the distribution of module offsets for SM1 after iteration one. The histograms show misalignment versus plane and module position. This is after the alignment correction has been made. The same information is shown in figure 5.

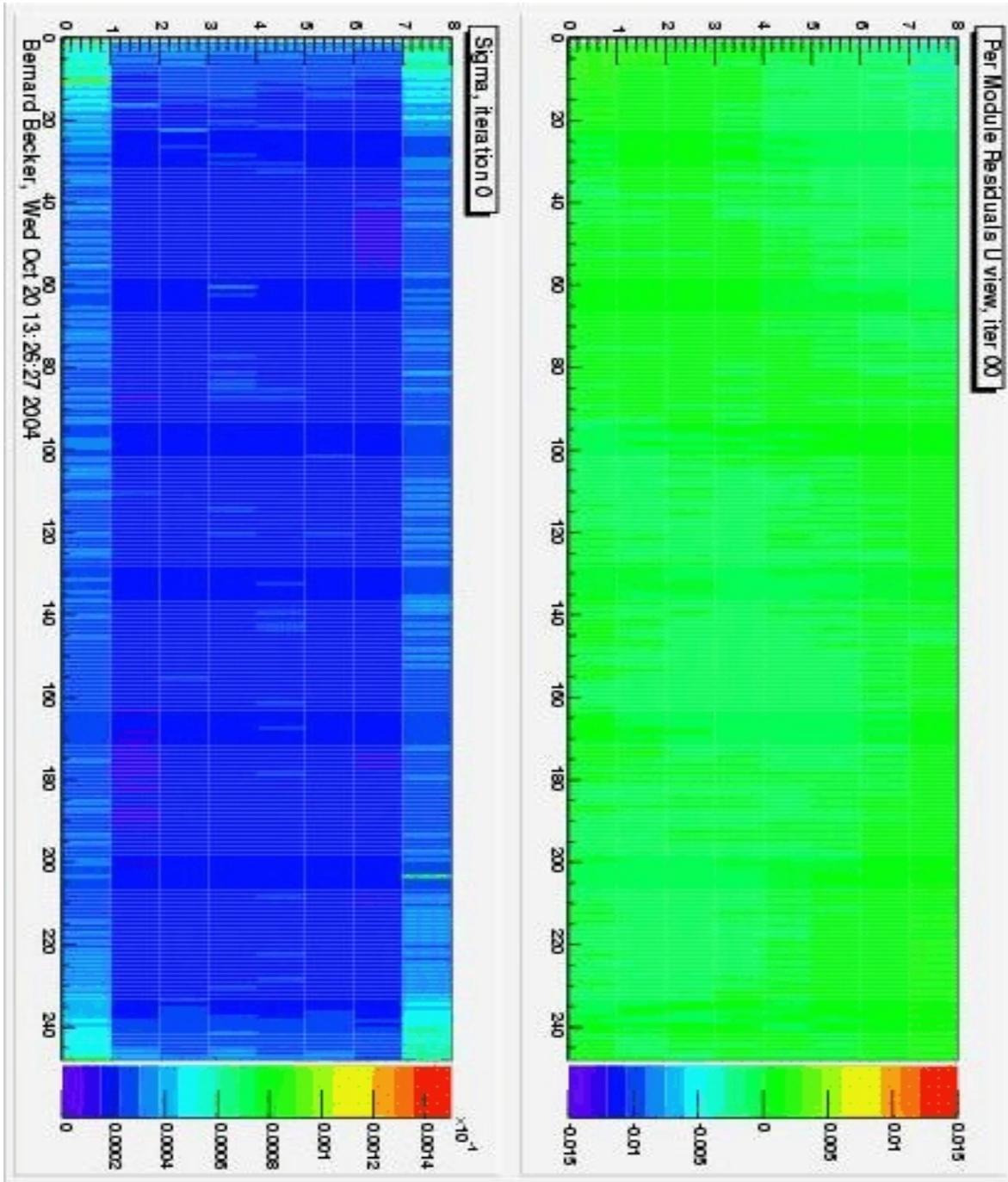


Figure 14: This shows the distribution of module offsets for SM1 after two iterations. The histograms show misalignment versus plane and module position. The results look very similar to the results shown in figure 11. The same information is shown in figure 6.

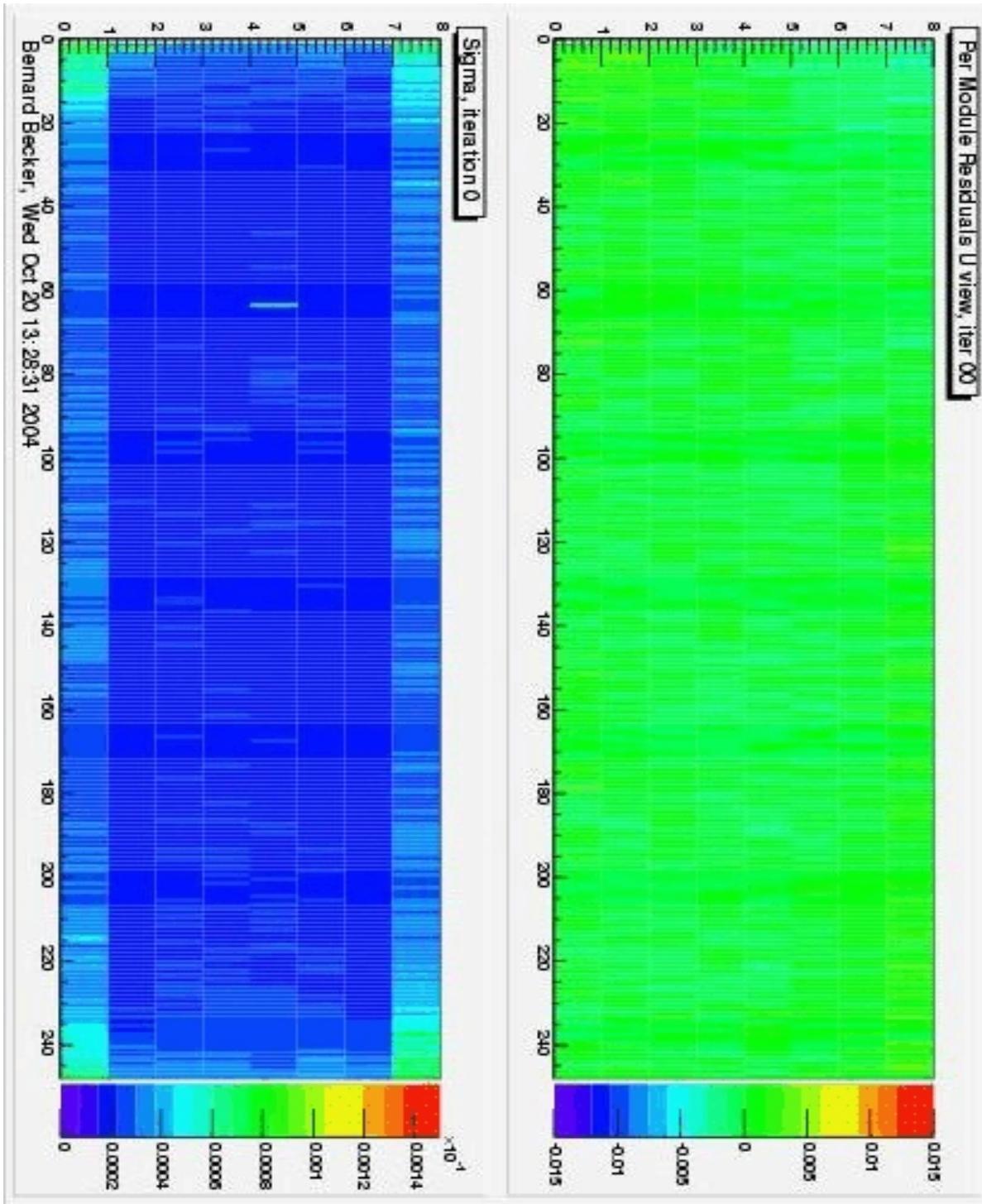


Figure 15: This shows the distribution of module offsets for SM1 after verification. The histograms show misalignment versus plane and module position. This is before any alignment correction has been made. The same information is shown in figure 7.

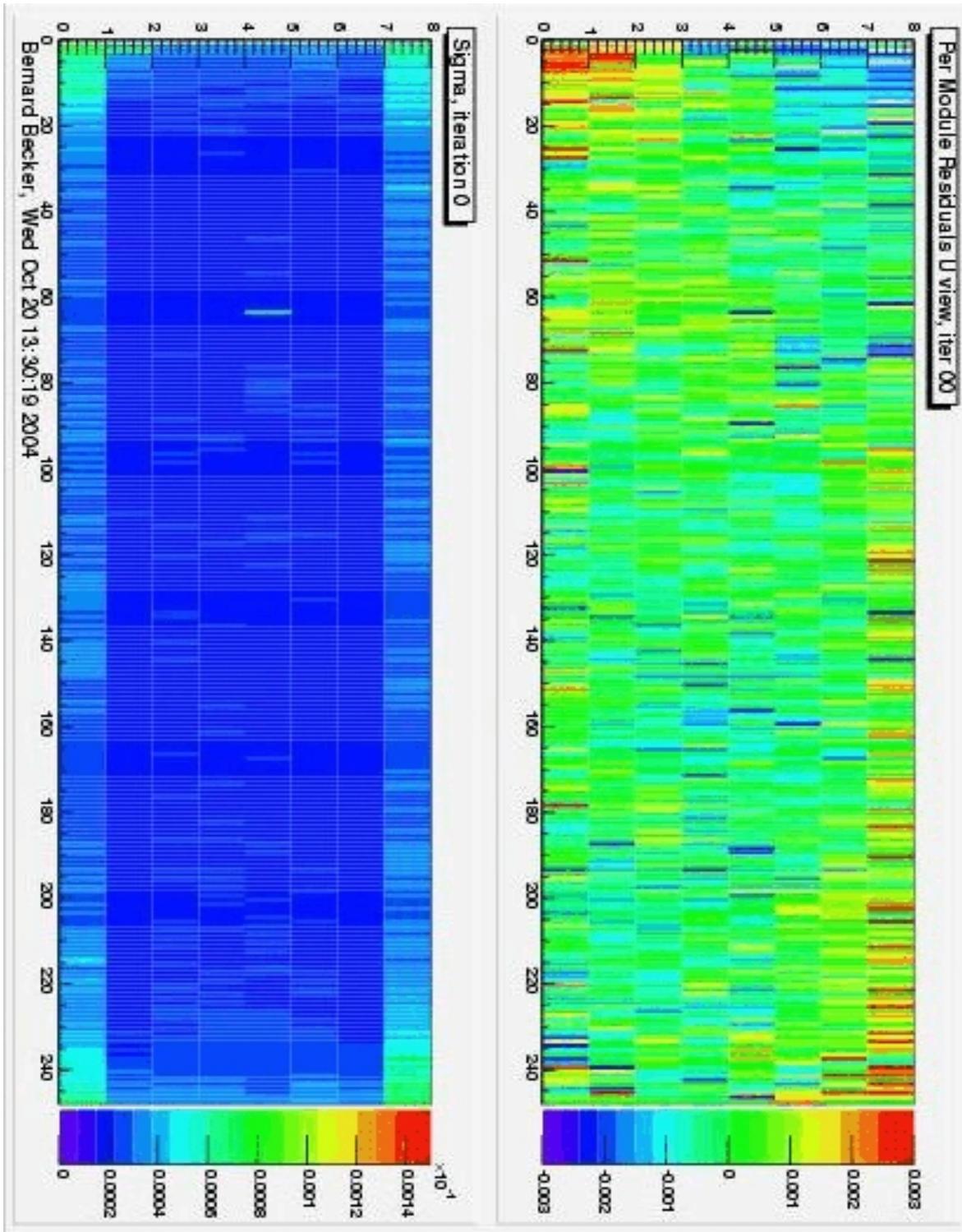


Figure 16: This shows the distribution of module offsets for SM1 after verification shown at higher resolution. The histograms show misalignment versus plane and module position. This is after the alignment correction has been made. The same information is shown in figure 8.

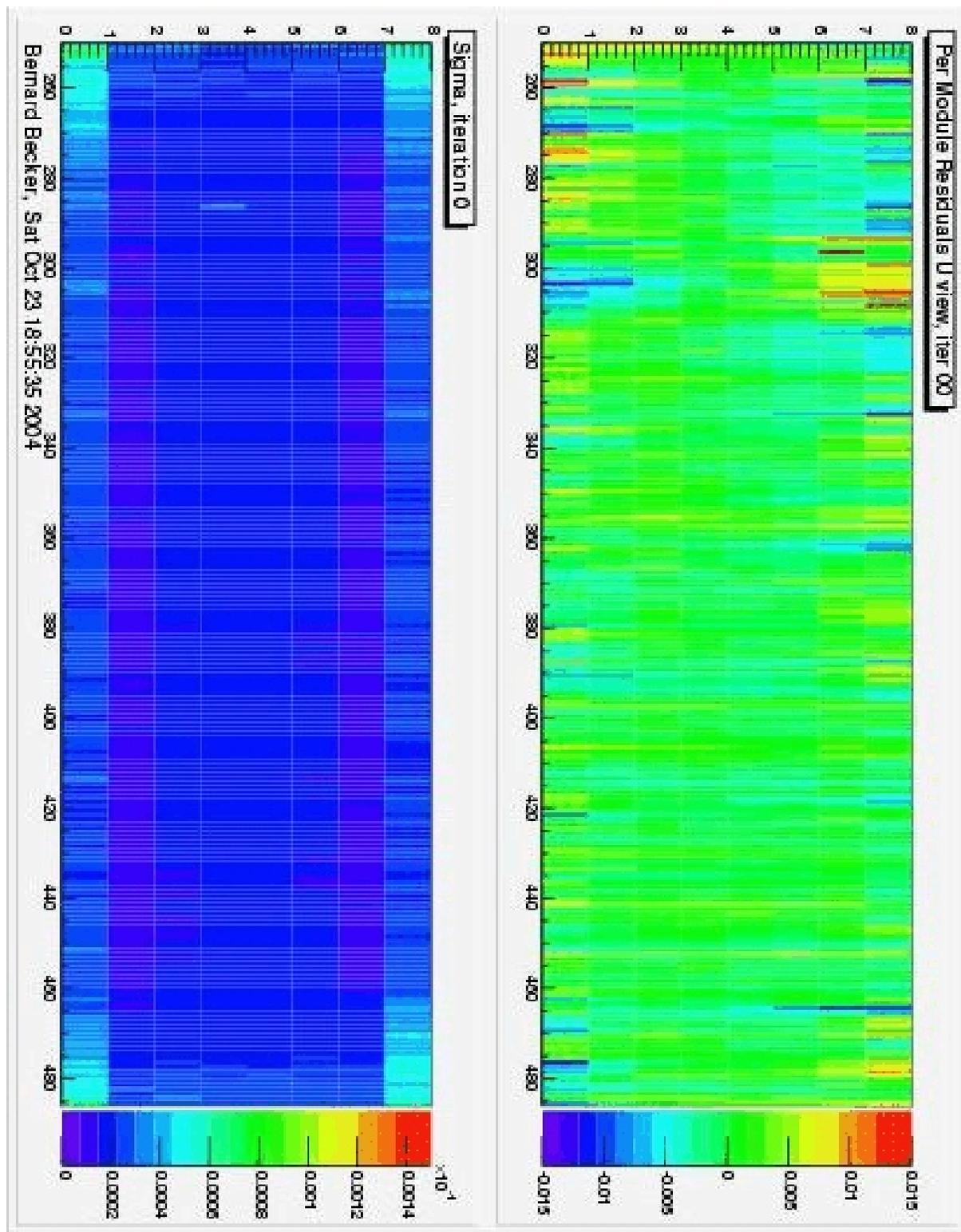


Figure 17: This shows the distribution of module offsets for SM2 after iteration zero. The histograms show misalignment versus plane and module position. The results look very similar to the results shown in figure 14. The same information is shown in figure 10.

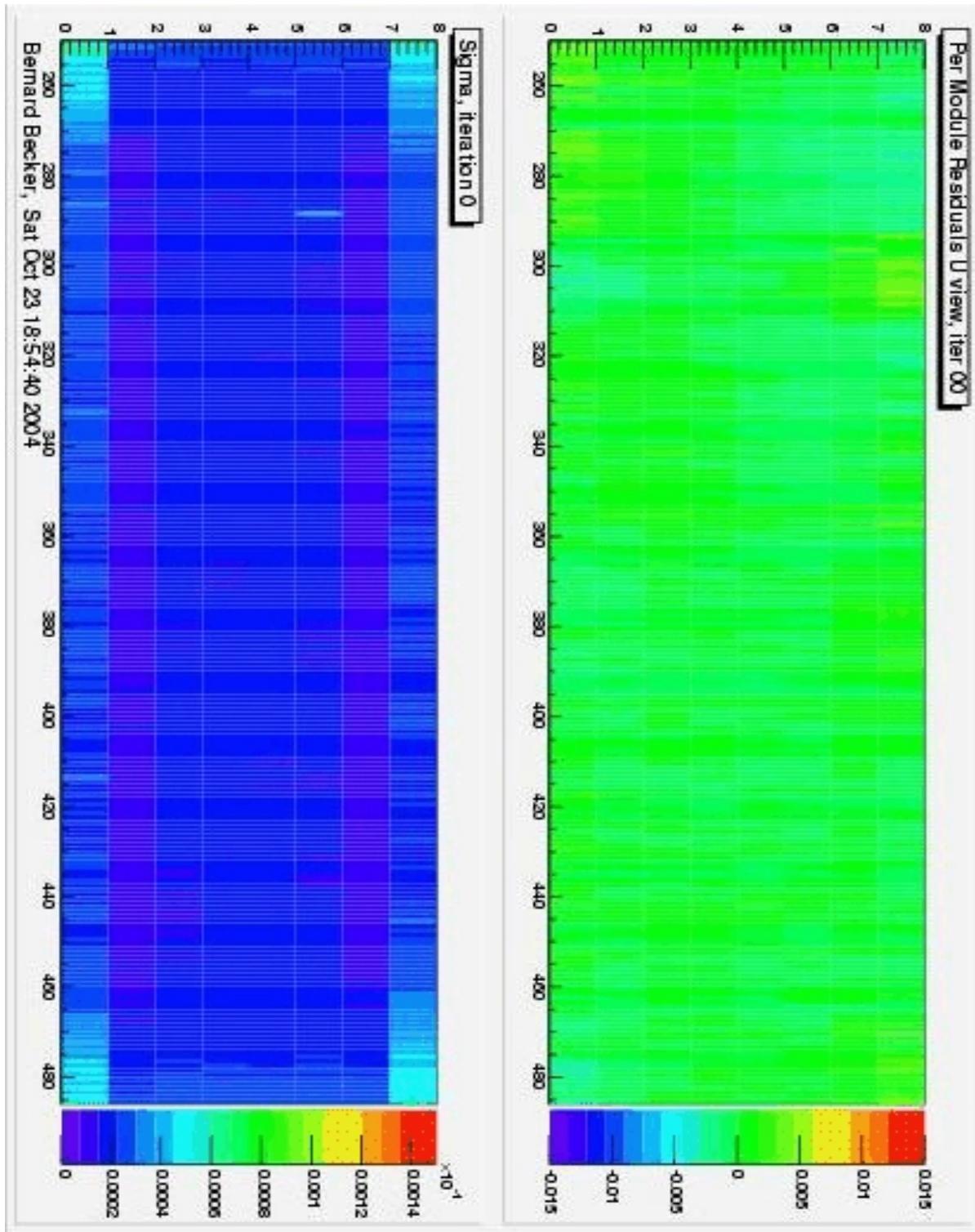


Figure 18: This shows the distribution of module offsets for SM2 after iteration one. The histograms show misalignment versus plane and module position. The results look very similar to the results shown in figure 14. The same information is shown in figure 10.

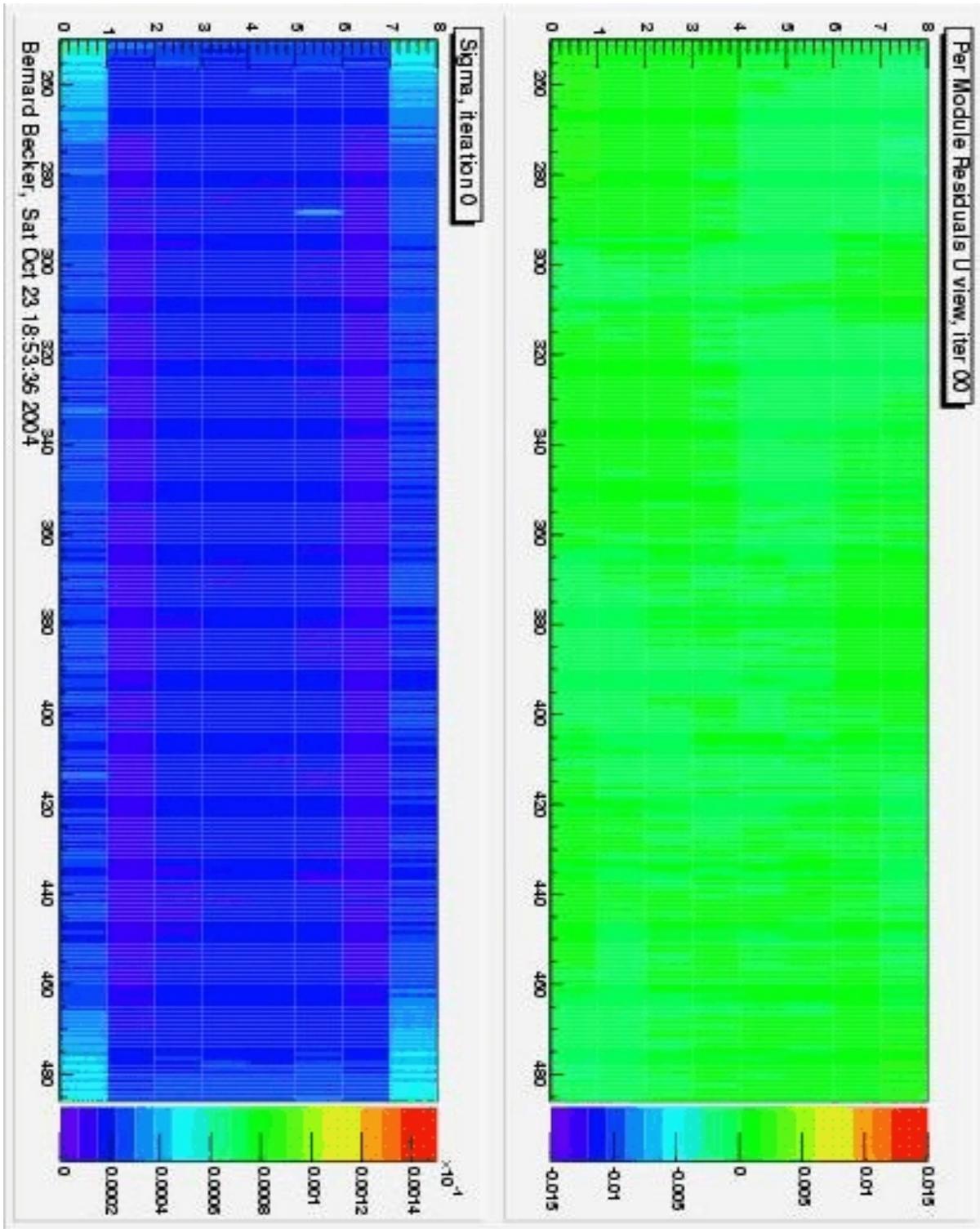


Figure 19: This shows the distribution of module offsets for SM2 after iteration two. The histograms show misalignment versus plane and module position. The results look very similar to the results shown in figure 14. The same information is shown in figure 10.

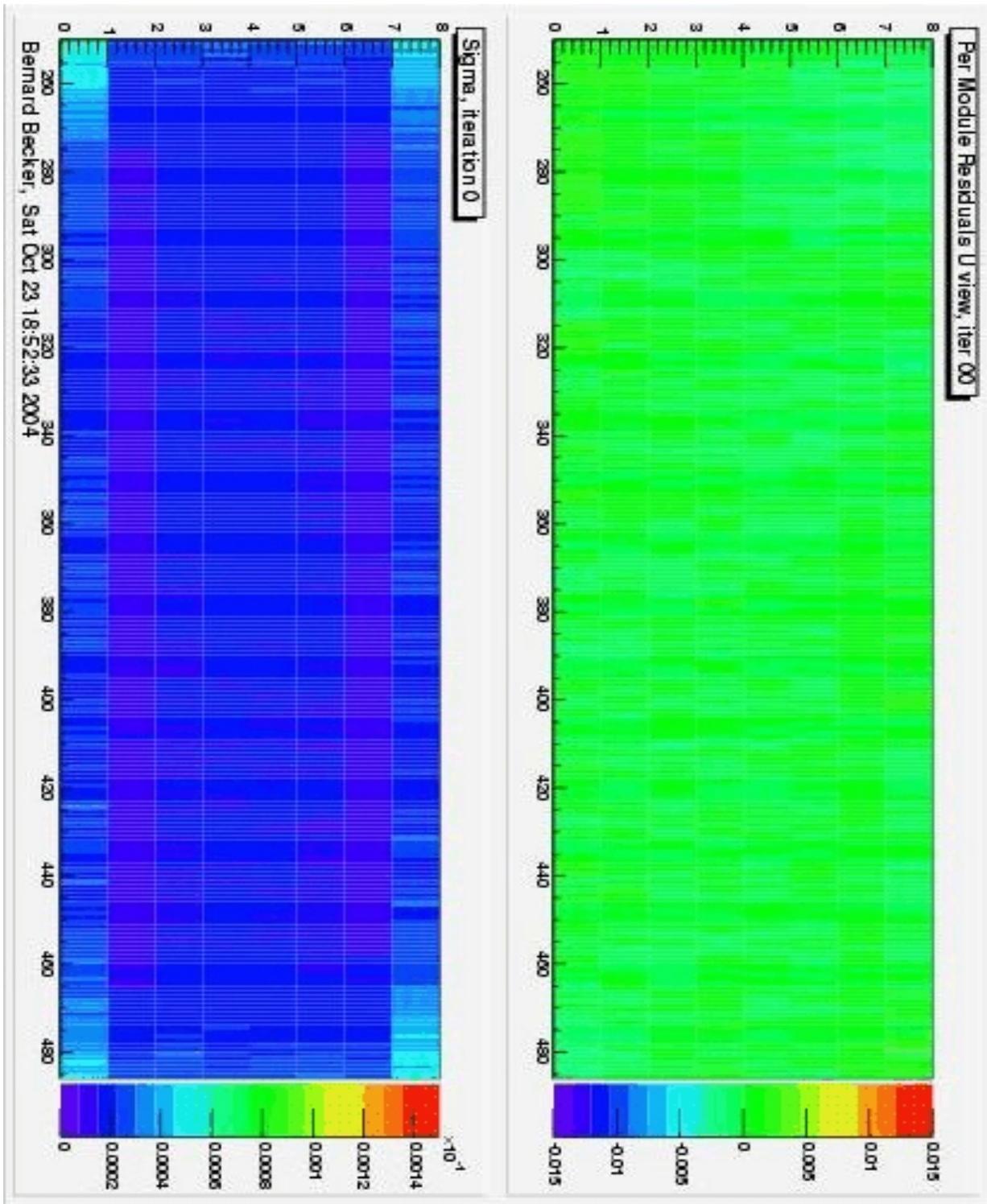


Figure 20: This shows the distribution of module offsets for SM2 after verification. The histograms show misalignment versus plane and module position. The results look very similar to the results shown in figure 14. The same information is shown in figure 10.

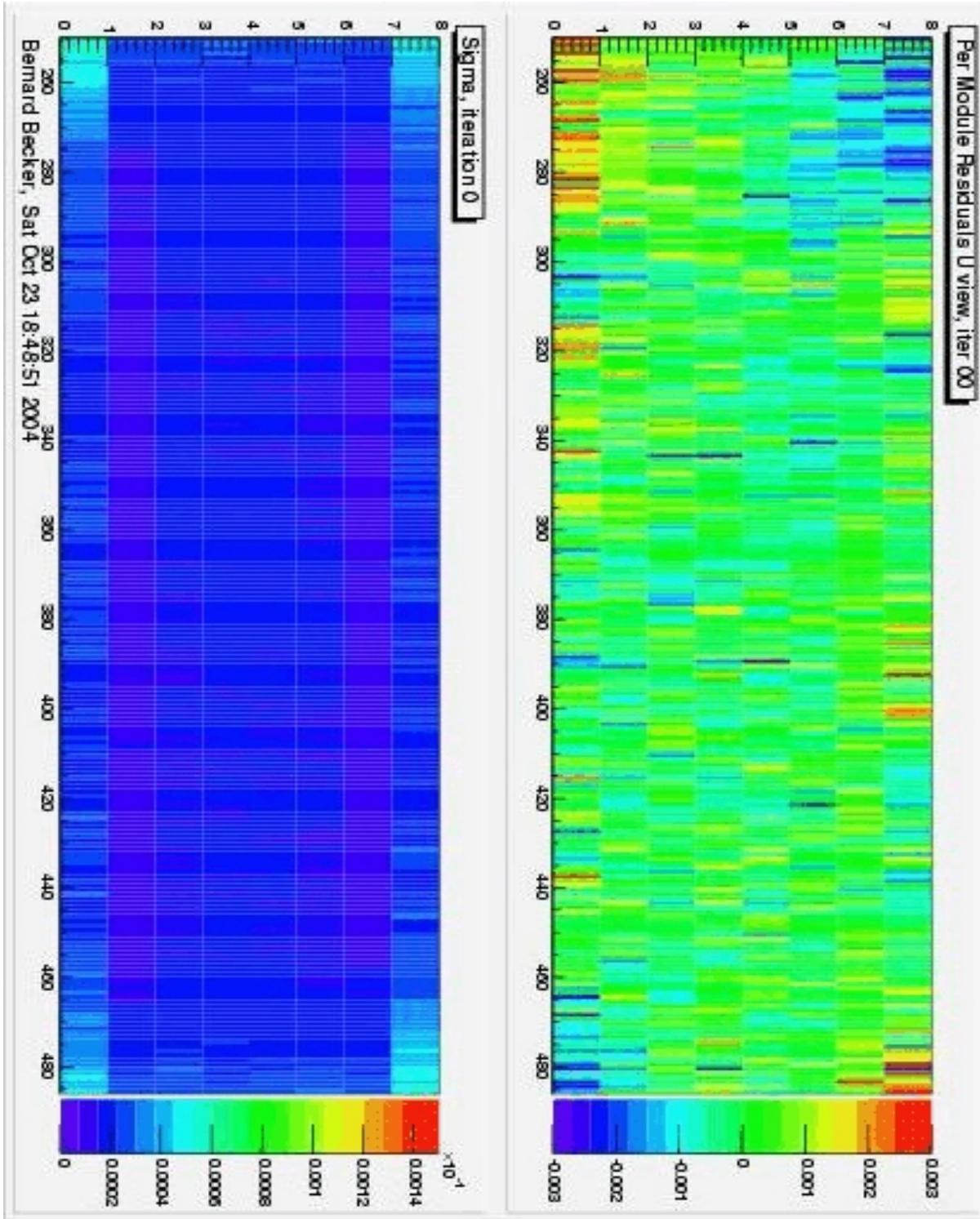


Figure 21: This shows the distribution of module offsets for SM2 after verification shown at higher resolution. The histograms show misalignment versus plane and module position. The results look very similar to the results shown in figure 14. The same information is shown in figure 10.

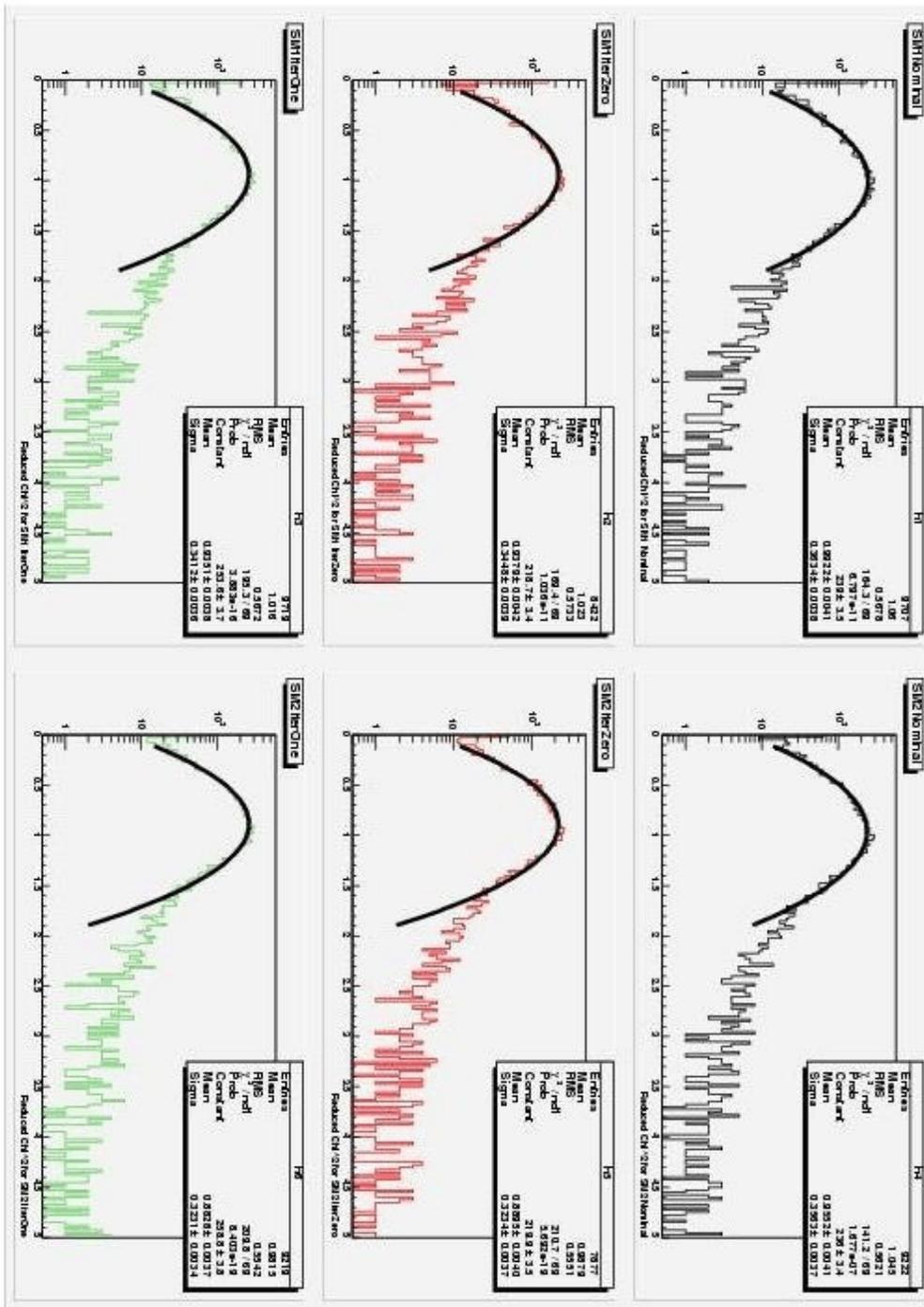


Figure 22: This shows the reduced  $\chi^2$  for all tracks. The mean position of the peak is improved after both the first and second iteration in both SM1 and SM2.

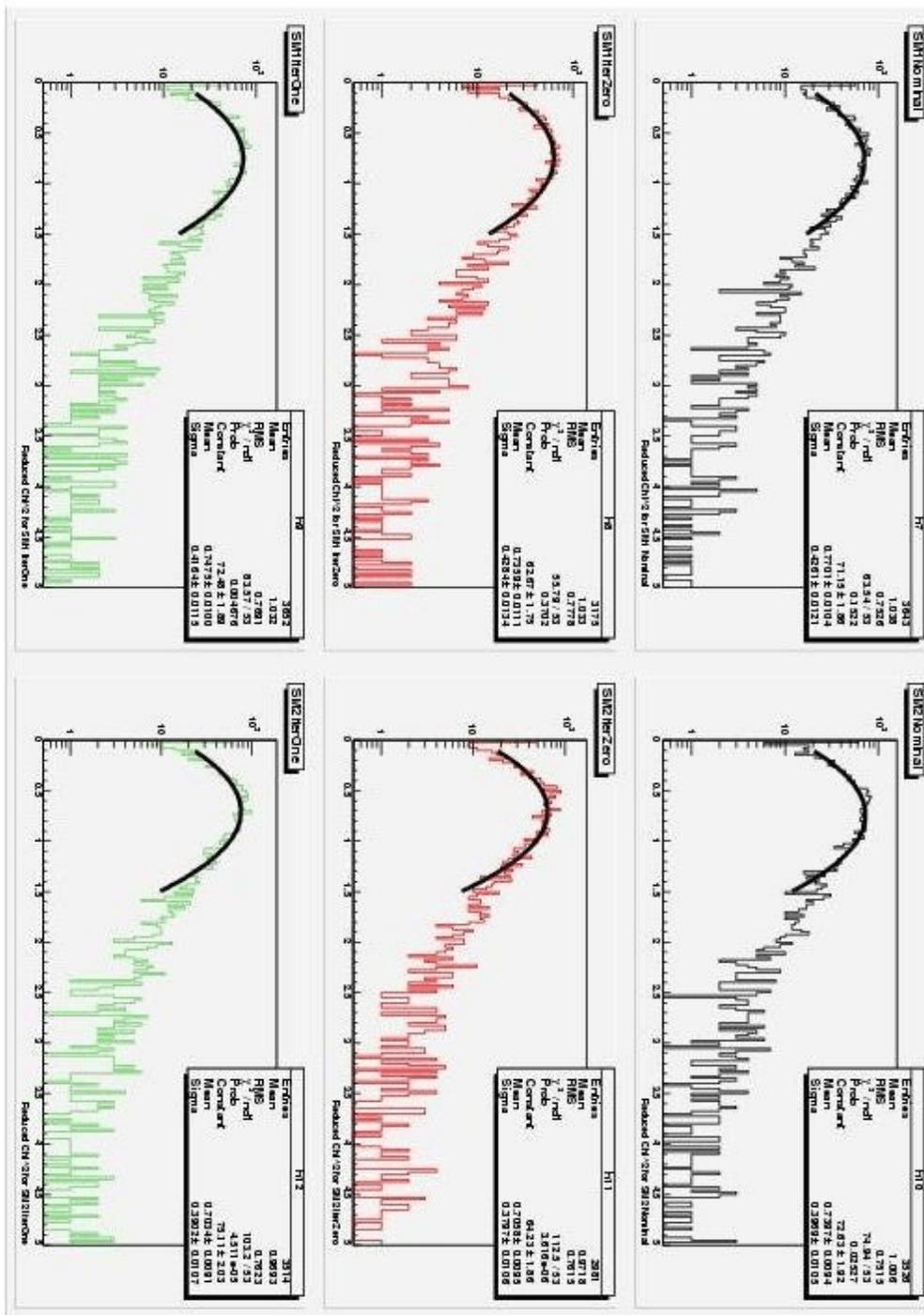


Figure 23: This shows the reduced  $\chi^2$  for short tracks ( $\leq 20$  tracklike planes). The mean position of the peak is improved after both the first and second iteration in both SM1 and SM2. The distributions all seem to have a low mean value.

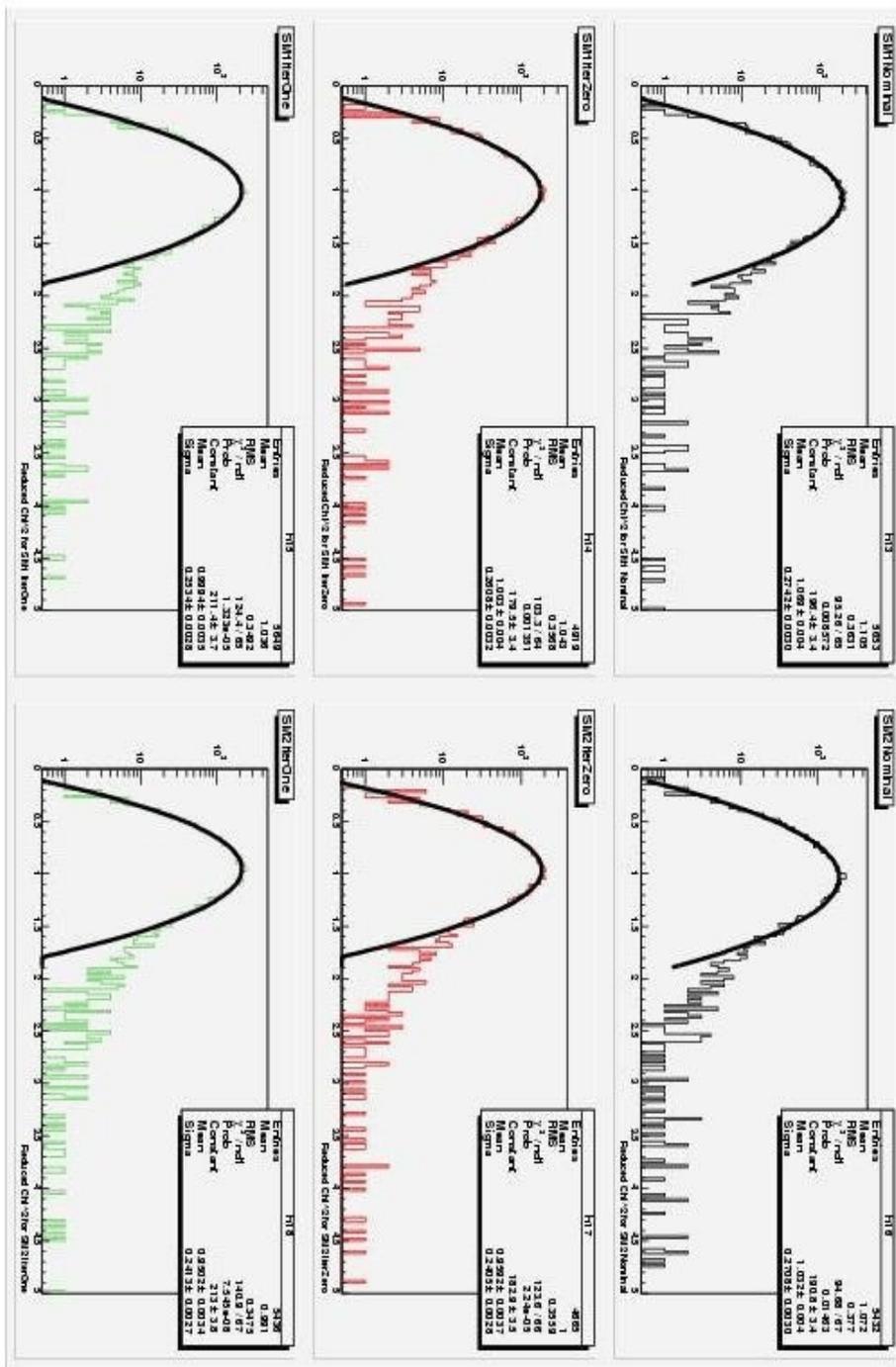


Figure 24: This shows the reduced  $\chi^2$  for long tracks. The mean position of the peak is improved after both the first and second iteration in both SM1 and SM2.

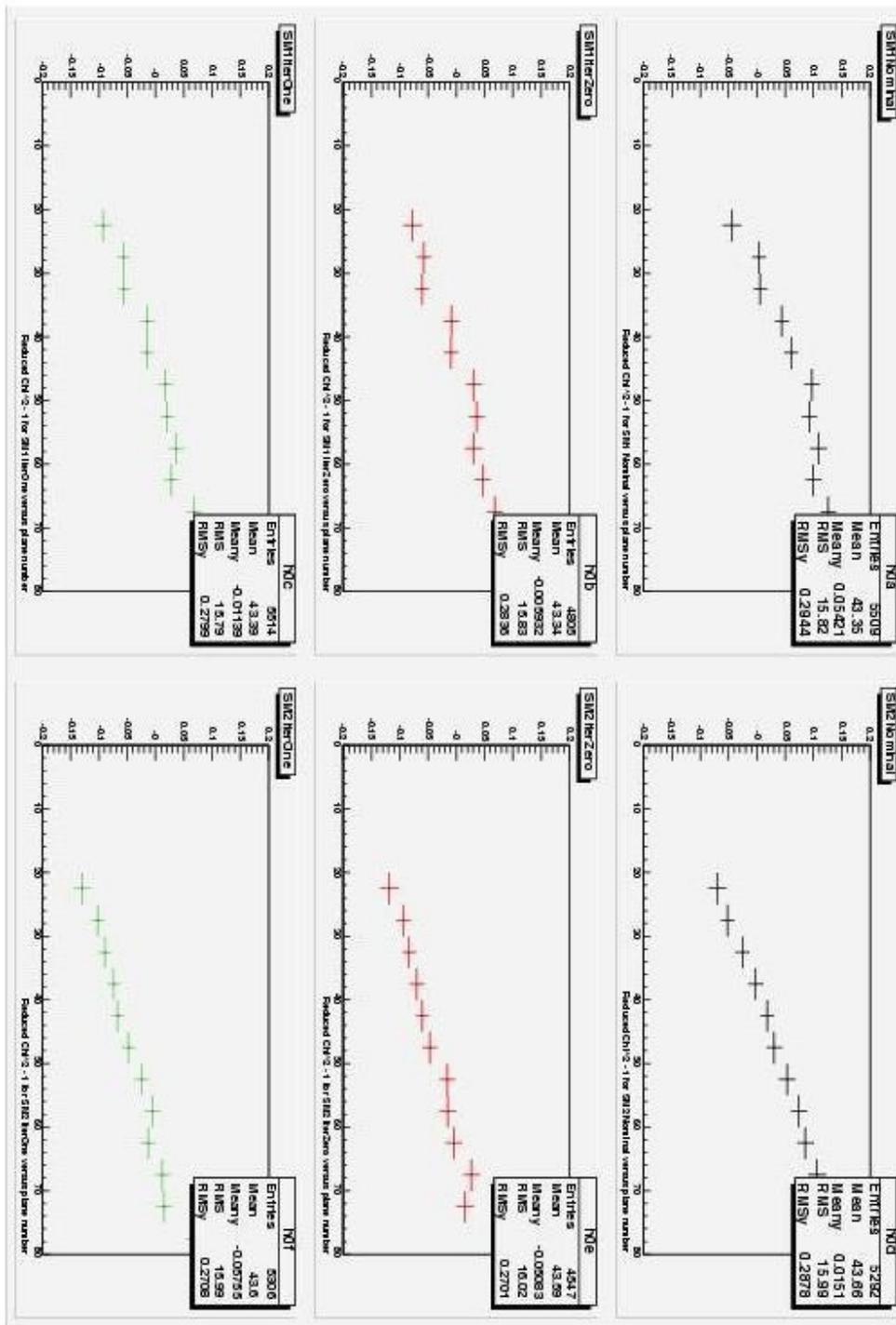


Figure 25: This shows the reduced  $\Delta R^2 - 1$ ) as a profile histogram plotting  $\Delta R^2$  versus number of tracklike planes. This only shows the tracks for ‘long events’ which have a reasonable distribution compared to the short tracks. This shows that the  $\Delta R^2$  increases with increasing track length.

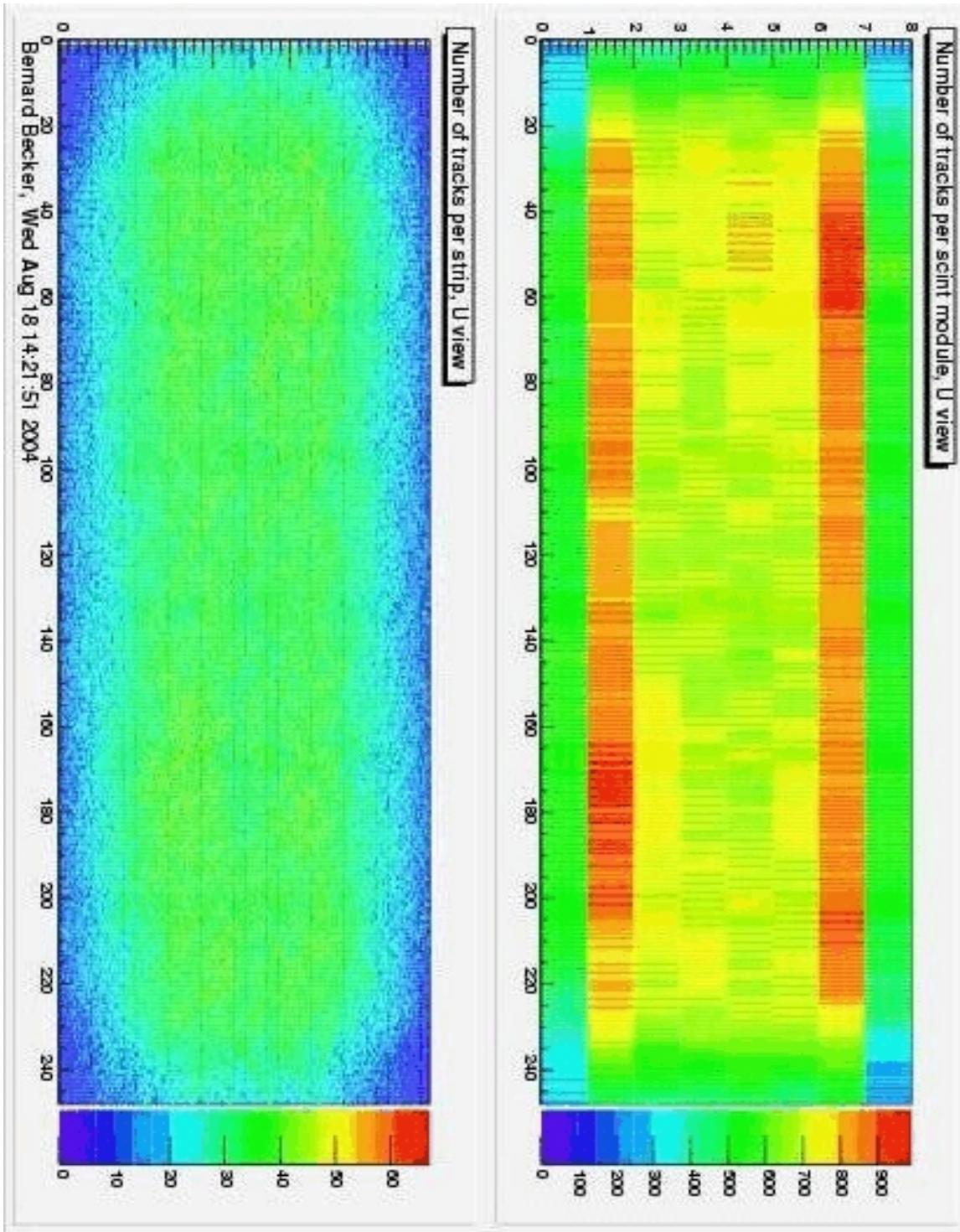


Figure 26: This shows the number of tracks used in the alignment that hit a particular module or strip in SM1. The detector acceptance drops off at the edge of the detector because of geometry.

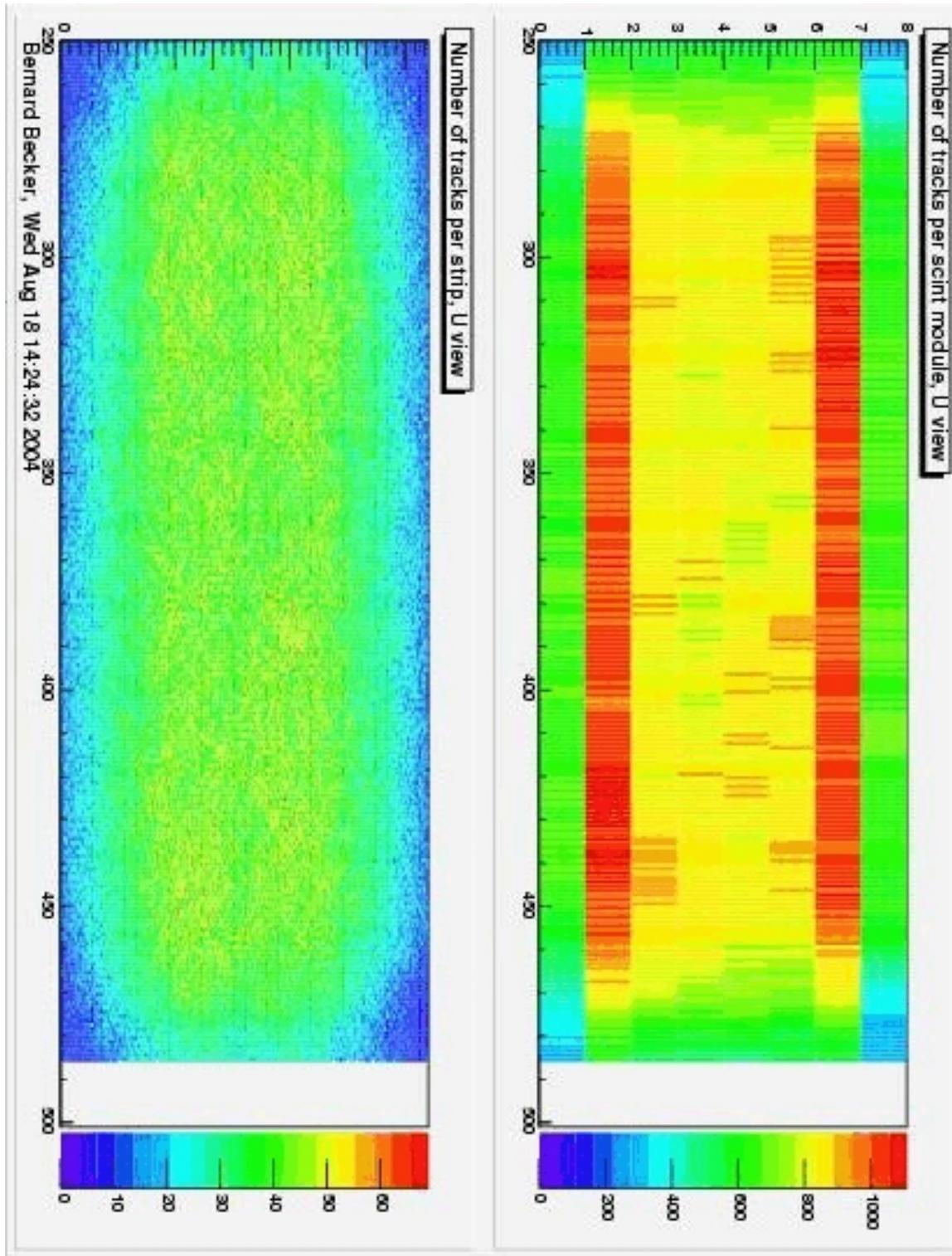


Figure 27: This shows the number of tracks used in the alignment that hit a particular module or strip in SM2. The detector acceptance drops off at the edge of the detector because of geometry.

# alignTrackerSM1.C

```
{
// This macro is derived from reco_R0.20.0_production.C
// Changed message levels to reduce output file size - DJB (1/29/04)
// Changed method for shower module to Config to avoid fatal run-time error
// in reconstruction - DJB (2/13/04)
// Added cut on FabPInInstall to take out veto shield.- DJB (3/1/04)

//macro for generating alignment constants for the far detector

//Link dynamic libraries
gSystem->Load("libNoiseFilter");
gSystem->Load("libFilterDigitSR");
gSystem->Load("libBField");
gSystem->Load("libNumericalMethods");
gSystem->Load("libSwimmer");
gSystem->Load("libRecoBase");
gSystem->Load("libDeMux");
gSystem->Load("libCandStripSR");
gSystem->Load("libCandSliceSR");
gSystem->Load("libCandTrackSR");
gSystem->Load("libCandClusterSR");
gSystem->Load("libCandShowerSR");
gSystem->Load("libCandFitTrackSR");
gSystem->Load("libMCNtuple");
gSystem->Load("libMCNtupleModule");
gSystem->Load("libCandEventSR");
gSystem->Load("libAstroUtil");
gSystem->Load("libCandNtupleSR");
gSystem->Load("libCandNtupleSRModule");
gSystem->Load("libCandFitTrackSA");

JobC jc;

//Create path
jc.Path.Create("Reco",
  "NoiseFilterModule::Ana "
  "RecordSetupModule::Get "
  "DigitListModule::Get "
  "DigitListModule::Reco "
  "FilterDigitListModule::Reco "
  "DeMuxDigitListModule::Reco "
  "DeMuxCosmicsModule::Ana "
  "StripSRListModule::Reco "
  "SliceSRListModule::Reco "
  "ClusterSRListModule::Reco "
  "ShowerSRListModule::Config "
  "TrackSRListModule::Reco "
  "FitTrackSRListModule::Reco "
  "EventSRListModule::Reco "
  "RecordSetupModule::Reco "
  "Output::Put");

//Input Parameters
jc.Input.Set("Format=input");
jc.Input.Set("Streams=DaqSnarl");

// Rollback database
// DbTableProxyRegistry is a CfgConfigurable
DbTableProxyRegistry& dbiCfg = DbTableProxyRegistry::Instance();
// dbiCfg.Set("Rollback:UGLIDBISCINTPLN = '2004-07-19'");
// dbiCfg.Set("Rollback:UGLIDBISCINTMDL = '2004-10-13'");
// dbiCfg.Set("Rollback:UGLIDBISTRIP = '2004-07-19'");
dbiCfg.Update();

//Set the output mode
jc.Path("Reco").Mod("Output").Cmd("DefineStream Config ConfigRecord");
jc.Path("Reco").Mod("Output").Set("Streams=DaqSnarl,Cand,Config");

//Set B=0 for SM2
BfldLoanPool* bfldpool = BfldLoanPool::Instance();
bfldpool->Set("NoFieldBeyondZ=0.0");
bfldpool->Update();

//Set Ugli to obey FabPInInstall
//UgliLoanPool* uglipool = UgliLoanPool::Instance();
```

```

//uglipool->Set("CutAppliesToVetoShield=1");
//uglipool->Update();

// Get the AlgFactory
AlgFactory &af = AlgFactory::GetInstance();

// AlgDeMuxDigitList AlgConfig parameters
AlgHandle ah = af.GetAlgHandle("AlgDeMuxDigitList", "default");
AlgConfig &acd = ah.GetAlgConfig();
acd.UnlockValues();
acd.Set("NormalizeWeights", 1); // Normalize weights to 1 if non-zero
acd.Set("TrimHyps", 1); // Drop "0" weights if neg., or keep top N
acd.LockValues();

//DigitListModule parameters
jc.Path("Reco").Mod("DigitListModule").Set("ListsToMake=1");

//Reco's filter parameters
jc.Path("Reco").Mod("StripSRListModule").Set("ListIn=candidigitlist");
jc.Path("Reco").Mod("FilterDigitListModule").Set("FilterDigitListAlgorithm=AlgFilterDigitListSR");
jc.Path("Reco").Mod("FilterDigitListModule").Set("SwitchPersToTemp=1");
jc.Path("Reco").Mod("StripSRListModule").Set("BegPlane=1");
jc.Path("Reco").Mod("StripSRListModule").Set("EndPlane=248");

// set the misalignment error in mm
jc.Path("Reco").Mod("EventSRListModule").Set("FilterEvent=1");
jc.Path("Reco").Mod("TrackSRListModule").Set("MisalignmentError=3.5");
jc.Path("Reco").Mod("FitTrackSRListModule").Set("MisalignmentError=3.5");

//Ntuple record has its own output file so needs its own output module
jc.Path.Create("NtpSR", "NtpSRModule::Reco "
              "Output::Put ");
jc.Path("NtpSR").Mod("Output").Cmd("DefineStream NtpSR NtpSRRecord");
jc.Path("NtpSR").Mod("Output").Set("Streams=NtpSR");
jc.Path("NtpSR").Mod("Output").Set("FileName=ntupleSR.root");
jc.Path.Attach("Reco", "NtpSR");

// Ntuple abridged record
jc.Path.Create("NtpSRFilter", "NtpSRFilterModule::Reco "
              "Output::Put ");
jc.Path("NtpSRFilter").Mod("Output").Cmd("DefineStream NtpSR NtpSRRecord");
jc.Path("NtpSRFilter").Mod("Output").Set("Streams=NtpSR");
jc.Path("NtpSRFilter").Mod("Output").Set("FileName=ntupleSR.sub.root");
jc.Path.Attach("Reco", "NtpSRFilter");

//Configure the message service
jc.Msg.SetLevel("Plex", "Error");
jc.Msg.SetLevel("FitTrackSR", "Fatal");
jc.Msg.SetLevel("Calibrator", "Fatal");
jc.Msg.SetLevel("Bfld", "Fatal");
jc.Msg.SetLevel("Ugli", "Fatal");

jc.Path("Reco").Run();

jc.Path("Reco").Report();

//Get Message Statistics
jc.Msg.Stats();

}

```

## align1.C

```

/*****
* Macro Name: align.C
* Date: 8-13-03
* Author: B. Viren (with large modification by B. Becker)
* Purpose: To align the MINOS FD using muons.
* E-Mail: bbecker@physics.umn.edu
* Use with minosoft R0.21.0 or later
*-----
* Command to run:
* > loon -bq 'align.C("output.root")' fl.cand.root ....
*****/

```

```

*-----*
* output.root : Is the file which will have the aligned *
* module positions (results of alignment) *
*-----*
* f1.cand.root : Is the candidate file which is generated *
* by running align_tracker.C . After a first round of *
* processing in which the output is used to generate a set *
* of new tpos positions for the DB, align_tracker.C MUST *
* BE RAN AGAIN and NEW f1.cand.root MUST BE MADE. *
*-----*
* Features: Multiple candidate files can be ran by adding *
* them to command line: f1.cand.root f2.cand.root .... *
*-----*
* There are cuts applied on the candidates by using the *
* NtpSR stream. These cuts and the files being read in are *
* hard coded in the macro. *
* *
*****/

```

```

class JobC;
JobC* align_init(const char* pathname,
                 const char* histname);

void align ( const char* histname)
{
    cout << "aligning with: " << histname << endl;
    const char* pathname = "Align";
    JobC* jc = align_init(pathname,histname);
    jc->Path(pathname).Run();
    delete jc;
    jc=0;
}

```

```

// Loads in the all the lib's needed for the job
// Make sure that this list of lib's matches the list
// of libs used in align_tracker.C

```

```

JobC* align_init(const char* pathname,
                 const char* histname)
{
    cerr << "Starting: " << histname << endl;

    const char* libs[80] = {

        "libAlignment.so",
        "libNoiseFilter.so",
        "libBField",
        "libNumericalMethods",
        "libSwimmer",
        "libDeMux",
        "libAltDeMux",
        "libAtNuReco",
        "libCandStripSR",
        "libCandSliceSR",
        "libCandTrackSR",
        "libCandClusterSR",
        "libCandShowerSR",
        "libCandFitTrackSR",
        "libCandEventSR",
        "libTimeCalibratorSR",
        "libAstroUtil",
        "libMCNtuple",
        "libMCNtupleModule",
        "libCandNtupleSR",
        "libCandNtupleSRModule",
    }
}

```

```

"libCandFitTrackSA",
"libTruthHelperNtuple",
"libTruthHelperNtupleModule",
0

};

for (int i=0; libs[i]; ++i) {
    gSystem->Load(libs[i]);
}

// The following three lines do all the actual alignment

// Rollback database
// DbiproxyRegistry is a CfgConfigurable
DbiproxyRegistry& dbiCfg = DbiproxyRegistry::Instance();
// dbiCfg.Set("Rollback:UGLIDBISCINTPLN = '2004-07-19'");
dbiCfg.Set("Rollback:UGLIDBISCINTMDL = '2004-10-21'");
// dbiCfg.Set("Rollback:UGLIDBISTRIP = '2004-07-19'");
dbiCfg.Update();

JobC* jc = new JobC;
jc->Path.Create(pathname,
    "Alignment::Ana ");

// The following code sets the NtpSR files to be read and set the cuts to be used.

jc->Input.Set("Streams=Cand, DaqSnarl, NtpSR");
jc->Input.Set("Format=input");
jc->Input.AddFile("/data/minos-pc3/bbecker/alignment_files/alignment_files/NewAlignmentRun/SM1/PassFour/n*.root", "NtpSR");
jc->Input.Select("NtpSR", "(evthdr.ntrack==1)&&(dmxstatus.ismultimuon==0)&&(trk.plane.ntrklike >= 20)&&(abs(trk.vtx.dcosx-trk.lin.dcosx) <= 0.0015)&&(abs(trk.vtx.dcosy-trk.lin.dcosy) <= 0.0015)&&(abs(trk.vtx.dcosz-trk.lin.dcosz) <= 0.0015)");

// Error messages

jc->Msg.SetLevel("BubJobC", "Fatal");
jc->Msg.SetLevel("BubCand", "Fatal");
jc->Msg.SetLevel("DMX", "Fatal");
jc->Msg.SetLevel("Ugli", "Fatal");
jc->Msg.SetLevel("Plex", "Fatal");
jc->Msg.SetLevel("Alignment", "Fatal");
jc->Msg.SetLevel("Calibrator", "Fatal");
jc->Msg.SetLevel("Dbi", "Fatal");
jc->Msg.SetLevel("SigCor Calibrator", "Fatal");
jc->Msg.SetLevel("Time Calibrator", "Fatal");
jc->Msg.SetLevel("MuonCalibrator", "Fatal");
jc->Msg.SetLevel("PE Calibrator", "Fatal");
jc->Msg.SetLevel("MapperCalibrator", "Fatal");
jc->Msg.SetLevel("NoiseFilter", "Fatal");

// Requires the use of the fit track. This is important as the "fit tracks" have a
// higher quality (track fitter remove bad digits) then the "tracks"

Registry ac = jc->Path(pathname).Mod("Alignment").DefaultConfig();
ac.UnlockValues();
ac.Set("TrackName", "CandFitTrackSRLList"); // Was CandTrackSRLList
ac.Set("TrackType", "CandFitTrackListHandle"); // Was CandTrackListHandle
ac.Set("HistFileName", histname);
jc->Path(pathname).Mod("Alignment").Config(ac);
return jc;
}

}

GetInfo.C

{
TFile *alignhist = TFile::Open("SM1/PassOne/SM1PassOneRollback.root");

```

```
Double_t Offset = 0;
```

```
TCanvas *T1 = new TCanvas("T1","U and V offsets 1");
T1->Divide(1,2);
TH1F *U1 = new TH1F("U","U",500,-0.025,0.025);
TH1F *V1 = new TH1F("V","V",500,-0.025,0.025);
TCanvas *T2 = new TCanvas("T2","U and V offsets 2");
T2->Divide(1,2);
TH1F *U2 = new TH1F("U","U",500,-0.025,0.025);
TH1F *V2 = new TH1F("V","V",500,-0.025,0.025);
TCanvas *T3 = new TCanvas("T3","U and V offsets 3");
T3->Divide(1,2);
TH1F *U3 = new TH1F("U","U",500,-0.025,0.025);
TH1F *V3 = new TH1F("V","V",500,-0.025,0.205);
TCanvas *T4 = new TCanvas("T4","U and V offsets 4");
T4->Divide(1,2);
TH1F *U4 = new TH1F("U","U",500,-0.025,0.025);
TH1F *V4 = new TH1F("V","V",500,-0.025,0.025);
TCanvas *T5 = new TCanvas("T5","U and V offsets 5");
T5->Divide(1,2);
TH1F *U5 = new TH1F("U","U",500,-0.025,0.025);
TH1F *V5 = new TH1F("V","V",500,-0.025,0.025);
TCanvas *T6 = new TCanvas("T6","U and V offsets 6");
T6->Divide(1,2);
TH1F *U6 = new TH1F("U","U",500,-0.025,0.025);
TH1F *V6 = new TH1F("V","V",500,-0.025,0.025);
TCanvas *T7 = new TCanvas("T7","U and V offsets 7");
T7->Divide(1,2);
TH1F *U7 = new TH1F("U","U",500,-0.025,0.025);
TH1F *V7 = new TH1F("V","V",500,-0.025,0.025);
TCanvas *T8 = new TCanvas("T8","U and V offsets 8");
T8->Divide(1,2);
TH1F *U8 = new TH1F("U","U",500,-0.025,0.025);
TH1F *V8 = new TH1F("V","V",500,-0.025,0.025);
TCanvas *T9 = new TCanvas("T9","U and V offsets ");
T9->Divide(1,2);
TH1F *U = new TH1F("U","U",1000,-0.025,0.025);
TH1F *V = new TH1F("V","V",1000,-0.025,0.025);
```

```
for (Int_t i=2;i<=249;i=i+1)
{
  while(i%2==0)
  {
    for (Int_t j=1;j<=8;j++)
    {
      Offset = ResidHistU00->GetBinContent(i,j)/NumberOfTracksU->GetBinContent(i,j);
      cout << "Plane Number " << i-1 << " Module Number " << j << " Offset " << Offset << " m " << "\n";
      U->Fill(Offset);
      if(j==1){
        Offset = ResidHistU00->GetBinContent(i,j)/NumberOfTracksU->GetBinContent(i,j);
        cout << "Plane Number " << i-1 << " Module Number " << j << " Offset " << Offset << " m " << "\n";
        U1->Fill(Offset);
      } else if(j==2) {
        Offset = ResidHistU00->GetBinContent(i,j)/NumberOfTracksU->GetBinContent(i,j);
        cout << "Plane Number " << i-1 << " Module Number " << j << " Offset " << Offset << " m " << "\n";
        U2->Fill(Offset);
      } else if(j==3) {
        Offset = ResidHistU00->GetBinContent(i,j)/NumberOfTracksU->GetBinContent(i,j);
        cout << "Plane Number " << i-1 << " Module Number " << j << " Offset " << Offset << " m " << "\n";
        U3->Fill(Offset);
      } else if(j==4) {
        Offset = ResidHistU00->GetBinContent(i,j)/NumberOfTracksU->GetBinContent(i,j);
        cout << "Plane Number " << i-1 << " Module Number " << j << " Offset " << Offset << " m " << "\n";
        U4->Fill(Offset);
      } else if(j==5) {
        Offset = ResidHistU00->GetBinContent(i,j)/NumberOfTracksU->GetBinContent(i,j);
```



```
T1->cd(1);
gStyle->SetOptFit(1111);
T1->SetFillColor(1);
T1->SetFillStyle(3001);
U1->Draw();
U1->Fit("gaus");
U1->GetXaxis()->SetTitle("U_1 offsets in meters");
U1->SetFillColor(kBlue);
T1->cd(2);
V1->Draw();
V1->Fit("gaus");
V1->GetXaxis()->SetTitle("V_1 offsets in meters");
V1->SetFillColor(kRed);
T1->Print("UV1.ps");
T1->Print("UV1.jpeg");
```

```
T2->cd(1);
gStyle->SetOptFit(1111);
T2->SetFillColor(1);
T2->SetFillStyle(3001);
U2->Draw();
U2->Fit("gaus");
U2->GetXaxis()->SetTitle("U_2 offsets in meters");
U2->SetFillColor(kBlue);
T2->cd(2);
V2->Draw();
V2->Fit("gaus");
V2->GetXaxis()->SetTitle("V_2 offsets in meters");
V2->SetFillColor(kRed);
T2->Print("UV2.ps");
T2->Print("UV2.jpeg");
```

```
T3->cd(1);
gStyle->SetOptFit(1111);
T3->SetFillColor(1);
T3->SetFillStyle(3001);
U3->Draw();
U3->Fit("gaus");
U3->GetXaxis()->SetTitle("U_3 offsets in meters");
U3->SetFillColor(kBlue);
T3->cd(2);
V3->Draw();
V3->Fit("gaus");
V3->GetXaxis()->SetTitle("V_3 offsets in meters");
V3->SetFillColor(kRed);
T3->Print("UV3.ps");
T3->Print("UV3.jpeg");
```

```
T4->cd(1);
gStyle->SetOptFit(1111);
T4->SetFillColor(1);
T4->SetFillStyle(3001);
U4->Draw();
U4->Fit("gaus");
U4->GetXaxis()->SetTitle("U_4 offsets in meters");
U4->SetFillColor(kBlue);
T4->cd(2);
V4->Draw();
V4->Fit("gaus");
V4->GetXaxis()->SetTitle("V_4 offsets in meters");
V4->SetFillColor(kRed);
T4->Print("UV4.ps");
T4->Print("UV4.jpeg");
```

```
T5->cd(1);
gStyle->SetOptFit(1111);
T5->SetFillColor(1);
```

```
T5->SetFillStyle(3001);
U5->Draw();
U5->Fit("gaus");
U5->GetXaxis()->SetTitle("U_5 offsets in meters");
U5->SetFillColor(kBlue);
T5->cd(2);
V5->Draw();
V5->Fit("gaus");
V5->GetXaxis()->SetTitle("V_5 offsets in meters");
V5->SetFillColor(kRed);
T5->Print("UV5.ps");
T5->Print("UV5.jpeg");
```

```
T6->cd(1);
gStyle->SetOptFit(1111);
T6->SetFillColor(1);
T6->SetFillStyle(3001);
U6->Draw();
U6->Fit("gaus");
U6->GetXaxis()->SetTitle("U_6 offsets in meters");
U6->SetFillColor(kBlue);
T6->cd(2);
V6->Draw();
V6->Fit("gaus");
V6->GetXaxis()->SetTitle("V_6 offsets in meters");
V6->SetFillColor(kRed);
T6->Print("UV6.ps");
T6->Print("UV6.jpeg");
```

```
T7->cd(1);
gStyle->SetOptFit(1111);
T7->SetFillColor(1);
T7->SetFillStyle(3001);
U7->Draw();
U7->Fit("gaus");
U7->GetXaxis()->SetTitle("U_7 offsets in meters");
U7->SetFillColor(kBlue);
T7->cd(2);
V7->Draw();
V7->Fit("gaus");
V7->GetXaxis()->SetTitle("V_7 offsets in meters");
V7->SetFillColor(kRed);
T7->Print("UV7.ps");
T7->Print("UV7.jpeg");
```

```
T8->cd(1);
gStyle->SetOptFit(1111);
T8->SetFillColor(1);
T8->SetFillStyle(3001);
U8->Draw();
U8->Fit("gaus");
U8->GetXaxis()->SetTitle("U_8 offsets in meters");
U8->SetFillColor(kBlue);
T8->cd(2);
V8->Draw();
V8->Fit("gaus");
V8->GetXaxis()->SetTitle("V_8 offsets in meters");
V8->SetFillColor(kRed);
T8->Print("UV8.ps");
T8->Print("UV8.jpeg");
```

```
T9->cd(1);
gStyle->SetOptFit(1111);
T9->SetFillColor(1);
T9->SetFillStyle(3001);
U->Draw();
U->Fit("gaus");
```

```
U->GetXaxis()->SetTitle("U offsets in meters");
U->SetFillColor(kBlue);
T9->cd(2);
V->Draw();
V->Fit("gaus");
V->GetXaxis()->SetTitle("V offsets in meters");
V->SetFillColor(kRed);
T9->Print("UV.ps");
T9->Print("UV.jpeg");
```

```
}
```