LC-TOOL-2014-010
`http://www-flc.desy.de/lcnotes`

Analysis Tools in MarlinTPC for pad-based readout systems

Astrid Münnich
DESY, Hamburg

January 9, 2015

Abstract

Common analysis tools were developed for the testbeam data taken by the LCTPC collaboration with a large TPC prototype to ensure coherent results between the different readout technologies under investigation. A basic knowledge of the modules and data format as well as the software framework is implied in this note.

Contents

1 Introduction

Different micro-pattern gas detector modules are being tested within the LCTPC Collaboration [1] in the Large Prototype TPC [2] at the testbeam facility at DESY [3]. Currently two different GEM module designs [4, 5] and a Micromegas module design [6] are being evaluated. In order to make comparisons and study the differences of the module performance a common analysis is mandatory. Therein, certain steps in the reconstruction have to be different for GEMs and Micromegas. Some of those differences are due to the technology itself, but some are due to the different electronics used. As all of the modules have a pad-based readout a common analysis is feasible. Once hits are found by the reconstruction, the same procedure and tools should be used to obtain the analysis results e.g. point resolution or the evaluation of distortions. A set of analysis tools has been agreed upon and implemented as Marlin processors [7] within MarlinTPC [8]. In this note the tools will be described followed by some practical information which serve as a User Guide.

2 Analysis Tools

The goal is to provide all necessary tools to ensure a procedure that allows to compare all proposed technologies on equal ground. This requires common and exchangeable code, but also a set of rules and conventions. For this purpose several Marlin processors were developed, which take a reconstructed track as input. Hits and pulses have to be accessed through the track. In addition, some common rules and conventions have been established concerning the definition of certain properties and the application of cuts.
Common tools and functionality should be available as classes to be used in the processors, e.g. a class to evaluate the width of a distribution in a histogram. The results of these new processors are stored in ROOT [9] files using the AIDA [10] processor. This will then result in one ROOT file per run with directories for each processor containing the histograms and fit results. Each processor is accompanied by a root macro to create the final plots.

2.1 Analysis Plots

Two different categories of plots have been defined. The first one are the performance plots which can serve as a first base to compare the technology options. The second category are plots to ensure the data quality and show the results in a more detailed way.

Performance Plots

- Resolution in $r\phi$ and z for B=0, 1T at $\theta = 0°$ and $\phi = 0°$. Two curves should be shown in the resolution curve. The first should show the resolution of the best row only indicating what the module could do. The second should combine all rows, but separate the effects caused by distortion. One simple way to achieve this is to evaluate the resolution of each row separately and then calculate the mean of all rows. This curve shows the overall performance of the module neglecting displacements of hits due to field distortions, thus gives an indication of the resolution if these distortions effects can be corrected.

- Distortions: mean deviation of all hits in a row with respect to the track. Evaluate those for each row in ($r\phi$,z) for B=0, 1T at $\theta = 0°$ and $\phi = 0°$.
- Diffusion: The width of the Pad Response Function in dependence on the drift distance
- Hit Efficiency for one exemplary row as a function of the drift distance
- Hit Efficiency with dependence on the row

Quality Plots

- Charge sharing at small drift distance: charge of the central pad in a hit vs charge on its left neighbor
- Charge versus drift distance. Charge as given by the MEAN90 from the hit charge spectrum
- Number of hits with a pulse in over-range and number of hits containing a dead channel vs drift distance
- Distortions after correction for field distortions
- Resolution vs drift, uncorrected and corrected
- Resolution vs row, uncorrected and corrected
- Example resolution distribution and fit
- Compare resolution extraction method in z (fit vs RMS90)
- Selection efficiency: Fraction of events left after cuts vs drift
- Resolution vs drift distance for different tracking methods, or triplet resolution method
- Track Parameters
- Alignment parameters of the modules (B=0, 1T) (GBL + millepede, not a processor)

The processors to extract the necessary information from the data sets to obtain these plots are described in the following.

2.2 Analysis Processors

In the following the available analysis processors for the basic common data analysis are listed. The example plots given for each processor are based on the DESY GEM module testbeam campaign in 2013.

AnalysisCutSelectorProcessor

Description:	The task of this processor is to apply analysis cuts and to evaluate the event selection efficiency for those cuts. In addition, as this will be the first processor to run in the analysis chain, it will extract the information about the z position of the given run. This is needed later for plots combining different runs e.g. for plotting quantities' dependence on the drift distance.
Aida output:	Histogram of cut flow: bin 1 all events, bin 2 after 1st cut and so on; histogram of z position and the fit results.
Root macro:	Plot the fraction of events left after cuts for different drift distances (fig. 1).

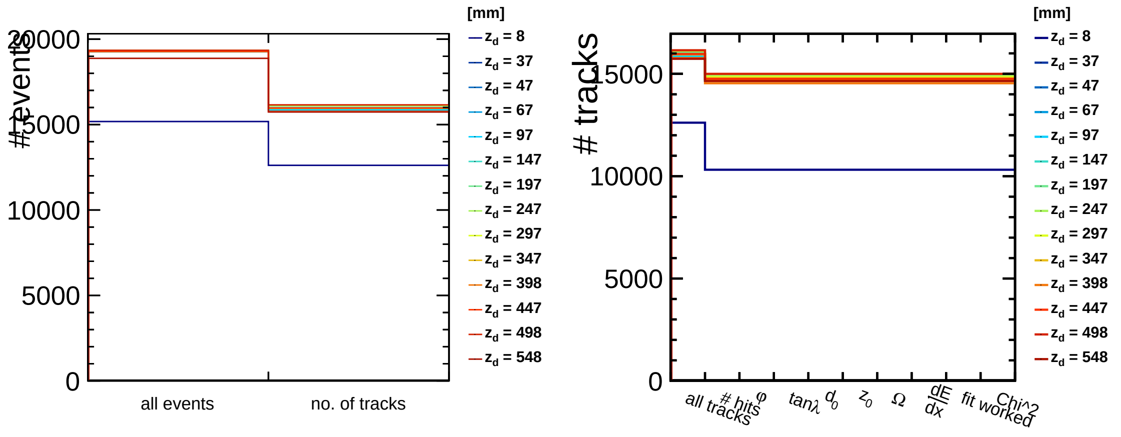

Figure 1: Cut flow of events (left) and on track level (right).

TrackParametersDistributionProcessor

Description:	Plots all track parameters.
Aida output:	Track parameter histograms.
Root macro:	Plot histograms (fig. 2).

HitQualityProcessor

Description:	Count number of hits in over-range and hits with a dead channel.
Aida output:	Number of hits in over-range, number of hits with a dead channel.
Root macro:	Plot numbers vs drift distance (fig. 3).

TrackHitEfficiencyProcessor

Description:	Evaluate the hit efficiency on a track for each row in a simple approach which counts the occurrence on each row of a reconstructed hit for all tracks.

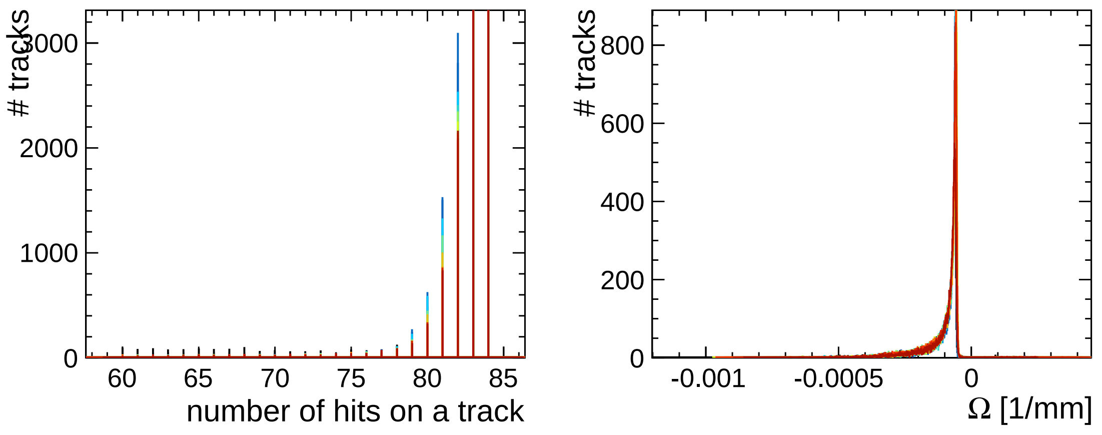

Figure 2: Example of track parameter distributions: Number of hits on track (left) and the distribution of the curvature Ω (right).

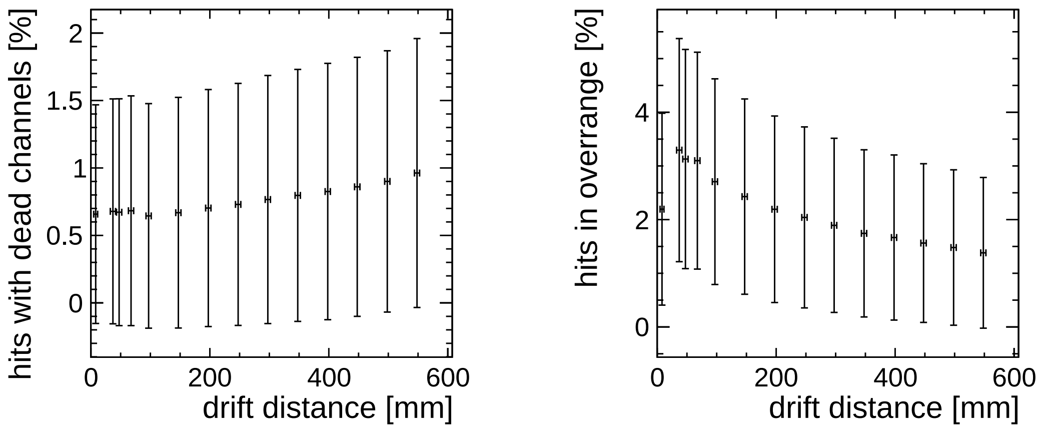

Figure 3: Fraction of dead channels (left) and of hits containing a pulse in over-range (right) as function of the drift distance.

Aida output:	Histogram of efficiency=(number of hits on row)/(number of tracks) vs row.
Root macro:	Plot efficiency vs row for different drift distances (fig. 4).

ChargeQualityProcessor

Description:	Evaluate hit charge behavior vs drift distance.
Aida output:	Hit charge spectrum plus result of MEAN90.
Root macro:	Plot MEAN90 result vs drift distance and vs row (fig. 5).

DistortionPerformanceProcessor

Description:	Calculate distortions in $r\phi$ (along row) and z.

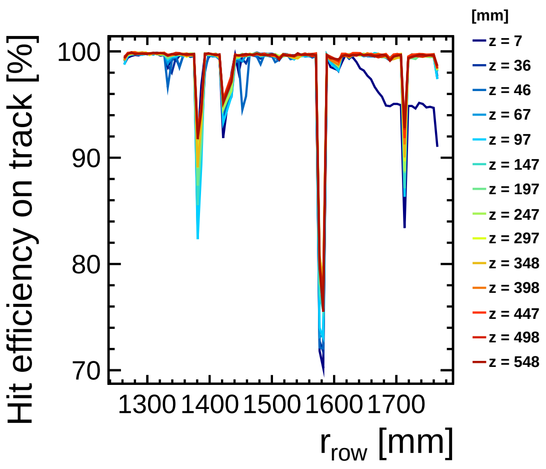

Figure 4: Number of hits found on each row normalized to the number of tracks.

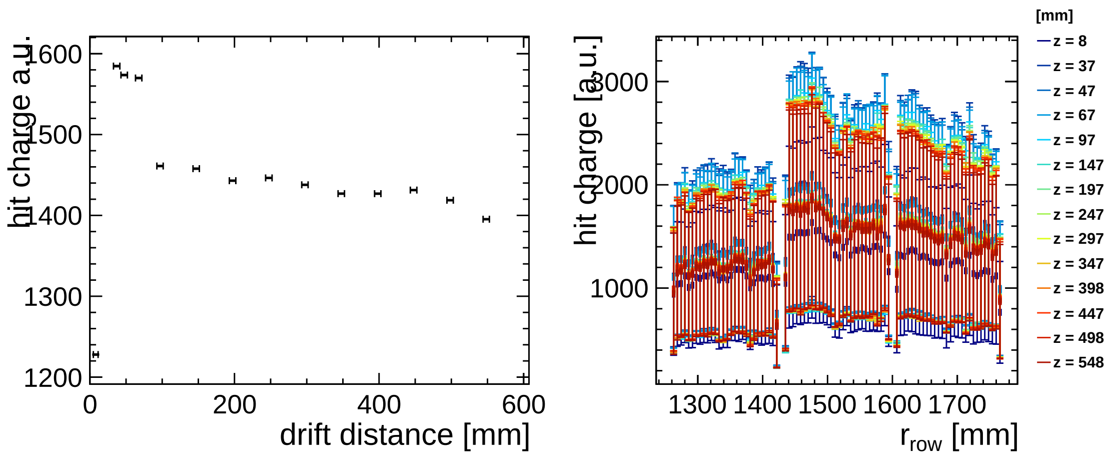

Figure 5: The mean charge of a hit as function of drift distance (left) and measurement row (right).

Aida output:	Histogram of row binned mean hit distance to track.
Root macro:	Plot distortions vs row for different z positions (fig. 6).

ResolutionPerformanceProcessor

Description:	Calculate resolution for given run in $r\phi$ (along row) and z.
Aida output:	Row binned residual histograms and results based on chosen evaluation method fit, RMS90 or 5 RMS cut.
Root macro:	Plot vs drift: best row only; all rows based on separate sigma evaluation per row and then taking the mean of all (eliminating effects of distortion displacement); all rows in one sigma for uncorrected and corrected tracks; resolution vs row (fig. 7,8).

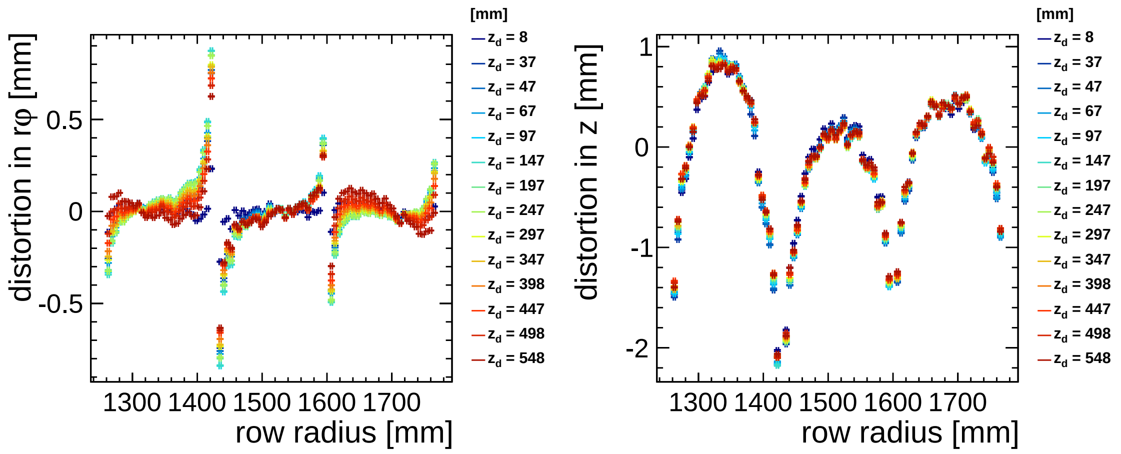

Figure 6: Distortion in $r\phi$ (left) and z direction (right) versus pad row.

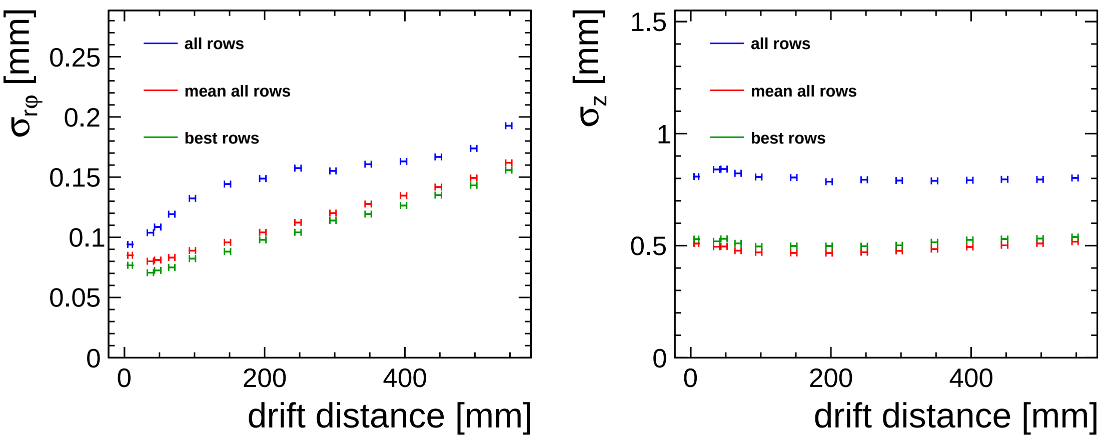

Figure 7: Different methods of evaluating the single point resolution in $r\phi$ (left) and z direction (right). Evaluating all rows in one step is sensitive to the distortions. To disentangle that effect either the mean of each single row or simply one row alone can be used to evaluate the resolution.

TripletResolutionPerformanceProcessor

Description:	Calculate resolution for a given run in $r\phi$ (along row) and z based on the evaluation of hit triplets. This offers a method to obtain the resolution independent from the track fit.
Aida output:	Row binned residual histograms and results based on chosen evaluation method fit, RMS90 or 5 RMS cut.
Root macro:	Plot vs drift: best row only; all rows based on separate sigma evaluation per row and then (fig. 9) taking the mean of all (eliminating effects of distortion displacements); all rows in one sigma for uncorrected and corrected tracks; resolution vs row.

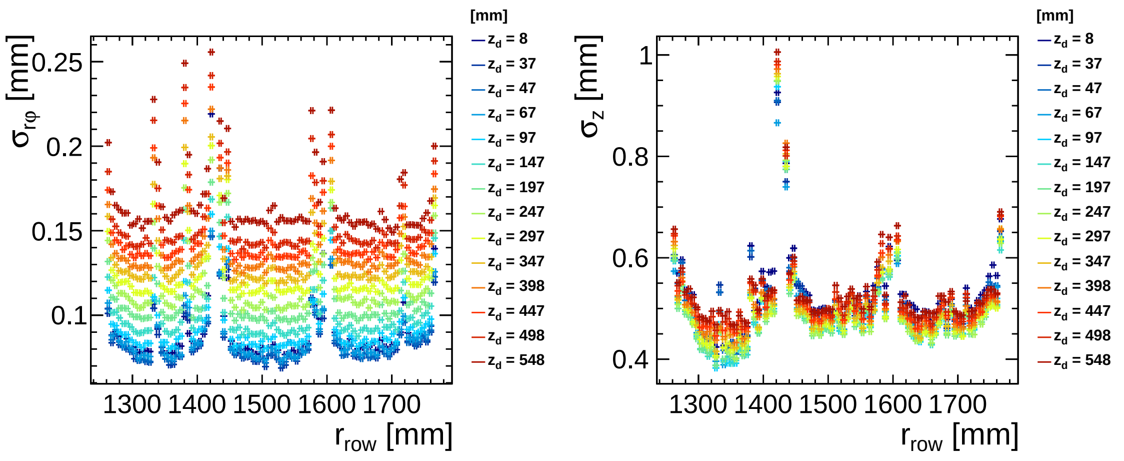

Figure 8: Dependence of the resolution in $r\phi$ (left) and z direction (right) on the position along the module. Distortions do not only displace the charge but also affect the resolution at the border of the modules.

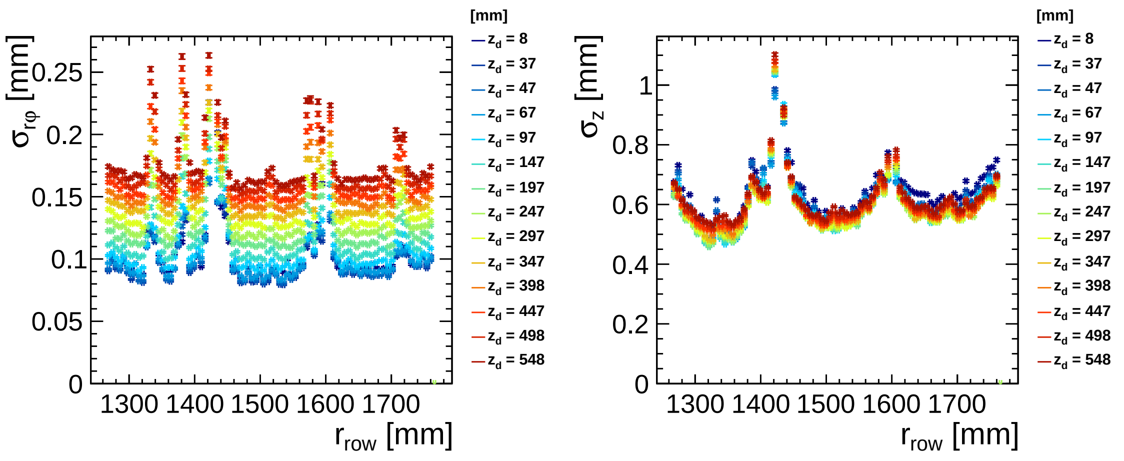

Figure 9: Dependence of the resolution based on triplets in $r\phi$ (left) and z direction (right) on the position along the module.

PRFPerformanceProcessor

Description:	Evaluate PRF distribution and fit width.
Aida Output:	PRF histogram and fit.
Root macro:	Mean PRF width vs drift distance and PRF width vs row (fig. 10).

2.3 Common Tools

HistogramWidthExtractionTools

An important task in performance evaluation is the extraction of the width of a distribution that is Gaussian-like but usually with a larger contribution in the tails. The HistogramWidthExtractionTool provides methods to calculate the RMS90 and MEAN90 as well as fitting procedures to return a 3 σ Gaussian fit and (the previous

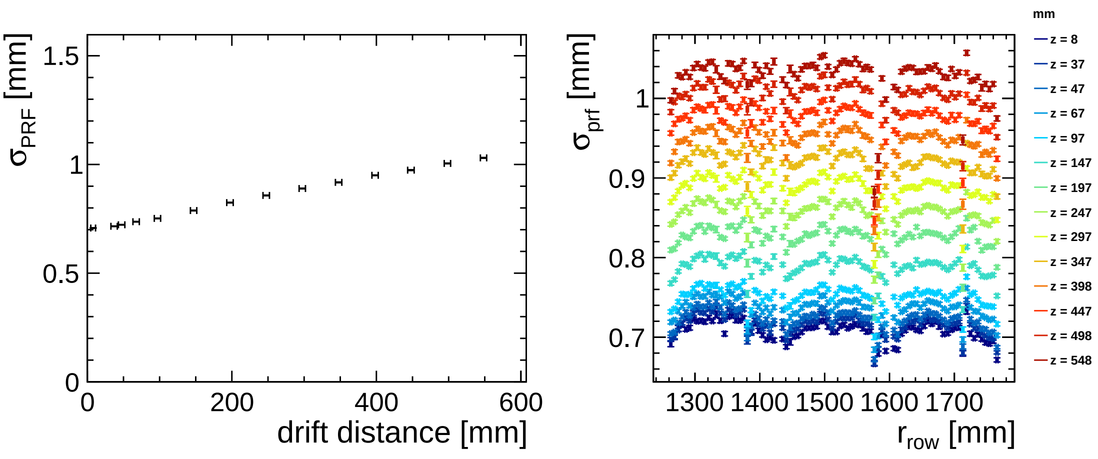

Figure 10: Width of the charge cloud over a pad row called the Pad Response Function (PRF) as function of the drift distance (left) and the row number (right).

agreement) an RMS based determination on an iterative 5 RMS cut. A common definition of what is the width of a distribution which is quoted as resolution is important as different definitions yield quite different results as shown in fig. 11.

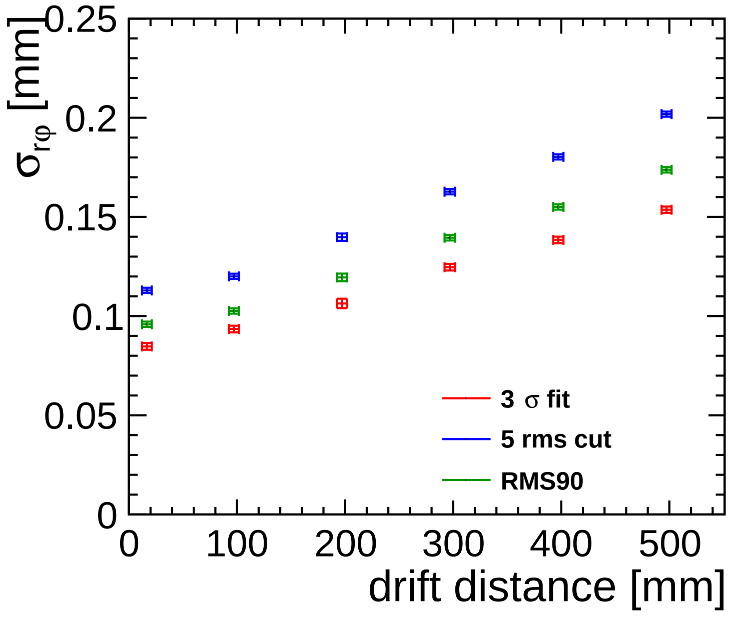

Figure 11: Comparison of different methods to extract the width from a distribution, here with the $r\phi$ resolution as example.

ModuleDistortionCalculator

This tool evaluates the distance of a point to the track on its measurement surface, meaning along a row. It must therefore take into account the details of the pad layout defined in GEAR [11]. This measurement surface can either be a circle or a straight line depending on the pad geometry. The track itself can also either be a circle or a straight line in the $r\phi$ plane given whether a magnetic field is present or not. In the z direction the track is always a straight line. The main task of calculating these intersection between the different combinations of circles and lines is outsourced to another tool in MARLINTPC called the IntersectionCalculator. This illustrates once more the benefits of encapsulating generic and often used procedures into classes usable by different applications and for various tasks.

3 User Guide

3.1 Analysis Flow

After the reconstruction is done, we require the following collections to be present in a lcio file:

1. Pulses
2. Hits
3. optional: Hits (corrected for distortions)
4. Tracks
5. optional: Tracks (corrected for distortions)

The input collection for all processors are the track objects. Hits and pulses are accessed through the track.
The common analysis contains the following processors that are defined in a MARLIN steering file:

```
<execute>
    <processor name="MyAIDAProcessor"/>
    <processor name="MyAnalysisCutSelectorProcessor"/>
    <processor name="MyTrackParametersDistributionProcessor"/>
    <processor name="MyHitQualityProcessor"/>
    <processor name="MyTrackHitEfficiencyProcessor"/>
    <processor name="MyChargeQualityProcessor"/>
    <processor name="MyDistortionPerformanceProcessor"/>
    <processor name="MyResolutionPerformanceProcessor"/>
    <processor name="MyTripletResolutionPerformanceProcessor"/>
    <processor name="MyPRFPerformanceProcessor"/>
 </execute>
```

These processors will produce one ROOT file, which should follow the naming convention of aida_runRUNNUMBER.root. Within this ROOT file there is a folder for each processor containing results in tuples and histograms.
In order to generate the plots each processor is accompanied by a ROOT script, which can be executed in one go within ROOT:

```
.L $MARLINTPC/analysis/rootscripts/LoadAnalysisScripts.C
LoadAnalysisScripts()
```

```
ProcessAllAnalyses("pathforOutput/","pathtoAIDAfiles/",
                   #of runs, runnumber1, runnumber2,....)
```

3.2 Analysis Rules & Conventions

The number of cuts should be limited and the cut itself not too stringent. We agreed to apply all cuts on track level with the exception of throwing full events away for special reasons like the observed gain drops in the Asian module.
Cuts being considered are the following:

- Single track events
- Number of hits on tracks $>$ 80% of active rows
- (Event cut: e.g. gain drop)
- (Track parameters (omega, angle): Still needs to be evaluated if this is necessary.)

The following conventions and definitions are used:

- The resolution and the distortions are to be evaluated along the row meaning in $r\phi$ not in the xy projection (label axes accordingly $\sigma_{r\phi}$ or $\Delta_{r\phi}$)
- Extracting the width from distributions: Gaussian fit in $r\phi$ with wide range, then a refit within 3 sigma, also keep previous method of RMS after iterative 5 RMS cut.
- For distributions with tails and asymmetries observed for example in z we will use the RMS90 with a factor to get back to σ.
- Use the radius (at $\phi = 0°$) instead of row number for plots vs row, to account for different row geometries of the modules.
- Define base units, e.g. mm.
- Always label with "drift distance" or z_{drift} not just z to better distinguish between drift distance and z position.

We defined some naming convention to facilitate the exchange of data sets. A processor called "SomeNameProcessor" writes an aida file identified through the run number: "aida_runxyz.root". The root macro takes a list of run numbers and an output path for the plots and an input location for the aida files:

```
SomeName.C(outputpath,inputpath,run1,run2,run3)
```

and saves the plots as pdf and png file and writes histograms and graphs to a root file "SomeName.root".
To achieve a coherent style of the result plots within our collaboration, a plot style is provided in the analysis folder of MarlinTPC which should be used.

References

[1] Homepage of the LCTPC collaboration, http://www.lctpc.org

[2] P. Schade, J. Kaminski on behalf of the LCTPC collaboration, A large TPC prototype for a linear collider detector, Proceedings of the 12th International Vienna Conference on Instrumentation, Nucl. Inst. & Meth. in Phys. Res. A 628, Issue 1, 1 February 2011, 128-132

[3] DESY test beam information. http://testbeam.desy.de/

[4] T. Matsuda, 2010 JINST 5 P01010, doi:10.1088/1748-0221/5/01/P01010

[5] T. Behnke et al 2013 JINST 8 C10010, doi:10.1088/1748-0221/8/10/C10010

[6] Wenxin Wang, *Etude d'un grand détecteur TPC Micromegas pour l'ILC*, irfu-13-03-T, Universite Paris XI - Paris Sud, UFR Orsay

[7] MARLIN homepage. `http://ilcsoft.desy.de/marlin`

[8] Christoph Rosemann, Executive Summary: Recent Software Developments for Time Projection Chambers, LC-DET-2012-071, 2012,
MarlinTPC homepage. `https://znwiki3.ifh.de/MarlinTPC/`

[9] Rene Brun and Fons Rademakers, ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also `http://root.cern.ch` (last accessed on November 16, 2012)

[10] RAIDA homepage. `http://ilcsoft.desy.de/portal/software_packages/raida/`

[11] GEAR homepage. `http://ilcsoft.desy.de/gear`