EUDET-Report-2008-09

Latest developments in the MarlinTPC software package

Jason Abernathy[*], Klaus Dehmelt[†], Ralf Diener[†], Jan Engels[†], Jim Hunt[‡], Martin Killenberg[§], Thorsten Krautscheid[§], Astrid Münnich[¶], Simone Zimmermann[§], Martin Ummenhofer[§], Adrian Vogel[†], Peter Wienemann[§]

December 3, 2008

Abstract

MarlinTPC is a TPC simulation, digitisation, reconstruction and analysis package developed by the LC-TPC collaboration. It is based on the ICLSoft core software LCIO, Marlin, LCCD and GEAR. Within the last year the functionality has increased a lot, especially in the simulation/digitisation section and in the reconstruction of Timepix data. We report on the current status and the plans, especially concerning the use at the joint test beam at DESY Hamburg, where different modules are to be read out and reconstructed together.

[*]University of Victoria, Victoria, BC, Canada
[†]DESY, Hamburg, Germany
[‡]Cornell University, Ithaca, NY, USA
[§]University of Bonn, Bonn, Germany
[¶]TRIUMF, Vancouver, BC, Canada

1 Introduction

As described in last year's EUDET report[1], the goal of MarlinTPC is to provide a flexible toolkit for TPC research and development, being able to handle different kinds of detector geometries and readout electronics. In the recent years many improved algorithms for TPCs with micro-pattern gaseous readout have been implemented in the software packages of the individual R&D groups. We are aiming at collecting these algorithms and making them available as Marlin[2] processors.

Within the last year the TPCCloudSimulation, which was developed at the University of Victoria, has been implemented as Marlin processors and integrated into MarlinTPC. The single electron digitisation has been extended and now is able to provide realistic event pileup during a bunch train. Both sub-packages will be discussed in detail in this document.

In addition to the hitherto existing reconstruction for TPCs with pad plane readout, the Timepix reconstruction chain used in Freiburg has been implemented in MarlinTPC, its functionality has been extended. This package will also be presented, as well as the various analysis processors which now are available.

2 Simulation and Digitisation

Detector optimisation studies often use Geant4[3] to calculate the energy deposition in the detector and afterwards apply a smearing function to the Geant hit, which implements the detector effects and usually produces one data hit per Geant4 hit. For TPC R&D studies this is not sufficient, as it is part of our research to study the signal shape on the readout pads. So a major requirement of the MarlinTPC digitisation is to provide realistic raw data, i. e. ADC counts on electronics channels. This allows to test the complete reconstruction chain, including the correction of the pad/detector response, compare directly with prototype data and implement effects of drift and diffusion in the gas, which currently also cannot be simulated with Geant4.
Figure 1 shows an overview of the three digitisation branches in MarlinTPC.

2.1 Simple Digitisation

The simple digitisation branch starts from Geant4 hits, which are energy depositions in the gas. The number of primary electrons corresponding to this energy is multiplied with a constant amplification factor emulating the gas amplification.

A TPC is a slow detector because primary electrons drift through the TPC before they reach the readout structure. At the ILC the drift time corresponds to $\mathcal{O}(150)$ bunch crossings. Electrons from several tracks originating from different bunch crossings may reach the readout structure at the same time, which leads to an event pile-up. To simulate this behaviour each charge deposition on a pad is stored in a map of 3D space buckets (voxels). This voxel map is held in memory independently from the Marlin event structure, so signals from multiple events can be stored. In the processor, all data

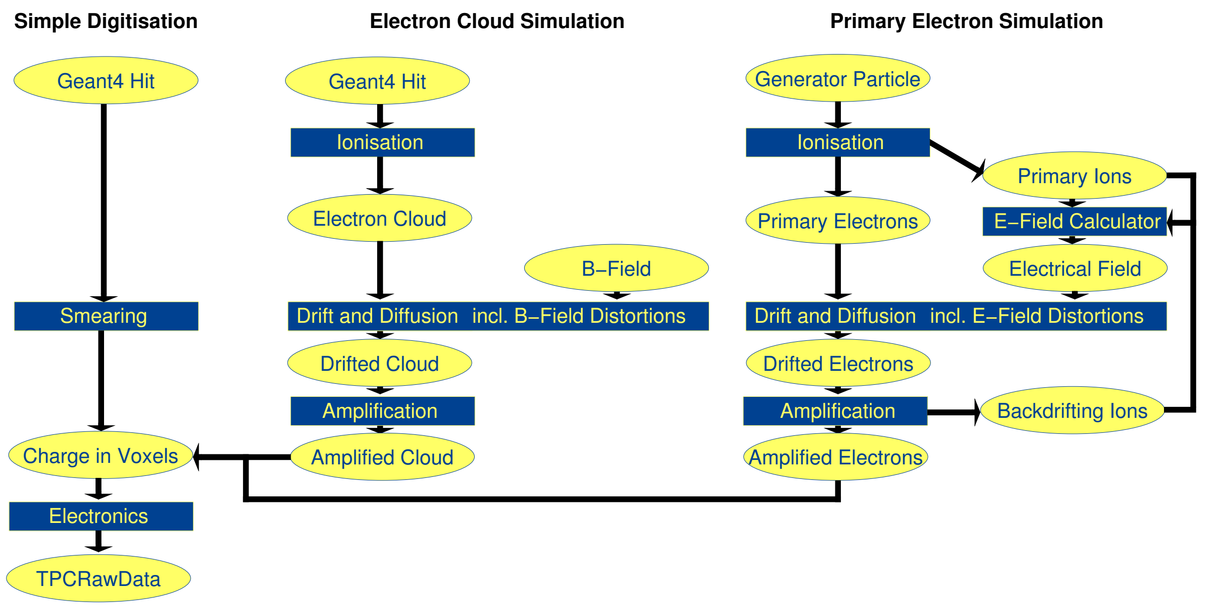


Figure 1: The three digitisation paths of MarlinTPC

corresponding to the drift time of the bunch crossing currently processed are read from the map and written to the output file or send to the next processor. When populating the voxel map the charge is spread using a Gaussian smearing with the width determined by the diffusion in the gas.

In a final step the charge in voxels is processed by the digitisation proper, which calculates raw data depending on the readout electronics. Currently there are implementations for ADCs and the Timepix.

This digitisation path does not take into account any particularities of the gas amplification or the drifting process, except for the simple factors applied. However it still provides quite realistic, electronics dependent raw data.

2.2 TPCCloudSimulation

The TPCCloudSimulation was originally developed at the University of Victoria as part of the JTPC simulation and analysis package[4], which is written in Java. Now the algorithms implemented in JTPC have been ported to C++ to be included in MarlinTPC.

The TPCCloudSimulation deals with electron clouds instead of single electrons to improve simulation performance. Each cloud has a mean position, a transverse and longitudinal deviation and a number of electrons. The simulation is broken into seven processors, details on each are given below. The complete processor chain for a typical TPC setup with two GEMs and three gaseous regions is shown in figure 2.

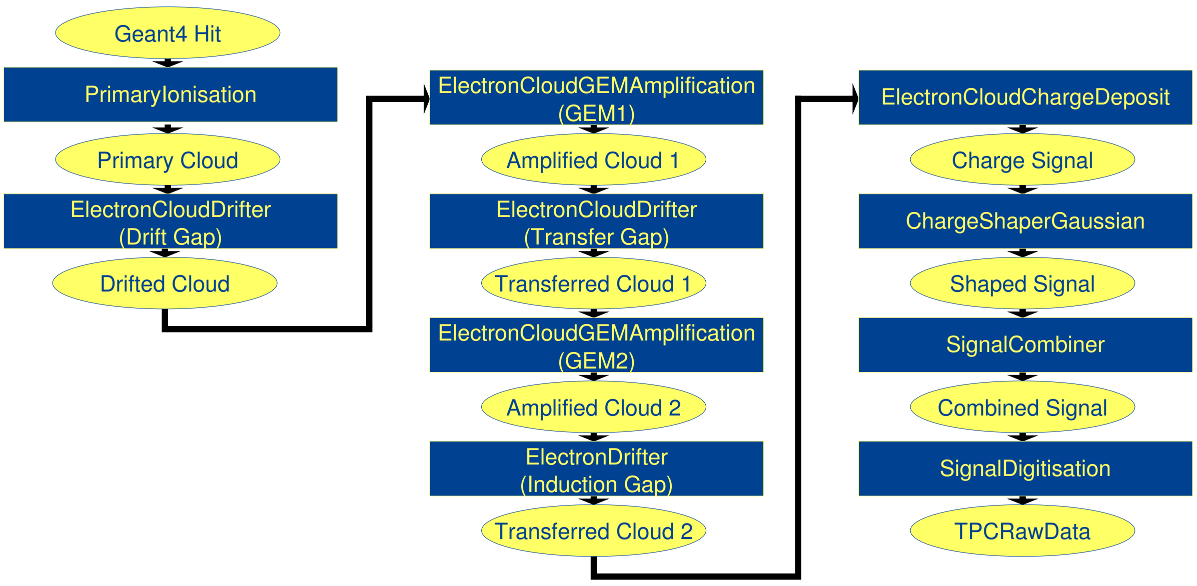


Figure 2: Processor chain of the TPCCloudSimulation with two GEMs.

2.2.1 PrimaryIonisationProcessor

This processor takes the TPC energy deposits produced by Mokka/Geant and converts each one to a cloud of electrons. The conversion is done using the approximate ratio of 26 eV per electron.

2.2.2 ElectronCloudDrifterProcessor

The drifter processor takes each cloud and drifts it through a gaseous section of the TPC. It has a number of input parameters for defining the geometry of the region and the gas. The gas parameters include the drift velocity, diffusion constants and the $\omega\tau$ value. The drifting is done using an adaptive Runge-Kutta-Fehlberg algorithm (RKF45) implemented in the GSL. The differential equation for it was derived using the Langevin method for the drifting of charged particles through gas. It uses a global field manager to query the value of the electric and magnetic fields. While drifting, the standard deviation of the position of the electrons in the cloud is increased to account for drift diffusion.

2.2.3 ElectronCloudGEMAmplificationProcessor

This processor simulates the amplification of a cloud by a Gaseous Electron Multiplier. The GEMs are parameterised by their collection and extraction efficiencies, gain and defocussing. As each cloud is collected by the GEM it is ripped apart into single electrons. Each of these electrons can create a new cloud on the other side of the GEM.

2.2.4 ElectronCloudChargeDepositProcessor

The `ElectronCloudChargeDepositProcessor` takes the collection of clouds that have reached the readout electronics and places the individual electrons onto pads. The pad information is provided by GEAR[5] and stored in a separate file. The charge information is stored as a custom LCIO object called a ChargeSignal. Each signal is parameterised by the number of electrons, mean arrival time and arrival time variance. There is one signal per pad per primary energy deposit. This is to allow signals from different particles to be shaped before they are combined.

2.2.5 ChargeShaperGaussianProcessor

This processor takes each ChargeSignal and smears the charge collected into a Gaussian shaped signal. The output format is TrackerData.

2.2.6 SignalCombinerProcessor

This processor simply loops through the signals and combines multiple signals per pad into one signal.

2.2.7 SignalDigitisationProcessor

The digitisation processor simulates the conversion of the analogue signal into digitised data. It is parameterised by a linear gain parameter, the number of bits per time slice, the frequency of the electronics and a pedestal value. This simple model assumes that there is one channel per readout pad. The final output of the simulation is an LCIO collection of TrackerRawData which can be used for reconstruction purposes. Some processors also add TPC conditions data to the LCIO file on a per event basis.

2.3 Single Electron Simulation

As the Timepix chip is able to resolve individual ionisation clusters (see section 3.2) and even single primary electrons, the Geant4 simulation is not sufficient to provide data on this level of accuracy. So the single electron simulation branch starts off with generator particles and simulates the interaction with the gas in the TPC in a detailed way.

It is based on the TPCGEMSimulation[7] developed in Aachen, which already provided a set of Marlin processors. These have been extended and the current functionality is described in the following sections. The interactions in the gas and also the gas amplification depend on the gas mixture. Currently this detailed simulation is only available for three gas mixtures (Ar/CH_4 95/5, Ar/CH_4 90/10, and Ar/CH_4/CO_2 93/5/2) which have been measured and parameterised.

2.3.1 PrimaryIonisationProcessor

The first processor in this branch simulates the primary ionisation which occurs when a charged particle crosses the TPC. The ionisation clusters are placed along the trajectory of the particle in the TPC. Information on the distance and the size of the electron clusters are known from simulations with HEED[8] and are used in a parameterised form to place the clusters along the track. The track may be straight or curved, depending on the presence of a magnetic field. A detailed treatment of δ-electrons is also included.

2.3.2 DriftProcessor

The `DriftProcessor` simulates the drift of single electrons through the fiducial volume of the TPC. The processor assumes a homogeneous electric field. Drift velocity, transverse and longitudinal diffusion are known for the different gas mixtures and different field strengths from simulations with Magboltz. To determine the diffusion of a drifting electron, a random value is taken from a Gaussian distribution which has the width of the diffusion constant. The position in the x-y-plane is modified by a value taken form the distribution of transverse diffusion and accordingly the time is modified by a random value of the distribution of longitudinal diffusion.

2.3.3 GEMProcessor

This processor simulates the amplification processes in a triple GEM stack. The amplification has been measured and parameterised for different settings of the GEM stack, which are contained within the processor. These parametrisations are combined with binominal statistics to simulate the amplification of electrons which arrive at the GEMs. Binning effects caused by the GEM holes are also taken into account.

2.3.4 ChargeDistributionProcessor

This processor distributes the charges of the amplified electrons onto the pads of the readout plane. The transverse diffusion is known from simulations with Magboltz for different values of the induction field. The charge of the amplification process is distributed by a two-dimensional Gaussian distribution. The processor selects all pads which are at least partly within an area given by a circle with its centre at the position of the amplified charge and a radius of four times the transverse diffusion. The charge on the individual pads is given by the integral of the Gaussian distribution within the limits of the pad area. This algorithm is implemented for rectangular as well as circular layouts of the pad plane. To emulate correct event pile-up the `ChargeDistributionProcessor` also uses the voxel map technique described in section 2.1.

2.3.5 TPCElectronicsProcessor and TimePixDigitisationProcessor

The `TPCElectronicsProcessor` transforms the charges on the individual pads into pulses of the readout electronics. A Gaussian pulse shape is used. Alternatively the

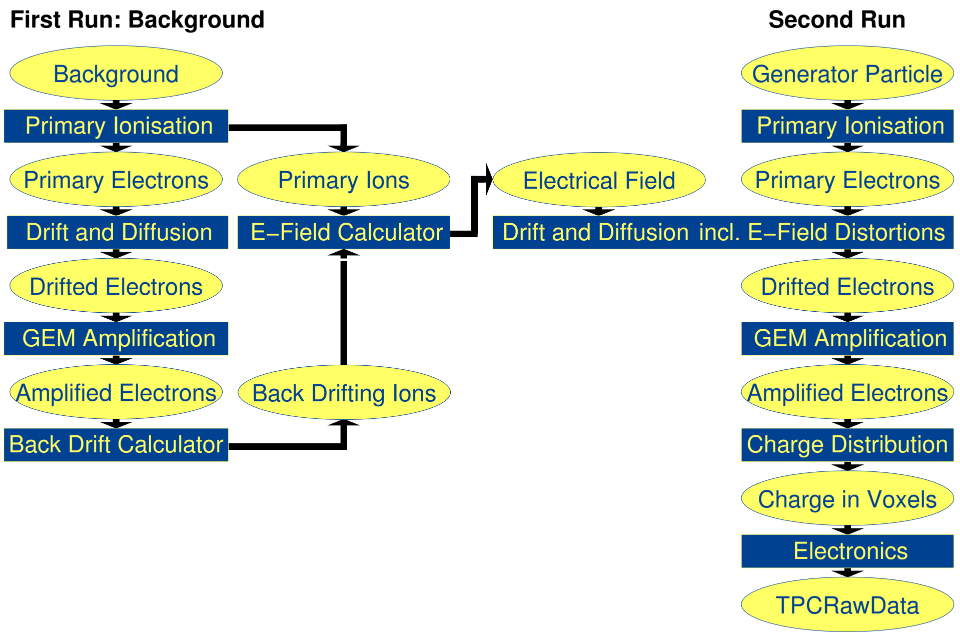


Figure 3: Processor chain for the single electron simulation including ion back drift and field distortions.

`TimePixDigitisationProcessor` can be used to generate raw data resembling the output of the Timepix ASIC.

2.4 Ion Back Drift

The knowledge of the charge transfer within the GEM structure also allows to calculate the number of ions drifting back into the fiducial volume. The drift velocity of the ions is a factor of $10^3 - 10^4$ smaller than the electron drift velocity. In combination with the beam structure at the ILC (1 ms bunch trains with a 199 ms interval between the trains) this leads to the build-up of an ion disc in front of the readout plane. The back drifting ions in the disc as well as the primary ions in the drift volume cause distortions of the electrical field.

From field inhomogeneities caused by primary and back drifting ions a field map can be calculated. This field map can be used together with a processor which takes the inhomogeneities into account when simulating the drift of electrons. To achieve this two simulation runs are necessary. In the first run primary electrons (usually from background events) are drifted through the TPC in a homogenous field, get amplified and are read out. From the primary ions corresponding to these electrons and the back drifting ions from the amplification processes a field map is calculated for each single event. After this a second run with physics events is started. In this run the field inhomogeneities belonging to the given event are taken into account when simulating the drift of the primary electrons. Figure 3 shows the according processor chain.

2.4.1 IonsInVoxelProcessor

This processor bins the ions of the primary ionisation processes into voxels. In the x-y-direction the binning is given by the pads of the readout plane. In z-direction the binning can be chosen by the user.

2.4.2 IonBackDriftProcessor

The `IonBackDriftProcessor` can be used to determine the amount of back drifting ions. The ion back drift has been measured and parameterised for different setups of the GEM stack. With these parametrisations an effective ion back drift coefficient can be determined for a given GEM setting. Multiplication of the charge which ended up on a given pad with the ion back drift coefficient gives the amount of charge which appears in the sensitive volume of the TPC just before this pad. The results are stored in voxels, where the binning in x-y-direction is given by the pads of the readout plane. In z-direction the length of the voxels is given by z-binning of the `IonsInVoxelsProcessor`, however back drifting ions are always stored in the voxels next to the readout plane. In this way the build up of an ion disc can be simulated during a whole bunch train.

2.4.3 EFieldCalculationProcessor

The `EFieldCalculationProcessor` calculates the electric field which results from primary and back drifting ions in the TPC. The processor is used in combination with the `IonsInVoxelsProcessor` and the `IonBackDriftProcessor` which provide the charges of primary and back drifting ions binned in voxels. The TPC is sectioned using a three-dimensional grid. The vector of the electric field which is induced by the ions in the TPC is calculated at each grid point. The spacing of the grid can be chosen by the user. Additional functions allow an interpolation of the electric field for each position between the grid points.

2.4.4 DistortedFieldDriftProcessor

In contrast to the `DriftProcessor` the `DistortedFieldDriftProcessor` takes distortions of the electric field into account when simulating the drift of primary electrons through the active volume of the TPC. The drift of an electron in an electric field can be described by the Langevin equation:

$$\vec{v} = \frac{e}{m}\left|\vec{E}\right|\tau\frac{1}{1+\omega^2\tau^2}\left(\widehat{E} + \omega\tau\left[\widehat{E}\times\widehat{B}\right] + \omega^2\tau^2\left(\widehat{E}\cdot\hat{B}\right)\hat{B}\right)$$

For very inhomogeneous electric fields it is no longer possible to solve this equation analytically. For this reason the processor also uses an RTF45 method for a numeric approximation of the solution. When calculating a Runge-Kutta step the value and direction of the electric field at the given position are used to determine the endpoint of the step. Through the step size control of the integrated method the size of the individual steps depends on the inhomogeneity of the electric field. For inhomogeneous

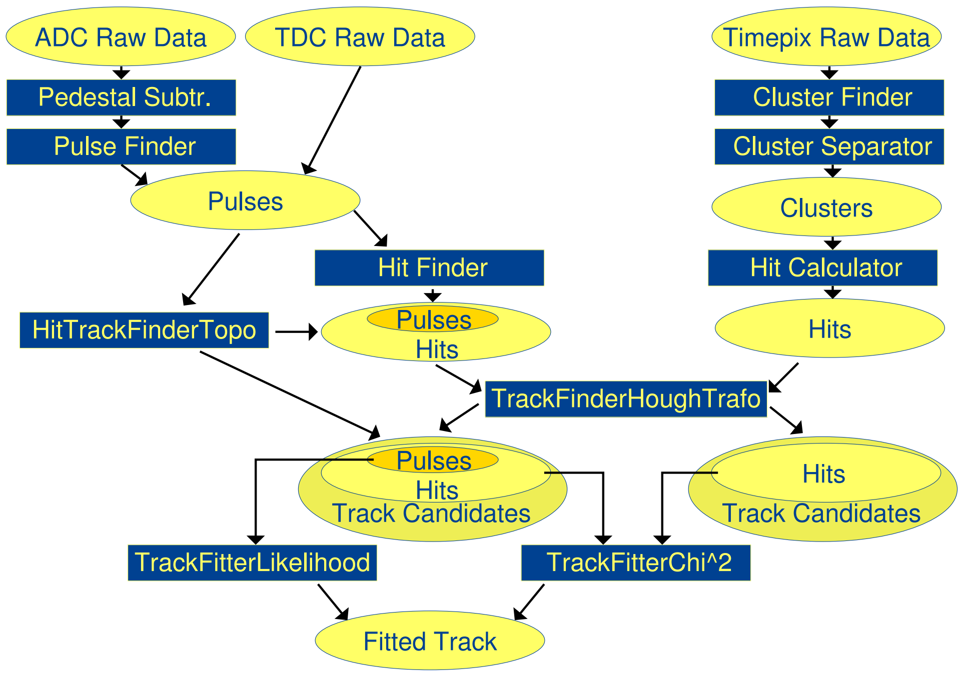


Figure 4: Overview of the reconstruction in MarlinTPC.

regions the steps get smaller whereas in comparatively homogenous regions larger steps are used. The processor no longer calculates the diffusion of the electrons for the whole drift length but takes it into account for each step in dependence on the step size.

3 Reconstruction

The MarlinTPC reconstruction is not only used for the simulated data, but mainly for reconstruction of prototype data in several R&D groups. It has to be able to deal with pad planes read out by ADCs or TDCs, and with data recorded with the Timepix ASIC. Figure 4 gives an overview of the different reconstruction paths in MarlinTPC.

To describe the different geometries, MarlinTPC uses the GEAR API. For the TPC, GEAR provides an abstract interface for a pad plane where the pads are arranged in rows. Currently there are three implementations for a pad plane with rectangular pads in a cartesian coordinate system, keystone shaped pads all having the same width in a polar coordinate system, and keystone pads all having the same angle. As all code in MarlinTPC is programmed against the abstract interface, it generically runs with all of these pad geometries and with any further implementations to come, without having to adapt the algorithm.

3.1 Pad Reconstruction

The main pad reconstruction chain starts with ADC counts (`lcio::TrackerRawData`) which are recorded on each channel. In case the zero suppression and pedestal subtraction have not been performed on the front-end electronics, this is the first step in the reconstruction chain.

3.1.1 PulseFinderProcessor

The next step is to calculate pulses (`lcio::TrackerPulse`) on the pads, which contain only information of the integrated charge and the time when the signal arrived, plus a link to the raw information the pulse was calculated from. In addition there are some quality flags, for instance unusual pulse shapes or double-hit candidates.

3.1.2 Time to Digital Converters

The raw data from time-to-digital converters (TDCs) are directly translated into pulses. The arrival time of the pulse is the first recorded value, the length of the pulse, which is proportional to the charge, is the second one. They directly translate into time and charge of the pulse. No further information about the pulse shape is available. This processor is currently being implemented.

3.1.3 HitFinderProcessor and hit based track finding

Hit finding can be performed in two different ways. The `HitFinderProcessor` searches for charge distributions on neighbouring pads in each row and calculates 3D hits (`lcio::TrackerHit`), each having the charge in the hit and a 3D space coordinate as variables. Afterwards track finding is done using the hits. The current implementation of the `TrackFinderHoughTrafoProcessor`, which is avaliable for hit based track finding is limited to straight tracks. It is described in detail in the Timepix reconstruction section (see 3.2.4).

3.1.4 HitTrackFinderTopoProcessor

The combined topological hit and track finder `HitTrackFinderTopoProcessor` searches for contiguous areas in the 2D pad plane, plus a cut in the time direction which allows a real three-dimensional separation of the tracks. Afterwards the charge in each row is again grouped to a hit, but these hits are already associated to a track candidate. This method does not use a track hypothesis and thus works for straight lines as well as for curved tracks and even for low energetic particles which curl up in the magnetic field and for tracks with kinks.

3.1.5 TrackFitterLikelihoodProcessor

To get an optimal fit result for the track parameters, the position of the charge with respect to the pads has to be taken into account and corrected according to the pad response function. As the width of the charge deposition depends on the track angle with respect to the pad row, this angle influences the pad response. As the track angle is not known before performing the fit, this leads to a deadlock. One way to solve this

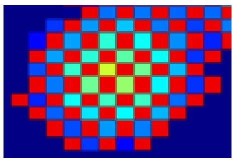


Figure 5: One cluster recorded with a Timepix and a triple GEM gas amplification structure. One can clearly see the *mixed mode* behaviour: red pixels correspond to `Time` values, while the colour range from blue to yellow describes the charge which has been recorded in `TOT` mode.

problem is to do a global likelihood fit which includes the pad response. For each row the likelihood for a track with given parameters to result in the measured charge distribution is calculated. Afterwards the log(likelihood) is summed up over all pad rows to obtain the global likelihood. Finally the track parameters are varied and the parameter set which yields the maximum likelihood is determined. This algorithm has been shown to give excellent spatial and momentum resolution for the fitted tracks[9]. The current version of the fitter can use the electron cloud drifting algorithm (see section 2.2) to calculate the likelihood, so field inhomogeneities in the drift region are automatically corrected for.

3.2 The Timepix Reconstruction Chain

The Timepix ASIC is a silicon readout chip with 55×55 µm^2 pixels.[10] It originally was designed to be operated with a silicon sensor, every pixel of the Timepix being bump-bonded to the active material. For gaseous detectors the bare Timepix without silicon sensor is used. The charge generated in GEMs or Micromegas is collected on the bond pads.

Each pixel can either be operated in `Time` mode, counting the clock cycles to the end of a time windows of defined width (shutter window), or in `Time Over Threshold` mode (`TOT`). In `TOT` mode the number of clock cycles in which the signal is above an adjustable threshold is recorded. The analogue preamplifier on each pixel is designed to have a linear decay time, so the `TOT` signal is proportional to the collected charge.

In GEM amplification structures, the charge of a primary ionisation cluster is spread over $\mathcal{O}(100)$ pixels. This allows to operate the chip in a chequerboard pattern, the pixels alternately recording `Time` or `TOT`. Figure 5 shows an example cluster recorded in this *mixed mode*.

3.2.1 TimePixClusterFinderProcessor

The first step in the cluster finding process is to group adjacent pixels to clusters (see figure 6 for illustration). An additional cut is applied on the minimum number of pixels a cluster needs to consist of in order to be stored. This size is a processor parameter and can be set by the user.

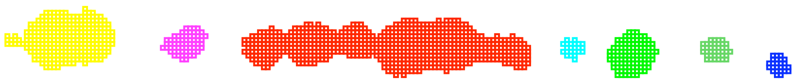


Figure 6: Example event: Each colour represents one of the seven cluster candidates.

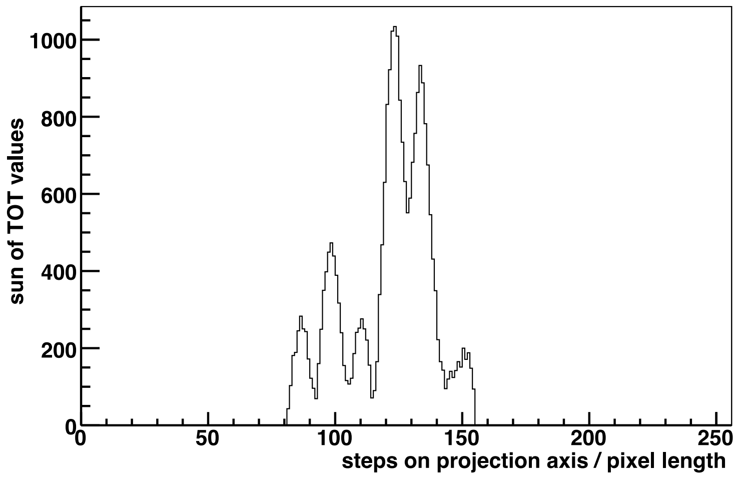


Figure 7: The projection of a single cluster onto its own axis. In this example it is the large central cluster (red) in figure 6.

3.2.2 TimePixClusterProjectionSeparatorProcessor

Due to the large charge spreading in the GEM structure the primary clusters merge. The large non-circular clusters need to be separated into smaller, nearly circular ones. To split the clusters, MarlinTPC implements the *projection* or *saddle point method*. In a first step the cluster's axis is determined by linear regression. Afterwards, all pixels in a cluster are projected onto this cluster axis, weighted by their charge information (figure 7). In this projection, the algorithm searches for local minima followed by a local maximum and cuts the cluster perpendicular to the axis at each minimum. Figure 8 shows how the clusters have been split.

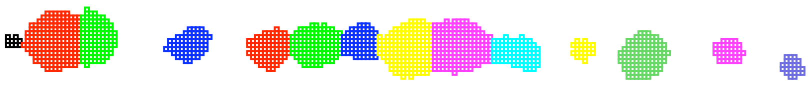


Figure 8: The clusters of the example event after separation.

3.2.3 TimePixHitCenterCalculatorProcessor

To gain the two-dimensional information about the cluster's position, the charge centroid for each cluster is calculated. Before storing the information these coordinates are transformed to millimetres with respect to the origin on the chip.

The third coordinate is determined from the mean `Time` value of the cluster, which is multiplied with the drift velocity and the readout frequency. For the time being the latter are read in as processor parameters, future plans, however, incorporate them in a database for conditions data.

Furthermore, the charge value of the cluster is needed for the `lcio::TrackerHit`. So far, only the sum of all `TOT` values is stored, since a correct conversion of `TOT` values to units of the elementary charge has not been calibrated. In contrast to the `Time` values, the overall charge of the cluster needs to be interpolated to the pixels run in `Time` mode. Hence, the formula to calculate the "charge value" of the cluster is

$$Q = \frac{N_{\text{pixel}}}{N_{\text{TOT}}} \cdot \sum_i q_i. \tag{1}$$

In the above formula q_i denotes the single `TOT` values and N_{pixel} and N_{TOT} stand for the total number of pixels and pixels run in `TOT` mode, respectively.

With the calculation of the charge and 3D space information of the hits the Timepix specific part of the reconstruction chain ends. All following processors can also be used on hits calculated from pad data.

3.2.4 TrackFinderHoughTrafoProcessor

In order to assign the clusters/hits to tracks a *Hough transform* is performed. It is implemented as a histogram method that can serve to group points on a track together. Considering a set of N points in the x-y-plane, the so called *image space*, all points on a straight track will follow

$$y = a \cdot x + b. \tag{2}$$

In the case that a given point (x_i, y_i) lies on the track,

$$y_i = a \cdot x_i + b, \tag{3}$$

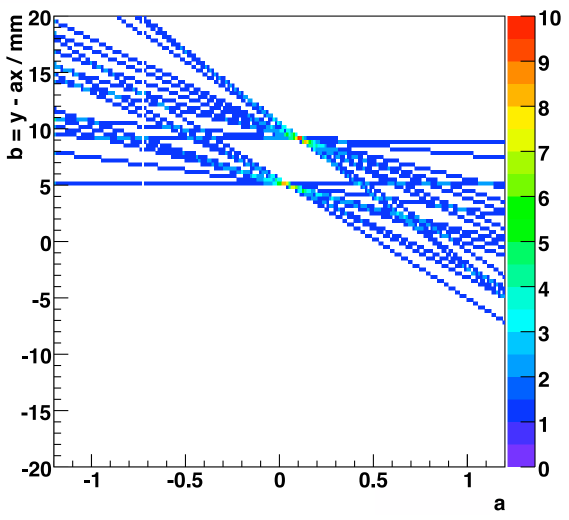
will hold. In this case

$$b = -x_i \cdot a + y_i \tag{4}$$

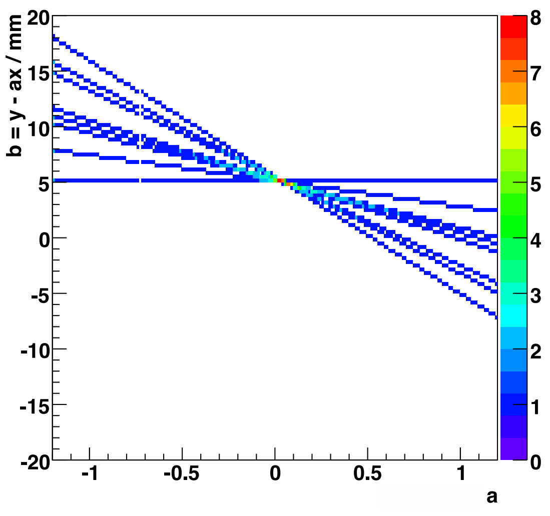
will also be a valid expression, which is a straight line in the a-b-plane (*parameter space*) with slope $-x_i$ and intercept y_i.

All collinear points in image space will correspond to lines in parameter space intercepting at a single point. This point gives the parameters of the line in image space that best represents the collinear points.

In practice the lines in parameter space are filled into histograms. If a bin of the histogram is crossed by multiple parameter space lines, it will have as many entries as



(a) Two track event in parameter space.



(b) Two track event in parameter space after removing all points of the first track.

Figure 9: Histograms in parameter space of a two track event.

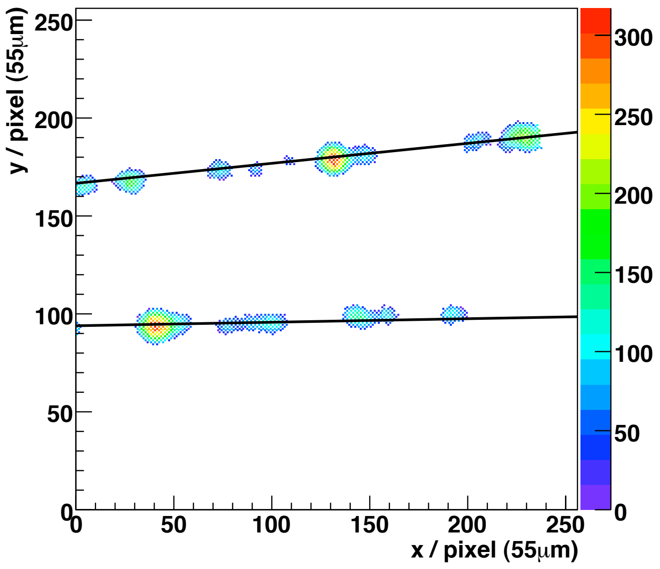


Figure 10: The `TOT` values of a Timepix event with two tracks and the tracks found by the Hough transform.

lines crossing it. By looking for the bin with the maximum amount of entries, the point of interception can be determined.

Every time a maximum in the histogram is found, its (a, b)-pair is stored and all points in image space lying in a vicinity of the track determined by (a, b) are removed from the histogram. This is repeated until the histogram is empty (figure 9). In this manner, the algorithm is able to find multiple tracks (figure 10).

Processor Name	Detector Specific	Multi z-bin
`BiasedResidualsProcessor`	no	no
`CutApplicationProcessor`	no	no
`HitAndTrackChargeProcessor`	no	yes
`HitAndTrackCounterProcessor`	no	yes
`LinearGeometricMeanResolutionProcessor`	no	yes
`LinearThreePointResolutionProcessor`	no	yes
`TimePixClusterSizeProcessor`	yes (TimePix)	yes
`TimePixOccupancyProcessor`	yes (TimePix)	no
`TimePixTOTDistributionProcessor`	yes (TimePix)	no
`TrackParametersDistributionProcessor`	no	no
`XYZDistributionProcessor`	no	no
`XYZDistributionTracksProcessor`	no	no

Table 1: List of the analysis processors available in MarlinTPC

3.2.5 LinearRegressionProcessor

The a and b values found in this way are a good starting value for further fitting, but have rather large uncertainties due to binning effects. To achieve better results, the track parameters are recalculated from the hits assigned by the Hough transform via linear regression. Additionally, the track parameters (c, d) in the third dimension are calculated in either the x-z-plane or the y-z-plane depending on the orientation of the track.

Many of the aforementioned algorithms need information not contained in the actual Timepix raw data, like for example the information in which recording mode the pixel had been run, the readout frequency of the chip or the shutter time. All of this is stored in *conditions data* along with all other experimental parameters. On the long run this will be done via a database with easy and user friendly access to all needed information. Until this database has been setup, these data are read in as processor parameters or are stored in separate files. For the information on the single pixels, a `TimePixModeMap` has been designed, which allows access to the mode of a pixel and its state (dead, noisy, intact). Chip imperfections in the form of dead or noisy pixels are accounted for by an interpolation method. For every dead or noisy pixel, the ADC counts are recalculated from the average of the surrounding pixels that recorded in the same mode.

4 Analysis

MarlinTPC provides ready-to use analysis processors which create histograms using the AIDA interface. Depending on whether Marlin has been linked against RAIDA or JAIDA/AidaJNI, the created histogram files are root files or XML files.

Table 1 shows a list of the currently available analysis processors. Most of them are geometry and readout independent, implementing some of the default analyses recommended at the first ILC TPC Analysis Jamboree.[11] It is planned to implement all the analyses which have been proposed. In addition there are detector specific processors for commissioning the Timepix chip. As in the reconstruction the processor names have been prefixed with “TimePix”.

Some of the processors have been implemented to divide the drift volume into several z-slices to study the dependence on the drift distance. For convenience they are derived from the `ZBinTemplateProcessor`, which provides all data structures and loops to generate and fill one histogram for each z-bin plus an additional histogram which integrates over the complete TPC.
The following sections briefly describe each processor.

4.0.6 BiasedResidualsProcessor

The `BiasedResidualsProcessor` generates a histogram of the residuals of the hits on a track with respect to the track fitted to the data. This is known to bias the width of the residual distribution towards smaller values, since the test hit was included in the track fit. This processor is especially useful for tracks with a large number of hits, as it is fast because it does not require numerical fitting, and for a large number of hits the bias is small.

4.0.7 CutApplicationProcessor

The `CutApplicationProcessor` allows to apply upper and lower cuts on the number of tracks per event and on the main track and hit parameters.
For the track these are

- number of hits on the track
- track angles, offsets and curvature
- $\mathrm{d}E/\mathrm{d}x$

The hit parameters can be cut on

- the charge
- the x, y and z coordinate

In case a hit does not pass all cuts, the complete track is rejected.

4.0.8 HitAndTrackChargeProcessor

The `HitAndTrackChargeProcessor` generates histograms of the charge per hit (all reconstructed hits and hits on tracks only) and of the charge per track length ($\mathrm{d}E/\mathrm{d}x$).

4.0.9 HitAndTrackCounterProcessor

The `HitAndTrackCounterProcessor` counts hits in events and tracks, and tracks per event and fills the respective histograms.

4.0.10 LinearGeometricMeanResolutionProcessor

As described above, calculating the residuals to a track fitted to all hits (n-fit), the width of the residual distribution is biased towards too small values. Excluding the test hit from the track fit ($n-1$-fit), however, gives too pessimistic values. It has been shown that the geometric mean of the two widths

$$\sigma_{\text{geo}} = \sqrt{\sigma_n \cdot \sigma_{n-1}}$$

is an unbiased estimator.[12] Obviously this algorithm needs refitting of the tracks for each hit, which makes it slow. The `LinearGeometricMeanResolutionProcessor` is an implementation for straight tracks, using linear regression.

In this approach every fit algorithm requires a reimplementation of the code in the track fitter processor, and some algorithms like the global likelihood fit require a special method to determine the residuals. To overcome this problem it is planned to have an abstract `TrackFitterBase` interface, which provides access to the fitting algorithm and the residual calculation. This code can be accessed from the track fitter and the geometric mean processor. A factory automatically provides the correct fitter implementation in the `GeometricMeanProcessor`, so this code is independent from the actual fitter and does not have to be changed, even if new fitting algorithms are introduced.

Like the other resolution processors, the geometric mean algorithm has been tested with toy Monte Carlo data.[13] Figure 11 shows the n-fit and $n-1$-fit distributions for hits distributed with a Gaussian width of 50 µm around the track.

4.0.11 LinearThreePointResolutionProcessor

The three point distance is calculated for a group of three neighbouring hits on the track as the distance of the middle hit to a straight line through the other two hits. Performing this calculation for all groups of three neighbouring hits results in $n-2$ distances for a track with n hits. The three point distances are histogrammed for all tracks and the resolution is calculated from the width of this distribution as

$$\sigma = \sqrt{\frac{2}{3}}\sigma_{\text{3point}}$$

This corresponds to the width of a Gaussian distribution around the real track position for equidistant hits.

4.0.12 TimePixClusterSizeProcessor

The `TimePixClusterSizeProcessor` generates histograms with the number of pixels per hit and the cluster radius.

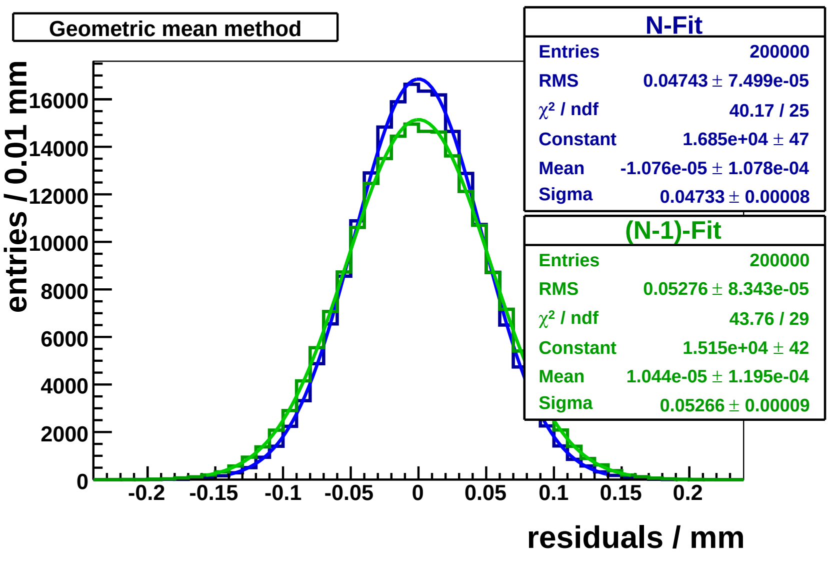


Figure 11: Monte Carlo test of the `LinearGeometricMeanResolutionProcessor`: The geometric mean of the two RMS values yields 50.02 ± 0.11 µm, the Monte Carlo truth value is 50 µm.

4.0.13 TimePixOccupancyProcessor

The `TimePixOccupancyProcessor` generates a 2D histogram with 256×256 bins, representing the pixels on the chip. For each bin it is stored how many times the pixel has been hit in the run.

4.0.14 TimePixTOTDistributionProcessor

The `TimeOverThreshold` value of all pixels running in this mode is histogrammed by this processor.

4.0.15 TrackParametersDistributionProcessor

The `TrackParametersDistributionProcessor` generates histograms of the track parameters ϕ, λ, $\tan(\lambda)$, d_0, and z_0.

4.0.16 XYZDistributionProcessor and XYZDistributionTracksProcessor

Both processors generate one-dimensional histograms of the x, y and z coordinates of the reconstructed hits. The `XYZDistributionProcessor` does it for all hits in the event, while the `XYZDistributionTracksProcessor` only uses the hits which have been assigned to a track.

5 Outlook

For the upcoming joint test beam with the large TPC prototype at DESY, MarlinTPC is planned to be the default reconstruction framework. To have a common readout scheme all DAQ systems will use EUDAQ[14]. In this framework each DAQ system is set up as a *data producer*, implementing a TPC/IP protocol to ship the data. In addition, the data producer can be controlled by a common run control which starts and stops all data producers, initialises a new run etc. One PC will be set up as the EUDAQ data collector. It collects the data from the individual DAQ subsystems and combines them to one common LCIO data stream. On this PC MarlinTPC will be installed and available to analyse the data. It is planned to transfer the data to a grid storage element to make them available from everywhere and to have a backup. Not only the different TPC readout setups (Altro ADCs, After ADCs, TDCs and Timepix) but also the silicon strip hodoscope, which will be installed outside the field cage, is planned to be integrated into this scheme.

In order to reconstruct the data from several readout modules, the GEAR description had to be extended.[16] Up to now only end plates consisting of one monolithic pad plane could be described. The new GEAR version can now handle several modules which are either rectangles or sectors of a circle. As the new description is fully backward compatible with the previous version, the existing code runs with the new geometry without modifications. In a next step MarlinTPC has to be adapted to take full advantage of the multi-module end plate and for instance perform track fits across several modules.

The functionality of each processor has been well documented in the source code, which allows to generate a browsable HTML documentation using Doxygen. But especially for beginners this is not sufficient to get an overview of how to get started. So it is planned to improve the tutorial, which is available on the MarlinTPC Wiki pages[15]. Also the examples will be revised and extended to facilitate the first steps. In addition it is planned to have tools which help to generate the conditions data which are needed, for instance the pad mapping.

6 Conclusions

In the last year the functionality of MarlinTPC hac increased a lot. It now contains more than 50 processors for simulation, digitisation, reconstruction, and analysis, which makes it a flexible toolbox for a large variety of applications.

The electron cloud digitisation enables the creation of realistic data on the pads, including distortions due to inhomogeneous magnetic fields in the drift region. It is used in the reconstruction by the likelihood fitter, which in this vein automatically takes the field distortions into account. The single electron digitisation produces raw data which is realistic enough for the Timepix, which is able to resolve individual ionisation clusters and even single primary electrons. The ion back drift processor allows to calculate the ions drifting back from the readout plane into the fiducial volume, forming an ion disc in front of the readout. The electrical field distortions caused by this charge build-up can be calculated and the effect on the electron drift is taken into account during simulation.

The reconstruction section has been extended by a complete branch for the Timepix readout chip, so now a pad based and a pixelated TPC can be used. For the Timepix MarlinTPC is used in Bonn and Amsterdam to reconstruct test beam data, for the pad reconstruction DESY is using it. MarlinTPC is planned to be the default reconstruction tool for the joint Large Prototype test beam at DESY.

Acknowledgement

This work is supported by the Commission of the European Communities under the 6th Framework Programme "Structuring the European Research Area", contract number RII3-026126.

References

[1] J. Abernathy et. al, “MarlinTPC: A Marlin based common TPC software framework for the LC-TPC collaboration”, EUDET-Report-2007-04, 2007

[2] F. Gaede, Marlin and LCCD: Software tools for the ILC, Nucl. Instrum. Meth. A 559 (2006) 177.

[3] S. Agostinelli *et al.* [GEANT4 Collaboration], “GEANT4: A simulation toolkit”, Nucl. Instrum. Meth. A**506** (2003) 250

[4] D. Karlen, “jtpc simulation and analysis software,” available at `http://particle.phys.uvic.ca/~karlen/jtpc`

[5] The GEAR home page: `http://ilcsoft.desy.de/gear/`

[6] F. Gaede, T. Behnke, N. Graf, T. Johnson, “LCIO - A persistency framework for linear collider simulation studies”, CHEP03, arXiv:physics/0306114, 2003

[7] A. Münnich, “Simulation Studies for a Time Projection Chamber at the ILC”, Vdm Verlag Dr. Müller (2008)

[8] I. B. Smirnov, “Modeling of ionization produced by fast charged particles in gases”, Nucl. Instrum. Meth. A **554** (2005) 474

[9] D. Karlen, P. Poffenberger, G. Rosenbaum, “TPC performance in magnetic fields with GEM and pad readout”, Nucl. Instrum. Meth. A **555** (2005) 80-92

[10] X. Llopart, et al., “Timepix, a 65k programmable pixel readout chip for arrival time, energy and/or photon counting measurements”, Nucl. Instrum. Meth. A **581** (2007) 485-494

[11] Peter Wienemann, “Recommendations from the first TPC Analysis Jamboree 2006”, https://twiki.cern.ch/twiki/bin/view/ILCTPC/JamboreeRecommendations, 2006

[12] R. K. Carnegie, M. S. Dixit, J. Dubeau, D. Karlen, J. P. Martin, H. Mes, K. Sachs, “Resolution studies of cosmic-ray tracks in a TPC with GEM readout”. Nucl. Instrum. Meth. A **538** (2004) 372

[13] Simone Zimmermann, “Data Reconstruction and Analysis of GEM-Based Time Projection Chambers with Pixel Readout”, diploma thesis, Bonn, 2007

[14] D. Haas, “The DAQ for the EUDET pixel telescope”, EUDET-Report-2007-08, 2007

[15] The MarlinTPC wiki page: https://twiki.cern.ch/twiki/bin/view/ILCTPC/MarlinTPC

[16] M. Killenberg and S. Turnbull, “A Modular TPC Endplate Description for GEAR”, EUDET-Memo-2008-31, 2008