MarlinTPC Simulation and Digitisation
Current Developments for my CLIC Detector Study

Martin Killenberg

MarlinTPC EVO Meeting

27. May 2010

Input for the Digitisation

Geant4 / Mokka

- `SimTrackerHit` with the deposited energy in GeV
- One primary electron corresponds to 26 eV

PrimaryIonisationProcessor

- One `SimTrackerHit` per primary electron, even if they are created at the same space point
- Energy information in `SimTrackerHit` was not used

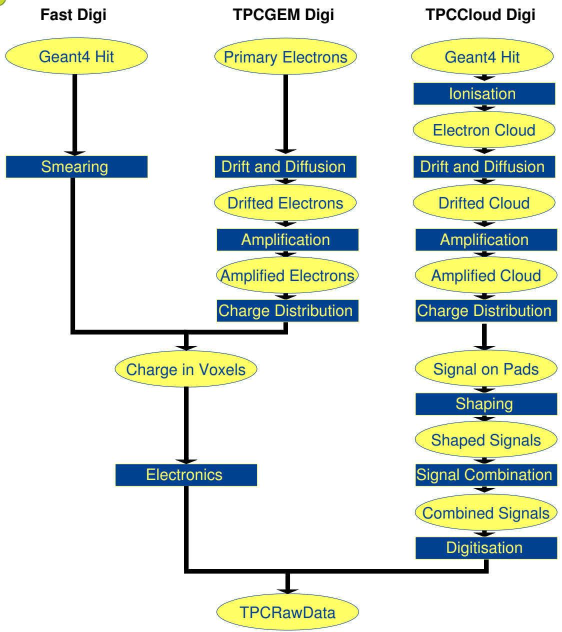
New:

- Energy is 26 eV per electron
- Electrons at the same space point can be grouped to one hit (optional)

$\Rightarrow$ Output compatible with Mokka output

– TPCGEM digi currently still expects one hit per electron (to be fixed)

Digitisation Overview



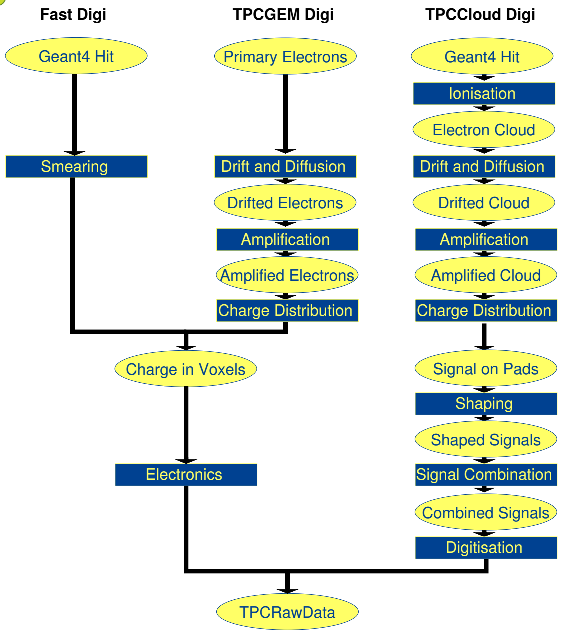
Fast Digi

- Only two steps from input to raw data
- Map of voxels (3D space buckets) implements pile-up
  - of several hits in the event
  - of events in a bunch train (not implemented yet)
  - of background (not implemented yet)

TPCGEM Digi

- Track every single electron
- Also uses the voxel map and the same electronics than Fast Digi

Digitisation Overview



TPCCloud Digi

- Uses clouds instead of single electrons (faster)
- Does not use a global voxel map
- Used in Likelihood fitter

Remarks:

- TPCGEM Digi and TPCCloud Digi are very similar ⇒ Can we use parts of the same code in both?
- In the following I will only show the fast branch which I use in my study

MokkaToVoxelProcessor

The voxel map:

- Segmentation in $xy/r\varphi$: One column of voxels per pad
- Segmentation in z: 3 voxels per ADC time sample (processor parameter)

The steps in this processor:

- Calculate number of primary electrons from Mokka hit (26 eV per e^-)
- Apply gas gain (currently without fluctuations)
- Distribute charge according to diffusion: `ChargeDistributor` helper class from TPCGEM Digi
  - Calculate number of electrons per pad with Gaussian distribution (σ_{trans})
  - Calculate number of electrons per time bin with Gaussian distribution (σ_{long})

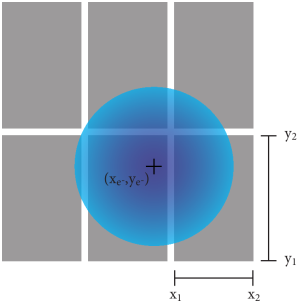


Image by Thorsten Krautscheid

Changes[1] in MokkaToVoxelProcessor

Position of readout anode (`zAnode`)

- In Mokka: 5 mm air gap at z=0
- Drift distance alone is not sufficient information
- `zAnode` currently user parameter in GEAR file

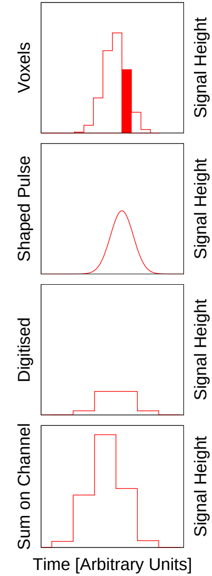
Two half TPCs

- Current solution (quick hack):
  Have two voxel maps and write two collection
- – All following processors have to be called twice

- Correct solution: Use different modules for each end cap
  - Voxel map has to be extended with module index
    (automatically introduced multi module capability per end plate)
  - Each module has to have a z position (extend GEAR?)
  - GEAR multi module information has to be merged into XML file from MOKKA

[1] not in the trunk yet

TPCElectronicsProcessor



The TPCElectronicsProcessor simulates *n* bit FADC
(e. g. ALTRO readout with 10 bit 40 MHz ADCs)

Old version:

- Calculate CoG of the voxel signal
- Apply Gaussian shaping around CoG and digitise
- Adjustable cutoff of trailing edge (asymmetric shape)
- Electronics threshold
- − Bad performance in case of overlapping pulses

New version:

- For the charge **in each voxel** a Gaussian shaping is applied
- The shaped signal is digitised
- The digitised signals for one channel are summed up
- + Realistic signal for overlapping pulses
- − No asymmetry and threshold yet

AddEmptyPedestalsProcessor

- The current digitisation does not introduce pedestals and noise
- Reconstruction needs noise levels

$\Rightarrow$ Helper processor which just writes 0 to both pedestal and pedestal width

Remarks:

- The empty pedestals collection uses 500 MB of memory (for an LDC type end plate)
- Pedestal subtraction is usually handled already on front end electronics
- Do we need an individual noise level for each channel?
  If not, we could have a reco which does not require the huge collection (but can use it if provided)