Status of Silicon Tracker digitization and hit reconstruction in MarlinReco [1]

Sergey Shulga [2], Tatiana Ilicheva [3]

JINR, Dubna, Russia

GSU, Gomel, Belarus

ILC Software and Tools Workshop

2 - 4 May 2007

LAL-Orsay

[1] This work is supported by BMBF(Germany)

[2] shulga@mail.desy.de, [3] ilicheva@mail.desy.de

Contents

What we had till now

- *FTDDigiProcessor* and *VXDDigiProcessor* use simple Gaussian smearing to produce reconstructed *TrackerHit* from *SimTrackerHit* **(S.Aplin)**
- *VTXDigitizer* **(A.Raspereza):** Input is *SimTrackerHit* and output is *TrackerHit;* physics in silicon is included (used CMS algoroitms), pixel mode
- *SiTrkDigiProcessor* and *SiTrkHitProcessor* **(S.Shulga):**

  SiTrkDigiProcessor transforms *SimTrackerHit* collection to *TrackerRawData.*

  SiTrkHitProcessor transforms *TrackerRawData* to *TrackerHit;*

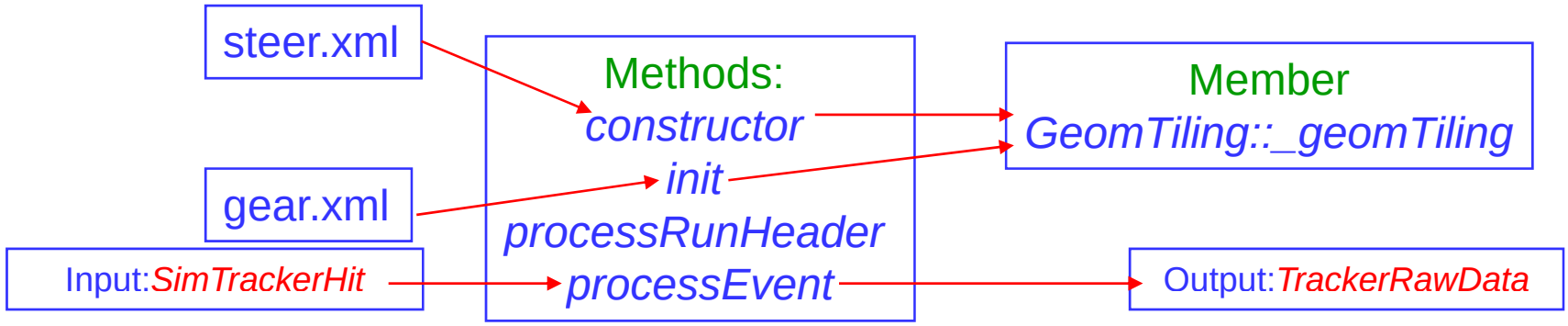
  only geometrical digitization and hit reconstruction,

  pixel and strip mode for barrel and FTD disks

The report describes new digitization and clustering processors within the framework of MarlinReco

with use of algorithms developed in CMS (ORCA, COBRA).

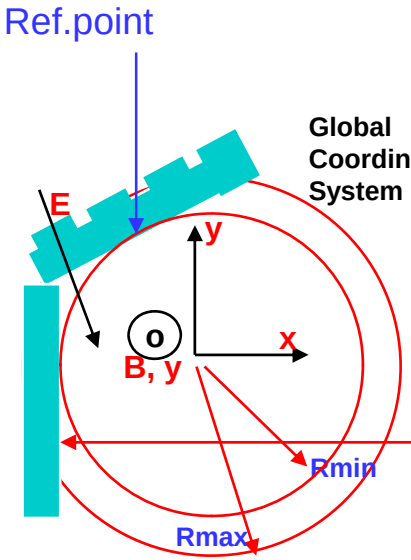
Class *SiTrkDigiProcessor*

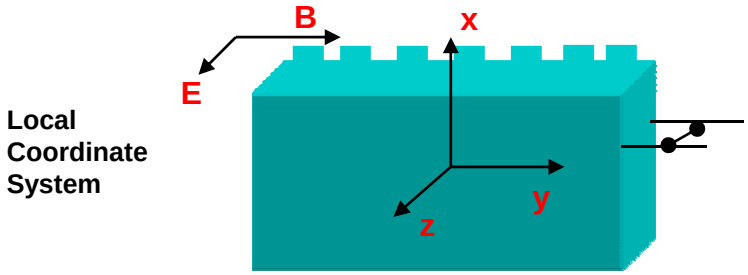


{Rmin,Rmax,Zmin,Zmax,NLadders} – complete information to define det.units for barrel

The objects to be digitized is "Detector Unit".
GeomTiling prepares vector of vectors of *DetUnit*'s:
first vector is over layer second vector is over *DetUnit in layer.*

DetUnit is rectangular thin parallelepiped in local coordinate system.





DU contain information about pixel XY-sizes and thickness

BarrelDetUnit, ForwardDetUnit are derived classes completely defined by type of DU (barrel or forward disk) and 2-dimensional reference point in global coordinate system.

Class *GeomTiling*

Main members

- *vector< vector< DetUnit * > > _detUnitVecVec*

- *SiPixelDetUnitDigitizer _pdigitizer*
- *SiPixelDetUnitClusterizer _pclusterizer*

- *SiStripDetUnitDigitizer _sdigitizer* to be done
- *SiStripDetUnitClusterizer _sclusterizer* to be done

Main methods

- *gearInit*
- *gearInitVXD* (called by *gearInit)*
- *gearInitFTD, gearInitSIT* to be done

- *fillDetUnits(SimTrackerHit *);*
- *digitizeDetUnits(vector<LCCollectionVec *> &)*
- *fillDetUnits(TrackerRawData *)*
- *clusterizeDetUnits(vector<LCCollectionVec *> &)*
- *clearDetUnits()*
- *howManySimHits()*
- *howManyRawHits()*

Abstract base class *DetUnit*

Main members

- *Hep3Vector _BFieldLocal ;*
- *vector< SiTrkSimHitExtended *> _simHits ;*
- *SiTrkSimHitExtended _simHit_tmp ;*
- *Hep3Vector _theDriftDirection ;*
- *map<int channel, class Amplitude> _signal ;*
- *vector< IMPL::TrackerRawDataImpl *> _rawHits ;*
- *int _electronPerADC;*
- *GaussianTileNoiseGenerator _theNoiser ; // From // ORCA*
- *double _noiseInElectrons,* // Noise (RMS)
  _noiseInAdcCounts,
  _pixelThresholdInNoiseUnits,
  _pixelThresholdInElectrons,
  _seedThresholdInNoiseUnits,
  _clusterThresholdInNoiseUnits;
- *vector<double> _thePixelEfficiency,*
  _thePixelColEfficiency,
  _thePixelChipEfficiency ;
- *int _rocSizeInX,* //in pixels, nb of row's pixels per ROC
  _rocSizeInY ; // --//-- column's pixels per ROC

Main methods

- *hitEntryExit(LCIO::IMPL::SimTrackerHit * hit)*
- *setNLayer()*
- *setDetUnitNumber()*
- *double coordX(unsigned int row)*
- *double coordY(unsigned int column)*
- *unsigned int NpixelX(double X)*
- *unsigned int NpixelY(double Y)*
- *vector< SiTrkSimHitExtended *> getSimHits()*
- *vector< IMPL::TrackerRawDataImpl*> getRawHits()*
- *Hep3Vector driftDirection()*
- *void addSignalFromSimHit(map<int,double> hit_signal), SiTrkSimHitExtended * hit)*
- *void add_noise()*
- *void pixel_inefficiency()*
- *void make_digis(LCCollectionVec *rawVec, LCCollectionVec *relVec)*

Why we need class *SiTrkSimHitExtended* ?

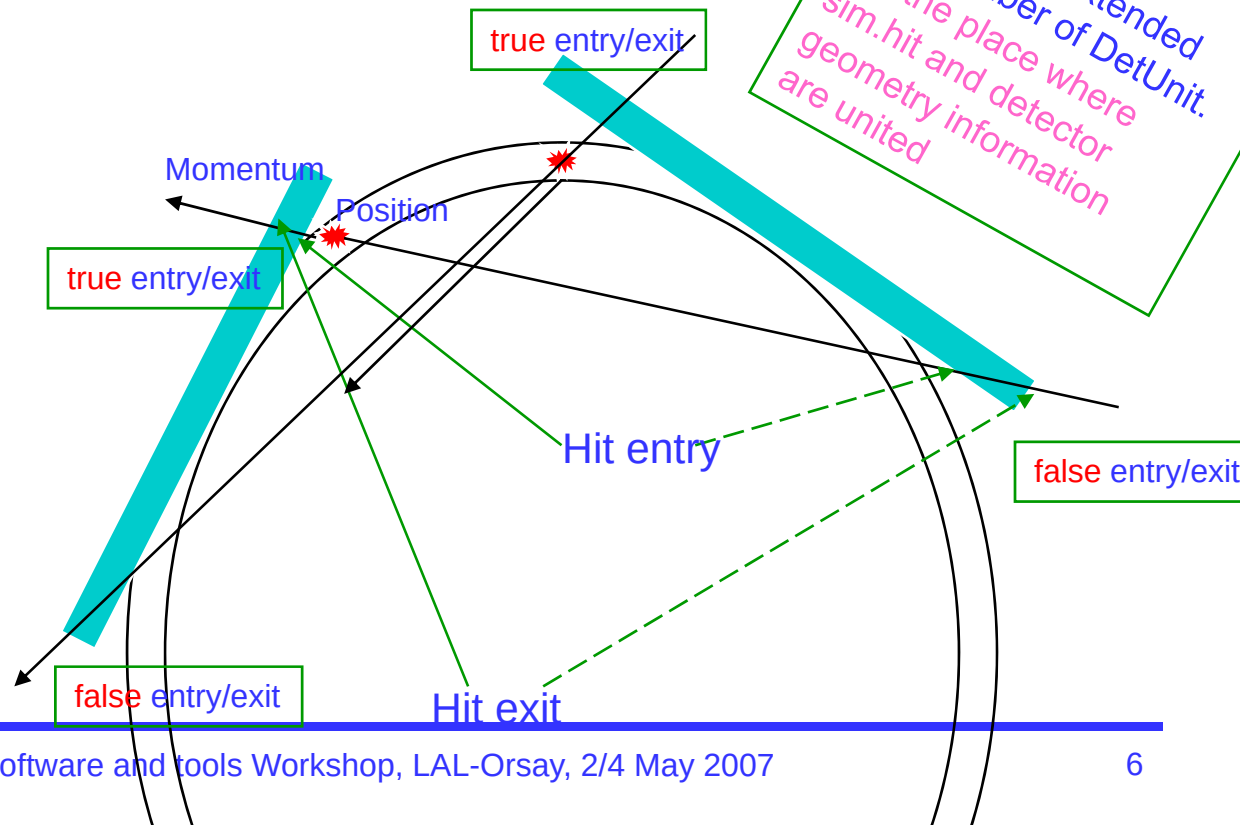
SiTrkSimHitExtended is derived class from LCIO class *IMPL::SimTrackerHit.*

The main additional members are: *Hep3Vector _entryPoint, _exitPoint;*
Entry and exit points are defined in *DetUnit*
by *SimTrackerHit* momentum and position information:

vector of SiTrkSimHitExtended is a member of DetUnit. It is the place where sim.hit and detector geometry information are united

Solution of the ambiguity shown:
nearest entry-exit are true.
It require two loops
over the det.units in *GeomTiling*
for given layer
at stage of *fillDetUnits(SimHit)*.

Flexibility:
It allow us to use
simple Mokka model
for different structure
of layers and det.units



SiTrkDetUnitDigitizer::digitize(DetUnit) (adapted from ORCA)

Primary ionization

NSegments = mag(exit - entry)/segment_length ;

vector< EnergyDepositUnit > _ionization_points ;

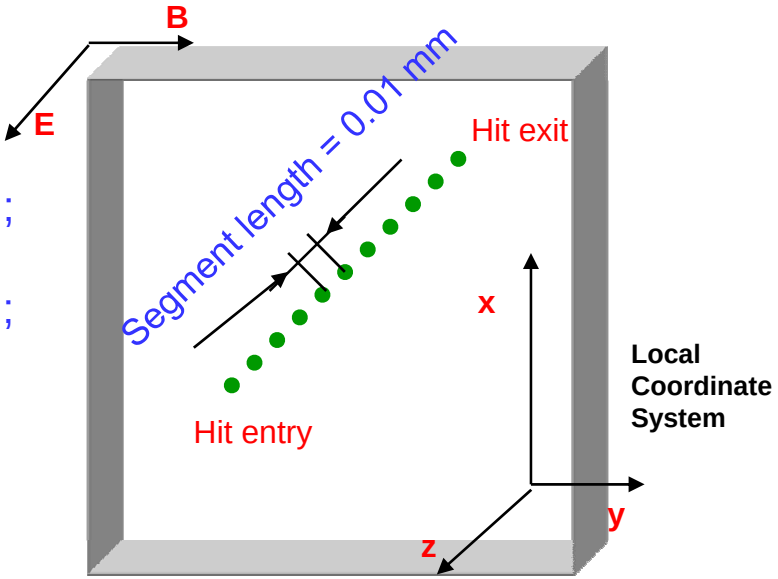
_ionization_points.resize(NSegments) ;

Members of *EnergyDepositUnit*
are *EnergyLoss in electrons*
and Position in local coord.system:

EnergyLoss = EnergyLossGEV[i]/_GevPerElectron ;

_GevPerElectron = 3.7 E-9 ; // 1 electron = 3.7 eV

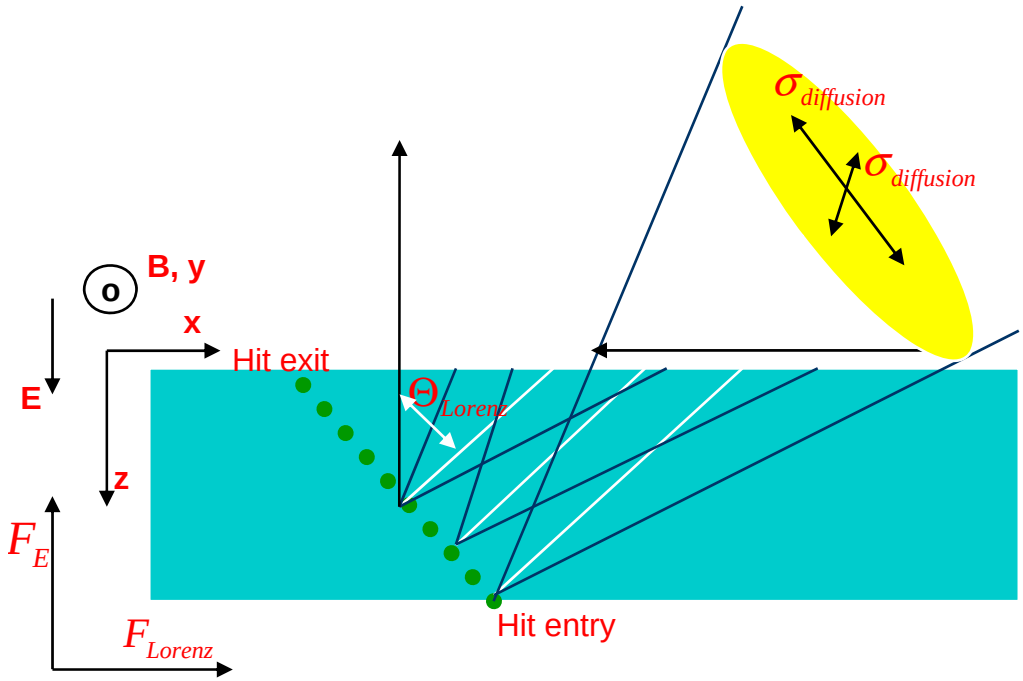
EnergyLossGEV[i] is defined by function
of Landau fluctuation in thin silicon layer
(GEANT4 function *MyG4UniversalFluctuationForSi*).

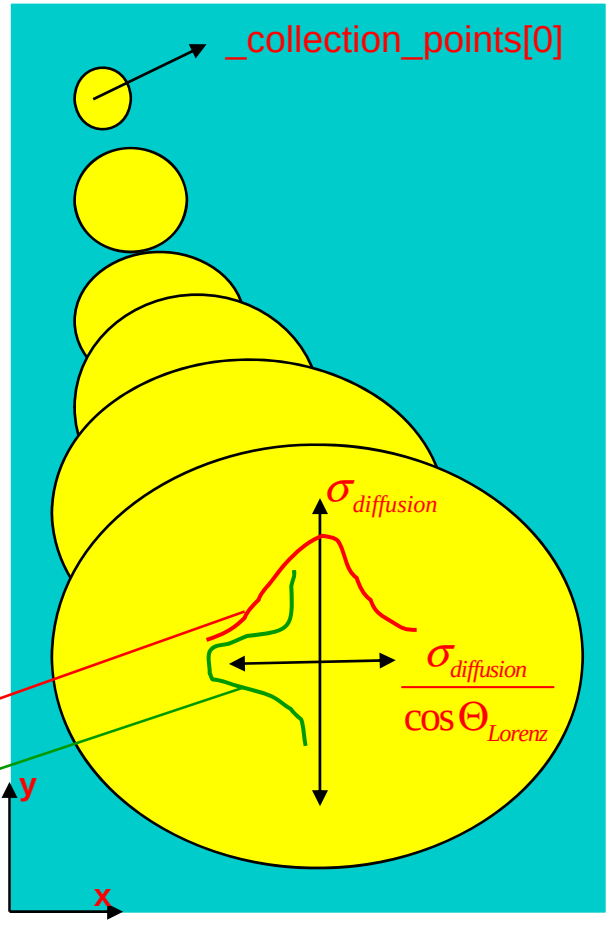


SiTrkDetUnitDigitizer::digitize(DetUnit)

Lorenz drift and diffusion

At the picture 6 points are shown **:**
_collection_points[6] from 6 *_ionization_points[6]*

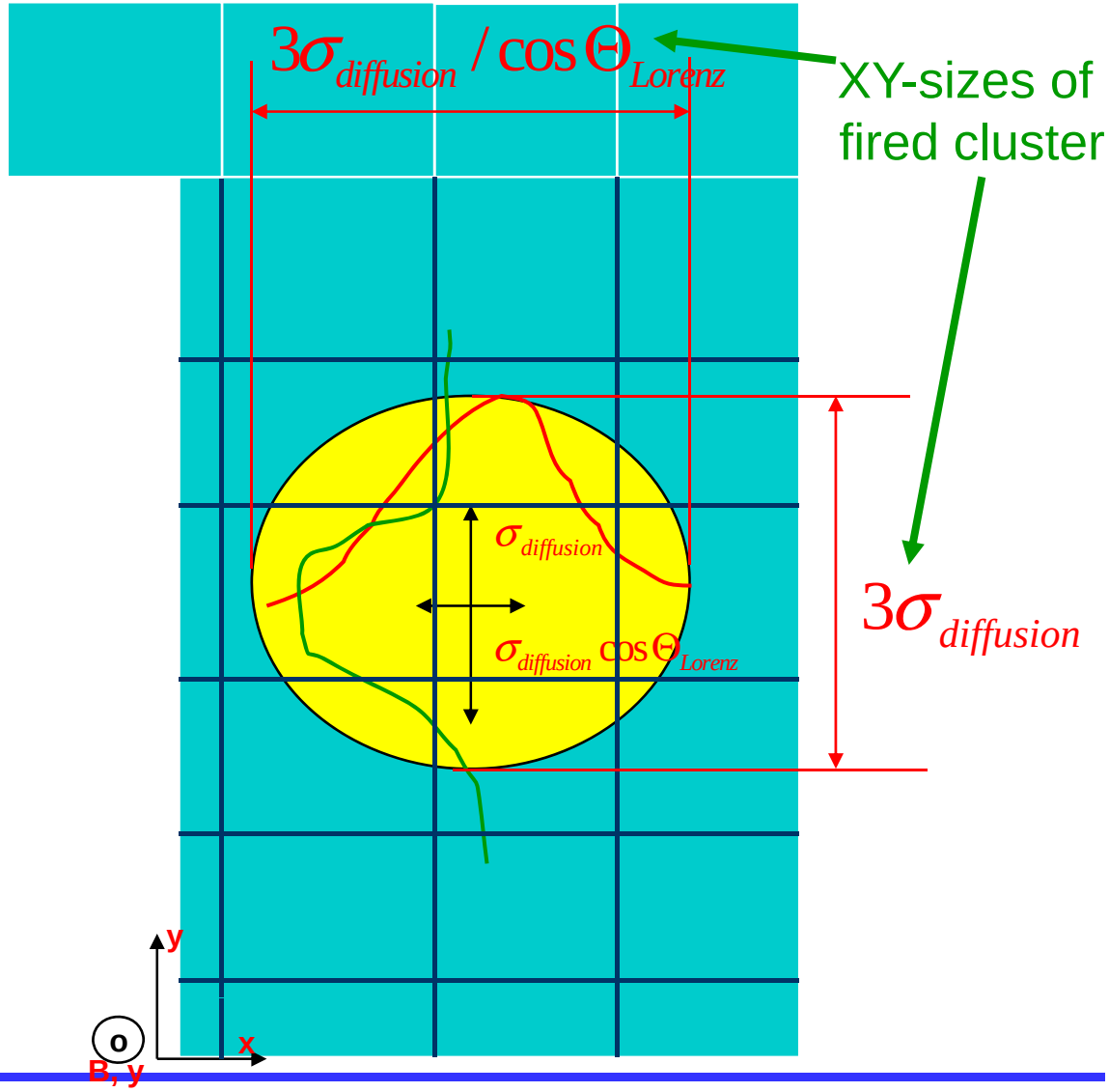




Gaussian charge distribution
along X
and along Y

SiTrkDetUnitDigitizer::digitize(DetUnit)

Induced signal



SiTrkDetUnitDigitizer::digitize(DetUnit)

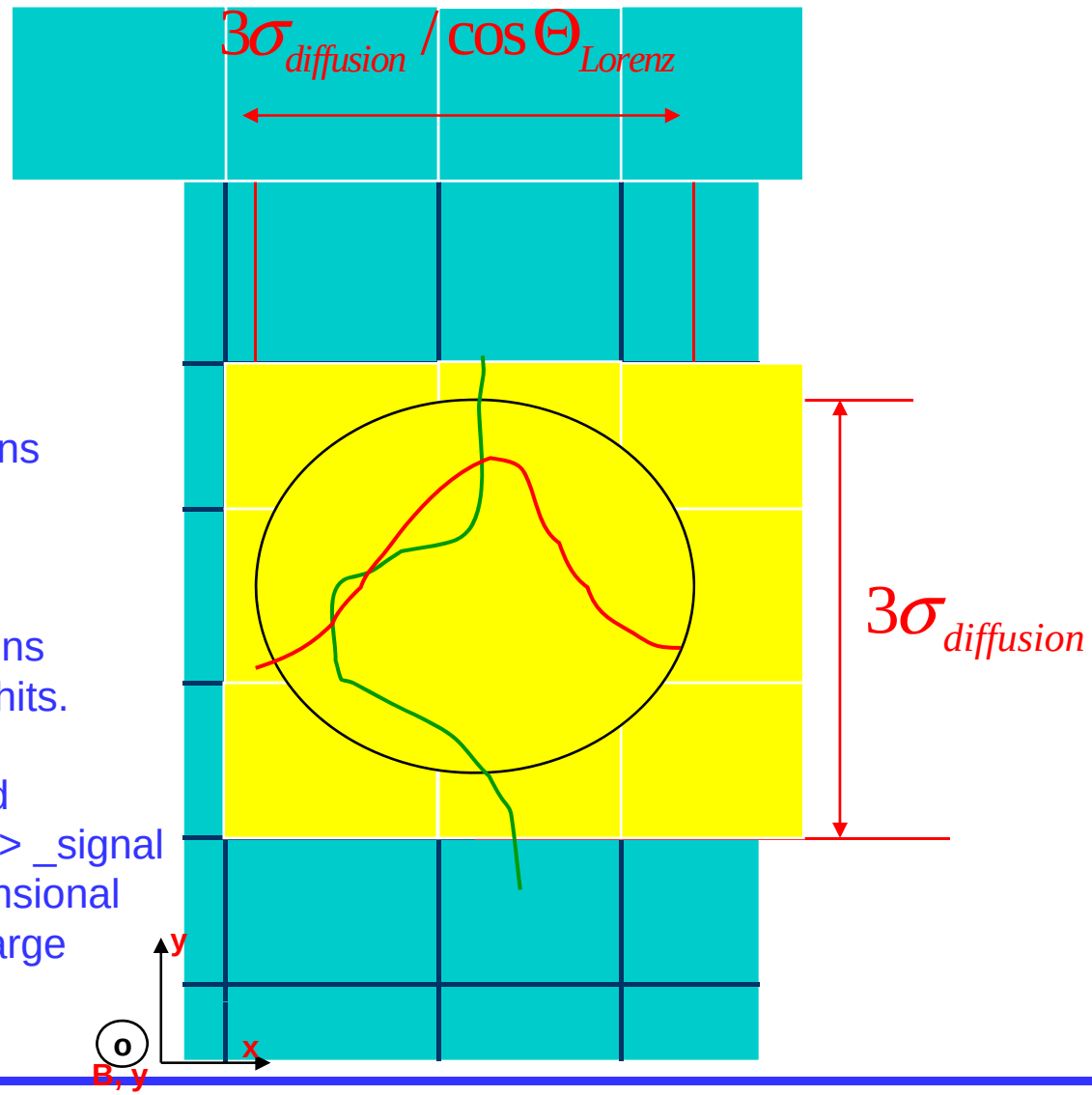
Induced signal

In each fired (yellow) pixel 2-dimensional integral is calculated according to gaussian distribution of charge for each *_collection_point[i].*

For given pixel charge fractions are summarized over the *_collection_point[i].*

For all sim.hits charge fractions are summarized over the sim.hits.

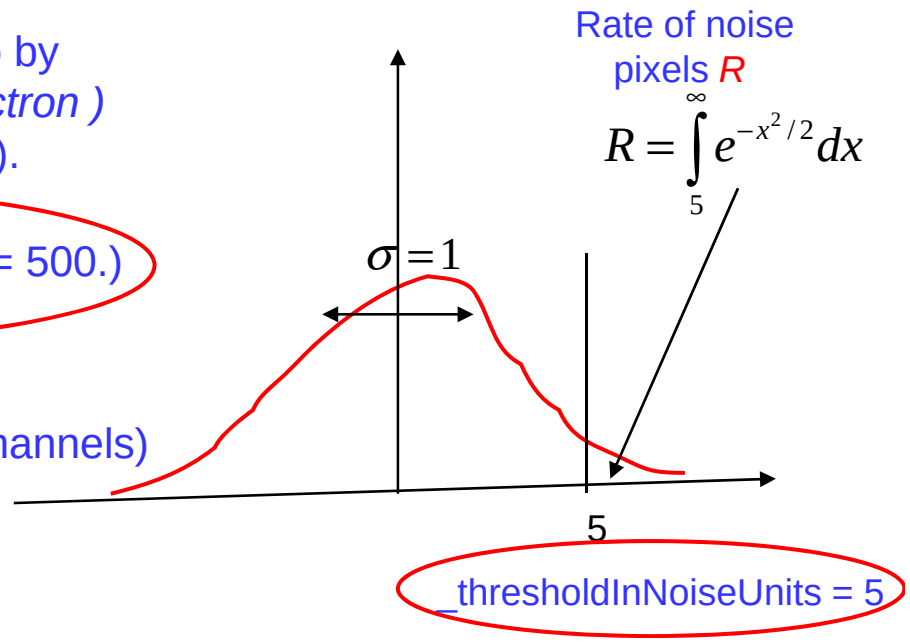
All fired pixels are collected in *map*< int *channel*, double *charge*> _signal where *channel* is packed 2-dimensional pixel number, *charge* is full charge from all sim.hits in event.



DetUnit::add_noise() (adapted from ORCA)

First calculate noise in each hit pixels around zero by
noise = *CLHEP::RandGaus::shoot (0., _noiseInElectron)*
and add them to hit pixel (noise can be negative).

_noiseInElectron == RMS of gaussian distribution (== 500.)



Secondly, calculate noise in non-hit pixels (noisy channels)
by *GaussianTileNoiseGenerator _theNoiser ;*

Mean number of noise pixels
Nmean = *R* * *Npixels*

Number of noise pixels *Npix* = *CLHEP::RandPoisson::shoot(Nmean)*

Next step is to calculate *Npix channel numbers*
= flat random number in [0, Npix].
It will be the numbers of noisy channel (pixel).
In each noisy channel
noise = random value of Gaus.distr.(mean=0, sigma= 500.)
in interval $[5*500, \infty]$

Two noise parameters:

_noiseInElectrons

and

_thresholdInNoiseUnits

DetUnit::pixel_inefficiency()

Inefficiency parameters :

_pixelEfficiency = 99%, for single pixel

_pixelDoubleColumnEfficieny = 99%, for double column of pixels

_pixelReadOutChipEfficiency = 99.75% for readout chip

To calculate ROC inefficiency two parameters are used:

_rocSizeInX and _rocSizeInY (in units of pixel sizes)

List of default parameters of *SiTrkDigiProcessor*

Ionization segment length = 0.01 mm

Lorenz drift and diffusion :

$$\tan\Theta_{Lorenz} = 0.106 * B, B = 4Tesla$$

$$\sigma_{diffusion} / (0.300mm) = 0.007$$

fired cluster width = $[3*\sigma_X, 3*\sigma_Y]$, $\sigma_X, \sigma_Y = f(\Theta_{Lorenz}, \sigma_{diffusion})$

Noise parameters :

_noiseInElectrons = 500.,

_thresholdInNoiseUnits = 5 ;

Inefficiency parameters :

_pixelEfficiency = 99%,

_pixelDoubleColumnEfficieny = 99%,

_pixelReadOutChipEfficiency = 99.75%

***_rocSizeInX* = 20 pixels**

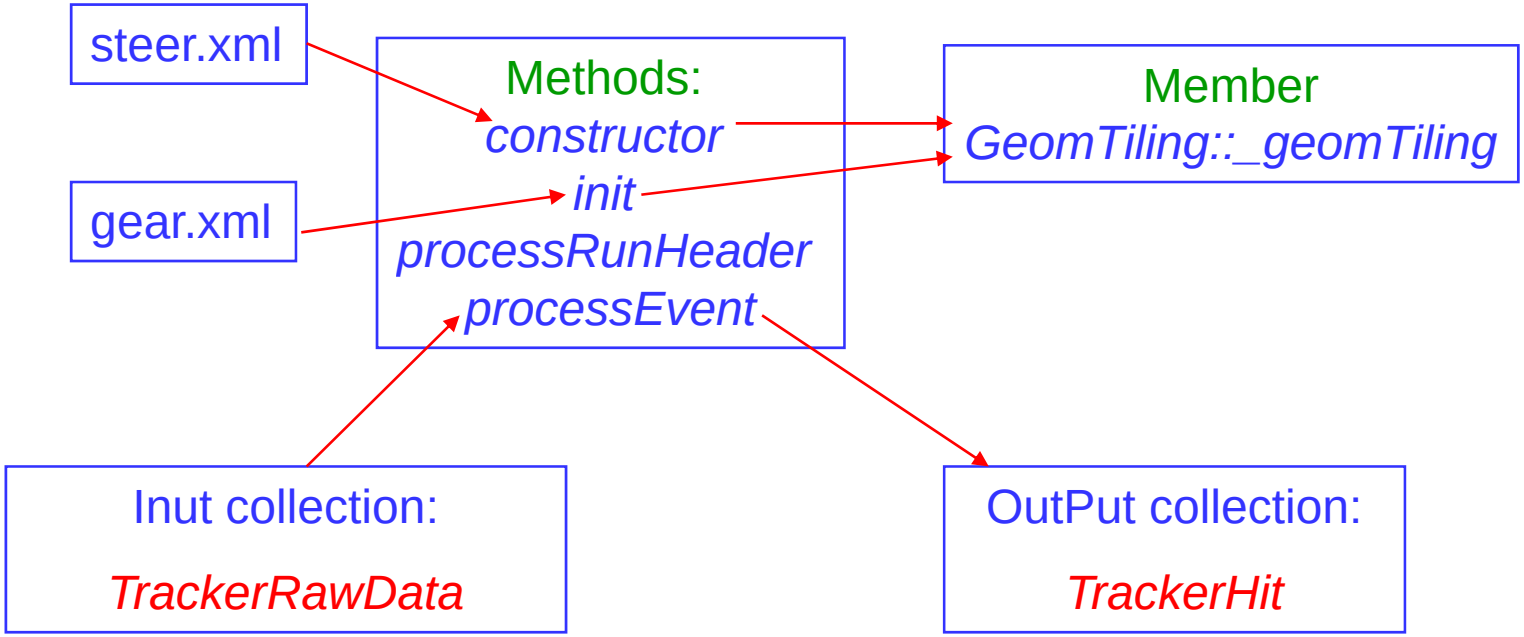
***_rocSizeInY* = 52 pixels**

Pixel sizes

***_pixSizeX* = 0.100 mm**

***_pixSizeY* = 0.150 mm**

Clustering: class *SiTrkClusteringProcessor*



Class *GeomTiling*

Main members

- *vector< vector< DetUnit * > > _detUnitVecVec*
- *SiPixelDetUnitDigitizer _pdigitizer*
- *SiPixelDetUnitClusterizer _pclusterizer*
- *SiStripDetUnitDigitizer _sdigitizer* to be done
- *SiStripDetUnitClusterizer _sclusterizer* to be done

Main methods

- *gearInit*
- *gearInitVXD* (called by *gearInit)*
- *gearInitFTD, gearInitSIT* to be done
- *fillDetUnits(SimTrackerHit *);*
- *digitizeDetUnits(vector<LCCollectionVec *> &)*
- *fillDetUnits(TrackerRawData *)*
- *clusterizeDetUnits(vector<LCCollectionVec *> &)*
- *clearDetUnits()*
- *howManySimHits()*
- *howManyRawHits()*

Abstract base class *DetUnit*

Main members

- *vector< IMPL::TrackerRawDataImpl *> _rawHits ;*

_seedThresholdInNoiseUnits,
_clusterThresholdInNoiseUnits;

Main methods

the threshold of channel charge to start clusterizing around seeded clusters (DEFAULT = 6)

the threshold of cluster charge to kill Some found clusters (DEFAULT = 10.1)

vector< IMPL::TrackerRawDataImpl> getRawHits()*

No any special methods for clustering In DetUnit class

class *SiTrkDetUnitClustering*

The clusterization is performed on a matrix with size equal to the size of the pixel detector.
Each cell contains the ADC count of the corresponding pixel.

The search starts from seed pixels, i.e. pixels with sufficiently large amplitudes.

Clusters are set of neighbour pixels including pixels which touched by corners.

ORCA/COBRA set of classes are used.

Performance: Mokka06-01, model "vxd_00",

MC events: Pythia6.410, MSEL=6, $e^+e^- \to t\bar{t} \to X$, c.m.s. energy=500 GeV

Preliminary

Event number	1	2	3	4	5	
Nb of charged MC particles	32	36	50	18	44	
Nb of MC part. X 5 Layers	160	180	250	120	220	
Sim.hits in VXD :	227	198	336	69	397	
Nb of sim.hits with p>1GeV:	105	125	170	41	124	
Raw hits in VXD:	595	428	780	164	761	pixels 50x75
Clusters in VXD:	138	103	169	37	211	
Noises = 0, inefficiencies = 0						
Raw hits in VXD:	544	374	725	150	684	pixels 50x75
Clusters in VXD:	137	105	178	39	202	
Raw hits in VXD:	290	168	326	81	77	pixels 70x100
Clusters in VXD:	69	31	69	15	62	
Raw hits in VXD:	101	43	102	32	73	pixels 100x150
Clusters in VXD:	17	4	29	4	8	

Plan

- Pixel mode for FTD layers
- Strip digitizer for FTD and SIT
- Clustering and hit reconstruction for strip detectors
- Performance study together with track finding processors

Digitization and clustering processors described above will be placed in official *MarlinReco* CVS directory till LCWS07