HighRR TFR Hands-On: simulation framework

Alessio Piucci

29 May 2017
HighRR TFR Hands-On

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

HighRR

First of all...

https://www.youtube.com/watch?v=aTwbDAYBVxA

First of all...

https://www.youtube.com/watch?v=aTwbDAYBVxA

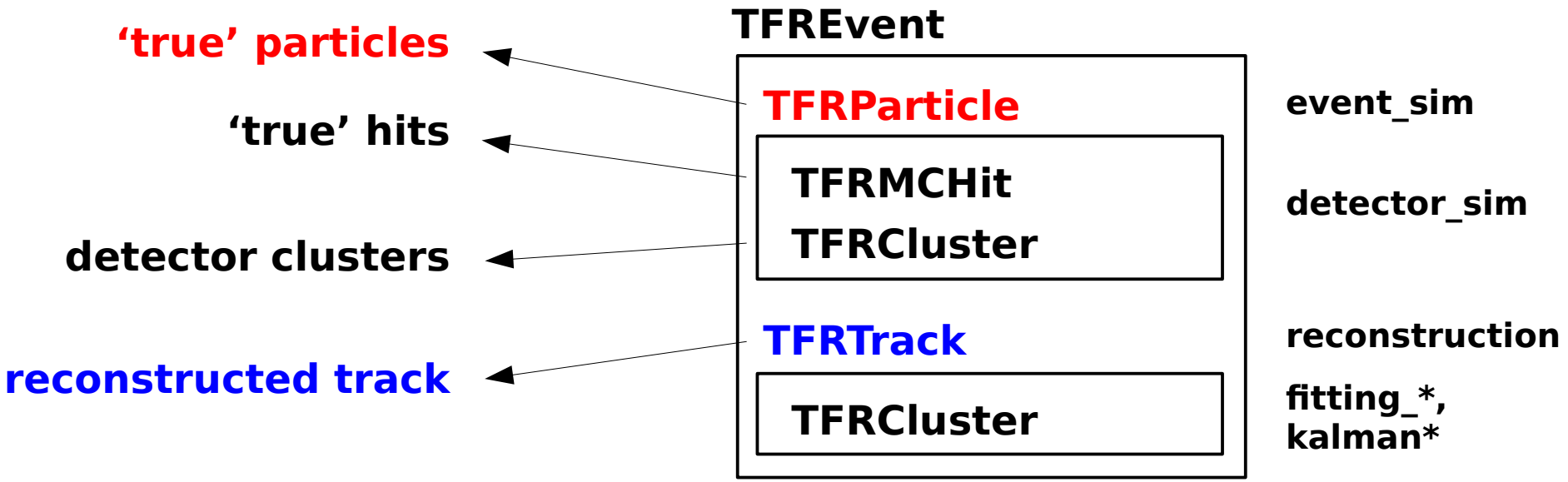
... and replace your rifle with your C++ compiler :)

The framework

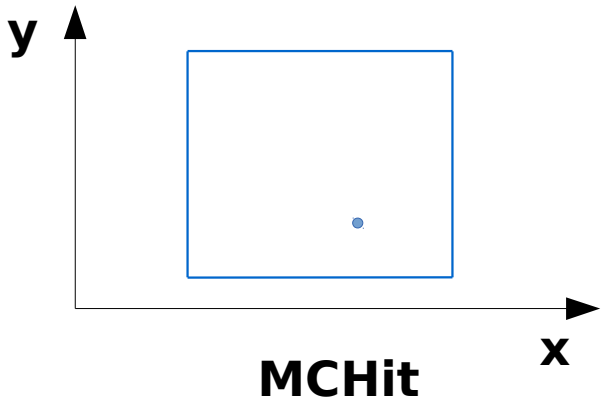
- stand-alone framework: Made in INF 226 03.215 – 03.210
- C++ - based
- relies on Root, Boost, Cmake
- deployed on GitLab

Different steps:
- simulation of events and particles
- detector simulation
- reconstruction
- fitting
- decay reconstruction

Basic classes

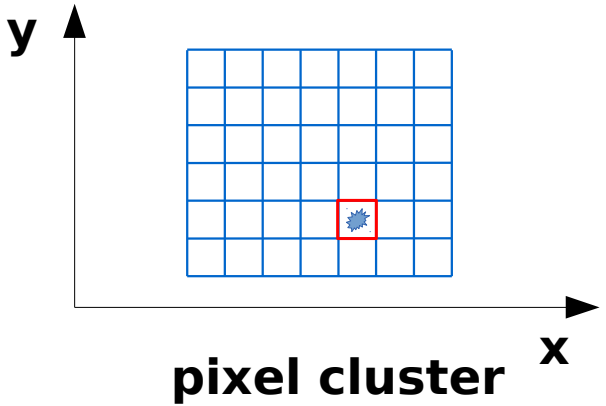


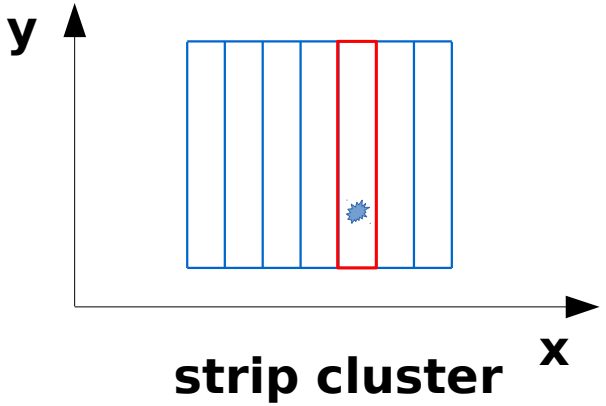
MCHits vs clusters



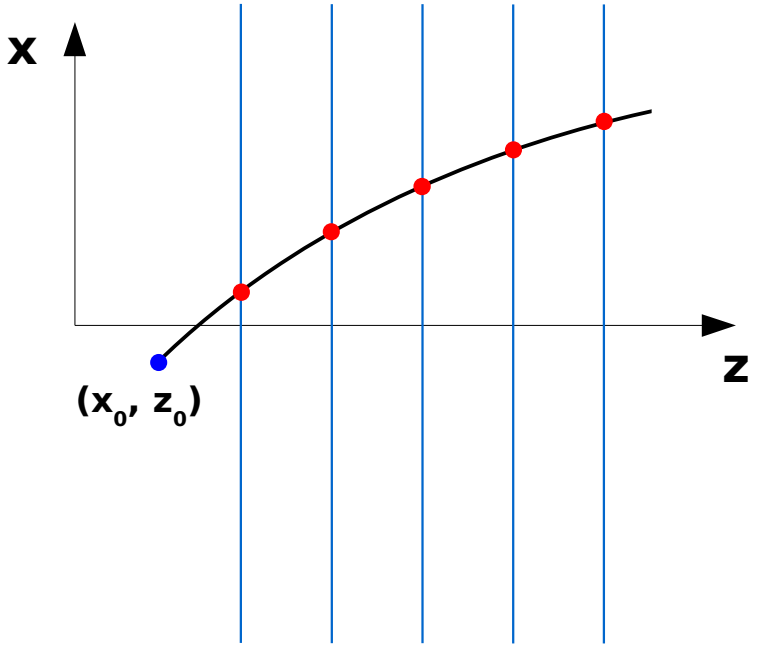
Detector:
- smearing
- multiple scatt.

Digitization:
- sensor res.



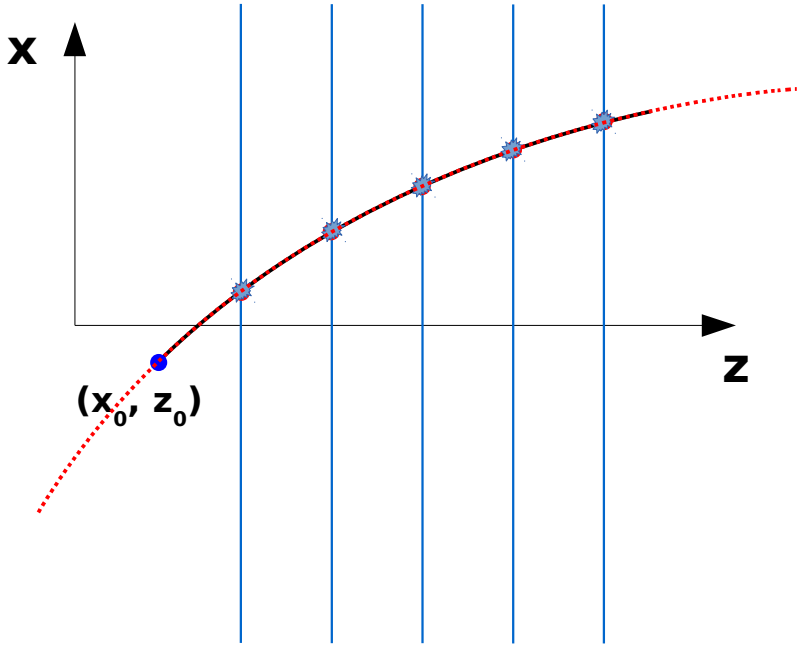


Particles vs tracks



Particle
MC-simulated particle,
leaving MCHits on the detector

Particles vs tracks



Particle
MC-simulated particle, leaving MCHits on the detector

Track
Reconstructed trajectory of the particle based on the clusters

Particles vs tracks - comments

Vertex

- well defined for a particle only
- for a single track you have to choose a reference:
  usually the intersection with some axis

Momentum

- to describe the 3 components you have to choose a reference:
  usually at the interaction (decay) vertex
- modulus is conserved

State

- tracks in 3D and B are parametrized by 5 parameters ($z_0 == 0$)
- a point in the parameter space (at fixed z) is called state:

$$s = (x, y, t_x, t_y, qop)$$
$$= (x, y, p_x/p_z, p_y/p_z, q/p)$$

Config files – B field

Event simulation and detector parameters defined in config files

```
; settings of the magnetic field
B_field {

   ; simple magnetic field: uniform, constant, long y direction only
   z_min       300.   ; [cm]
   z_max       400.  ; [cm]
   magnitude   4    ; [T]
}
```

Config files – Detector layers

```
; detector layers
layers {

   layer_0 {

      ; set the sensor
      #include "config/pixel.info"

      ; center position of the layer [cm]
      position {
         x    0.
         y    0.
         z    50.
      }

      ; layer size [cm]
      x_size    10000000.
      y_size    10000000.

      ; simple layer: perpendicular to the beam, not tilted
      #include "config/simple_layer.info"
   }
}
```

strips/fibers also implemented, for hard-core players only

Flowchart

$ source runSim.sh

→ event_sim : simulation of events and MC particles
→ detector_sim : simulation of MC hits, creation of clusters
→ reconstruction : (fake) reconstruction of tracks from clusters: 100% efficiency, 0% ghost rate
→ fitting : fitting of reconstructed tracks
→ make_ntuples : creation of Root ntuples with generated and reconstructed quantities, for comparison

What is already there

What is already implemented:

- generation of kinematic values of particles

- detector layer description

- single MC hit → cluster digitisation

What is missing?

Particle propagation in B

Simple B: uniform (in space), constant (in time), only By != 0

→ xz bending plane, circular trajectory

Task: implement a particle propagator in uniform B

Material: Wiki / Propagator

TFRPropagator

- complete the TFRPropagator class

- do not take in account multiple scattering (for now...)
- you can implement new methods if you need
- do not remove the methods already defined
- do not change the input arguments and output of the methods which are already there

- for debug purpose use the geometry:
config / config_geo_B_debug.info
and check the output root file:

```
particle_tree->Draw("particle_MCHit_x:particle_MCHit_z",
                    "(evtID == 1) && (particle_ID == 1)", "*")
```

with same z and x ranges you obtain a circular trajectory in B

TFRMultipleScattering

Task: implement a Gaussian model for multiple scattering

Material: Wiki / Multiple Scattering

- angular scattering only;
- MC hits are also affected!
- the propagator covariance matrix has to take in account it

A few suggestions

Magic to build the framework:

```
mkdir build
cd build
cmake ../
make -j 8
```

To run the full simulation (including building):

```
source runSim.sh
```

Documentation (evolving every day):

http://physi.uni-heidelberg.de/%7Epiucci/HighRR_TFRHandsOn/html/annotated.html

A few suggestions

Suggested Root class to use. More info searching on Google!

TVector3
3-dimensional vector

```
TVector3 v(0., 1., 2.)
```

TVectorD
n-dimensional vector

```
double array[4] = {0., 1., 2., 3.}
TVectorD v(4, array)
```

TMatrixD
n-dimensional matrix

```
double matrix[2, 2] = {0., 1, 2., 3.}
TMatrixD m(2, 2, matrix)
```