24-1-2019

SFGD beam test analysis

Towards an event reconstruction software

C. Jesús-Valls
cjesus@ifae.es

CONTEXT:

- The upgrade activities require to develop software tools to convert, classify and analyze the input data from the future ND280.
- Before thinking in global reconstruction it is necessary to locally reconstruct the event in each detector module.
- SFGD reconstruction is particularly important being a key element for event characterization.
- The evaluation of the performance of the SFGD prototypes needs a minimal reconstructing software.

GOAL:

- The purpose of this talk is to present a possible way to develop the SFGD final reconstruction software while using its versions in parallel to study the prototypes data.

CURRENT STATUS AND NEW WORK-FLOW PROPOSAL

CURRENTLY:

going from raw-data to event visualization takes several steps:

1. Extract .daq FEB files from each mini-crate's .daq file.
2. Convert .daq to .root
3. Apply calibration to .root
4. Convert calibrated .root to the new data structure.

The process is slow and time consuming, and has to be re-done each time something is changed.

PROPOSAL:

Write a script that does all steps and send each run as a job.

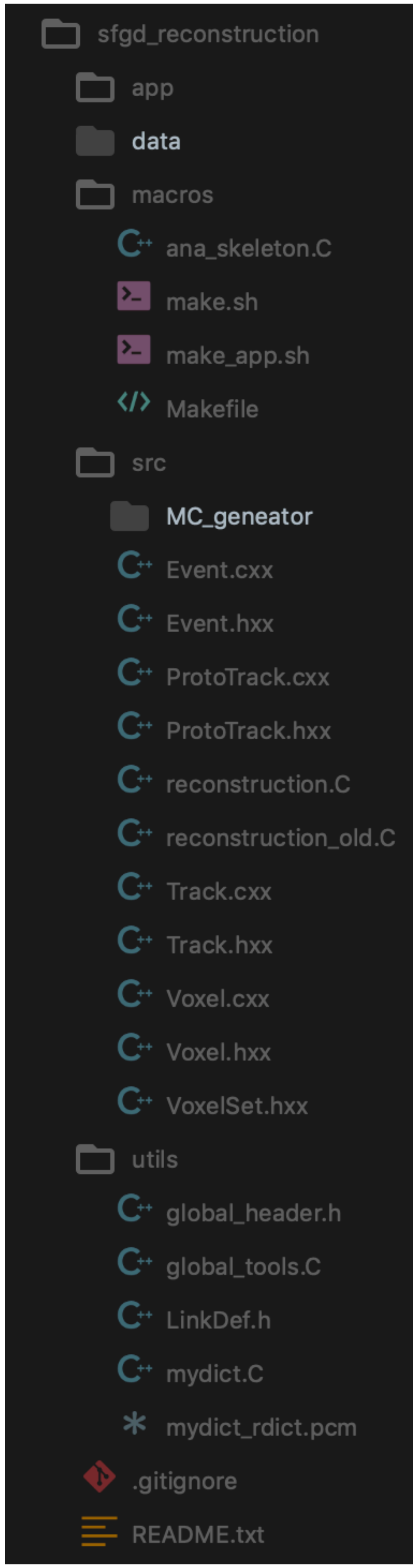
All data files could be stored in: `/eos/experiment/neutplatform/t2knd280/Super_FGD/data/`

with a common numeration: run0001, run0002, run0003, … and add a new column with run number to the Google's Log.

WHY:

- Easier access to all data.
- All data can be updated at once by running a single script.
- New numeration will make easier to send jobs with sets of runs in the close future.
- It is easy to do!

CURRENT STATUS AND NEW WORK-FLOW PROPOSAL



```
sfgd_reconstruction
  app
  data
  macros
    ana_skeleton.C
    make.sh
    make_app.sh
    Makefile
  src
    MC_geneator
    Event.cxx
    Event.hxx
    ProtoTrack.cxx
    ProtoTrack.hxx
    reconstruction.C
    reconstruction_old.C
    Track.cxx
    Track.hxx
    Voxel.cxx
    Voxel.hxx
    VoxelSet.hxx
  utils
    global_header.h
    global_tools.C
    LinkDef.h
    mydict.C
    mydict_rdict.pcm
  .gitignore
  README.txt
```

MOTIVATION: There is an extra motivation to do this!

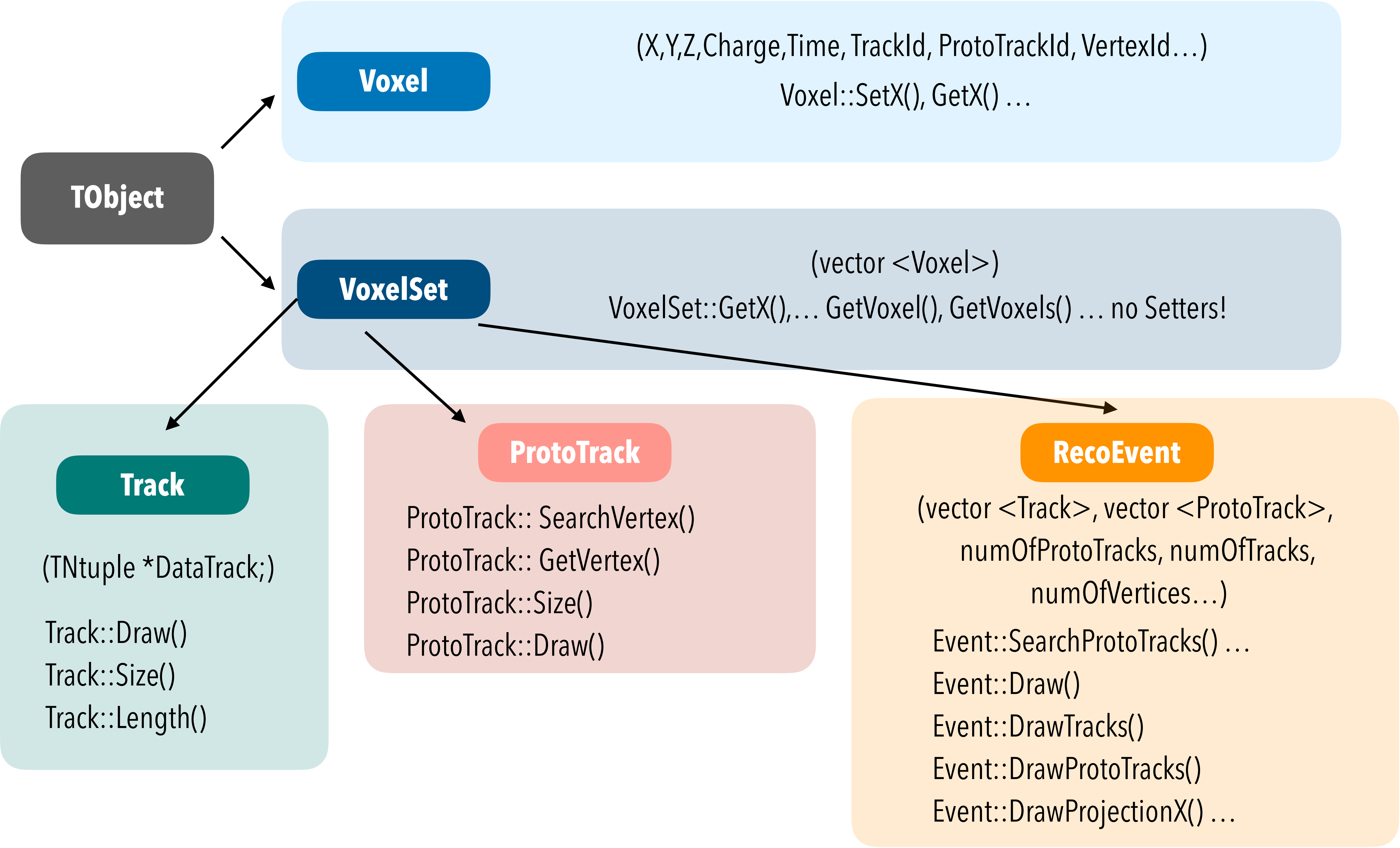
- Once all newDataStructure.root are available, we can start working with a much more specific and powerful SFGD reconstruction software that will be used to do **only analysis**.

- The package to do that it **is already existing.** Is is inspired on the structure of TPC beam test analysis software. You can find it in gitlab: (https://gitlab.com/cesarjesusvalls/sfgd_reconstruction.git)

- The new package is easy to use. Allows to search tracks applying cuts and conditions, interactively look 3D events, jump from one selected event to the next one with a mouse click… and much more!

ABOUT THE RECONSTRUCTION PACKAGE

The classes hierarchy is the following:



TObject

Voxel

(X,Y,Z,Charge,Time, TrackId, ProtoTrackId, VertexId…)

Voxel::SetX(), GetX() …

VoxelSet

(vector <Voxel>)

VoxelSet::GetX(),… GetVoxel(), GetVoxels() … no Setters!

Track

(TNtuple *DataTrack;)

Track::Draw()

Track::Size()

Track::Length()

ProtoTrack

ProtoTrack:: SearchVertex()

ProtoTrack:: GetVertex()

ProtoTrack::Size()

ProtoTrack::Draw()

RecoEvent

(vector <Track>, vector <ProtoTrack>, numOfProtoTracks, numOfTracks, numOfVertices…)

Event::SearchProtoTracks() …

Event::Draw()

Event::DrawTracks()

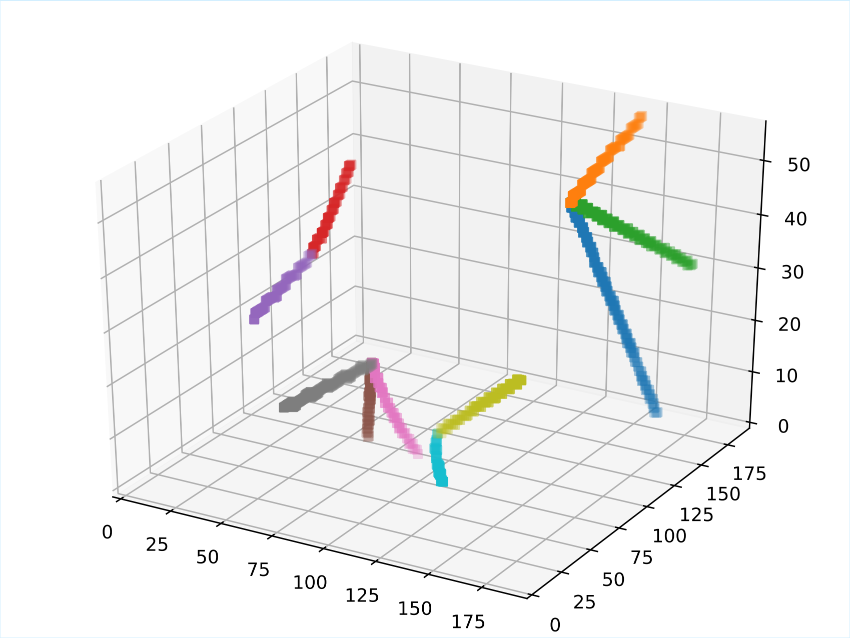
Event::DrawProtoTracks()

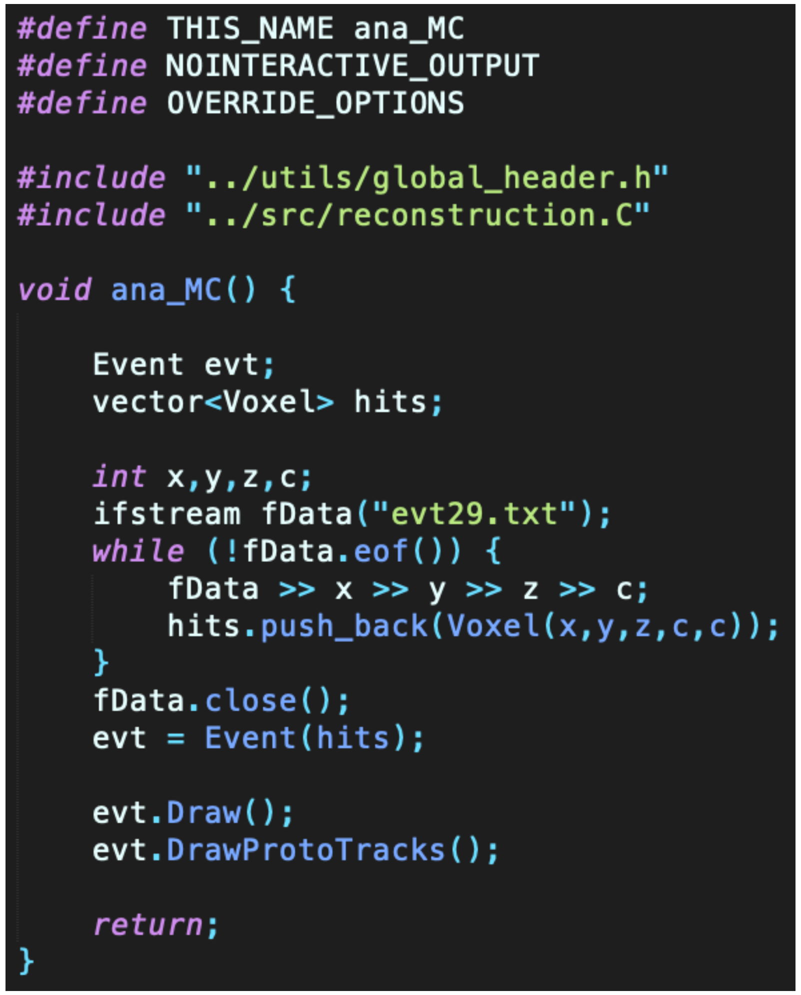
Event::DrawProjectionX() …

MC EXAMPLES

the MC is available to anyone (in python) to make short tests and clustering debug for development.

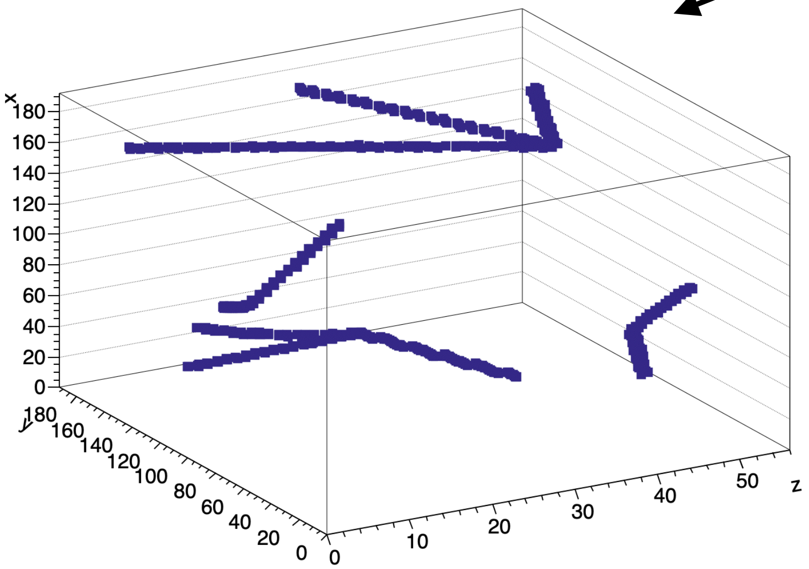
data from MC



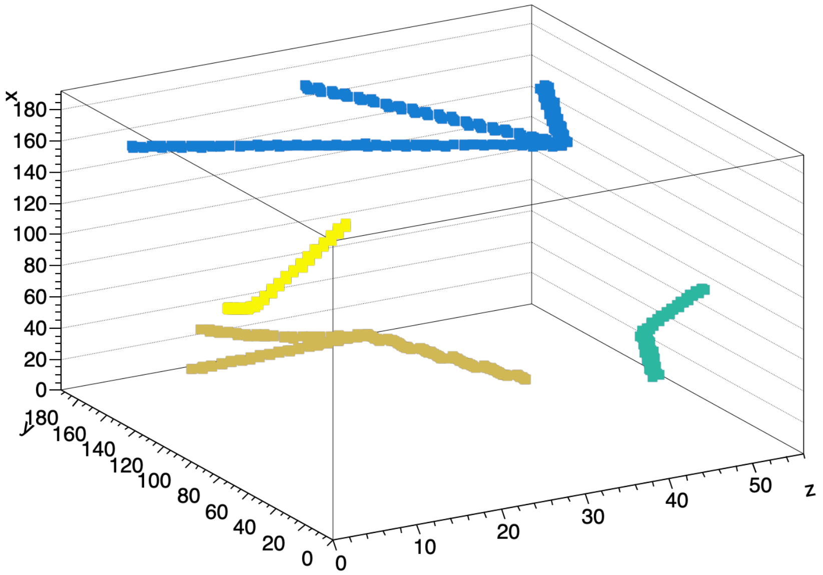


```
#define THIS_NAME ana_MC
#define NOINTERACTIVE_OUTPUT
#define OVERRIDE_OPTIONS

#include "../utils/global_header.h"
#include "../src/reconstruction.C"

void ana_MC() {

    Event evt;
    vector<Voxel> hits;

    int x,y,z,c;
    ifstream fData("evt29.txt");
    while (!fData.eof()) {
        fData >> x >> y >> z >> c;
        hits.push_back(Voxel(x,y,z,c,c));
    }
    fData.close();
    evt = Event(hits);

    evt.Draw();
    evt.DrawProtoTracks();

    return;
}
```

Event::Draw()



Event::DrawProtoTracks()

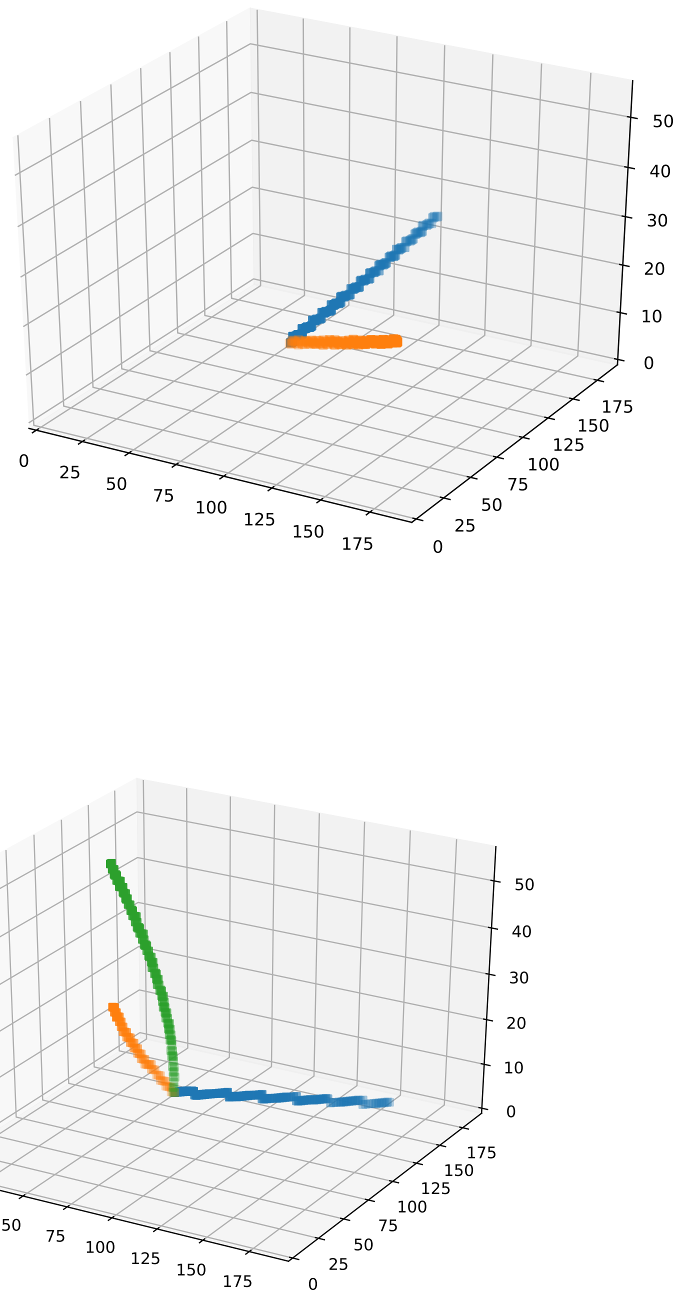


few lines of code do a lot of helpful work

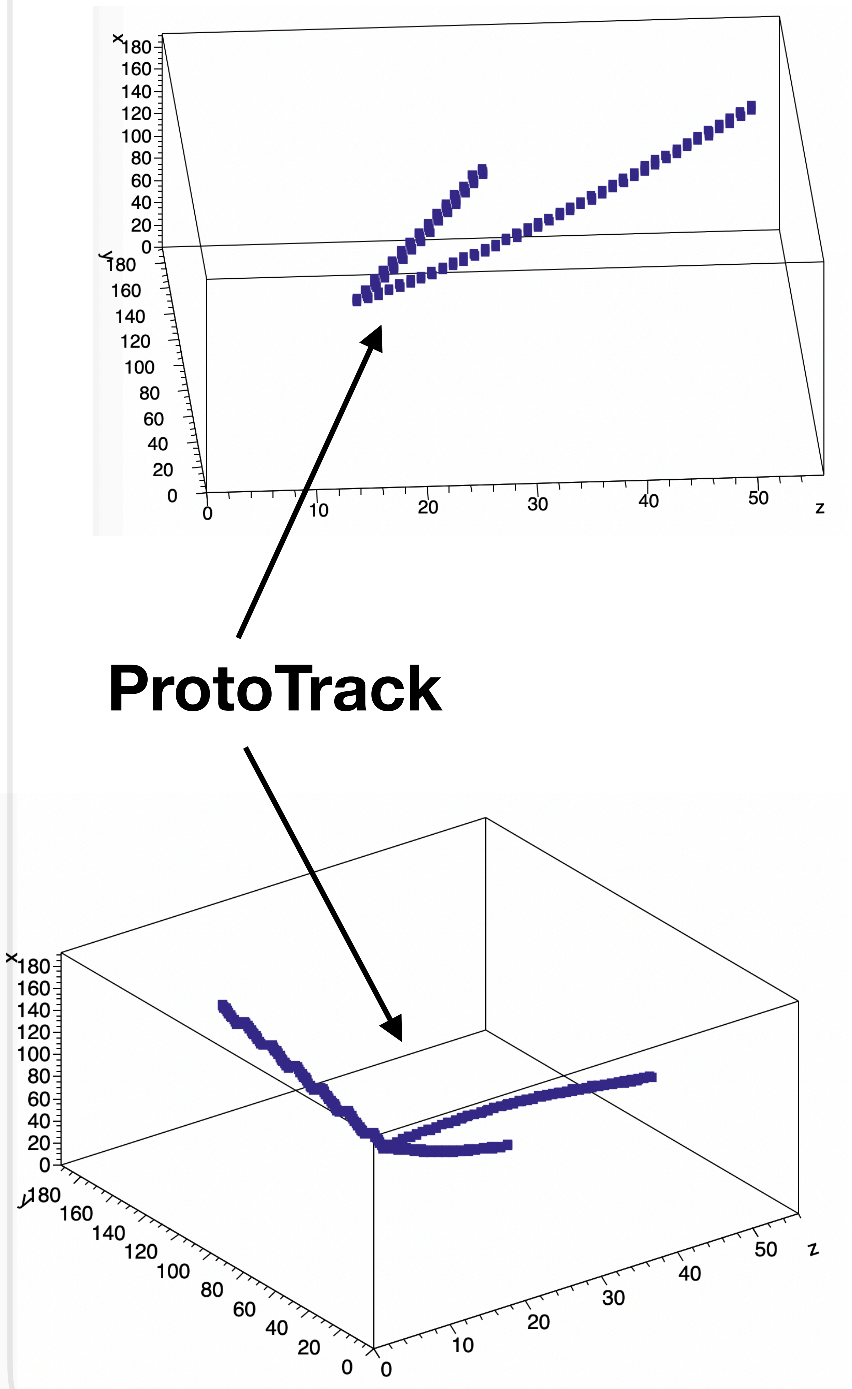
MC EXAMPLES

consider a single ProtoTrack

data from MC

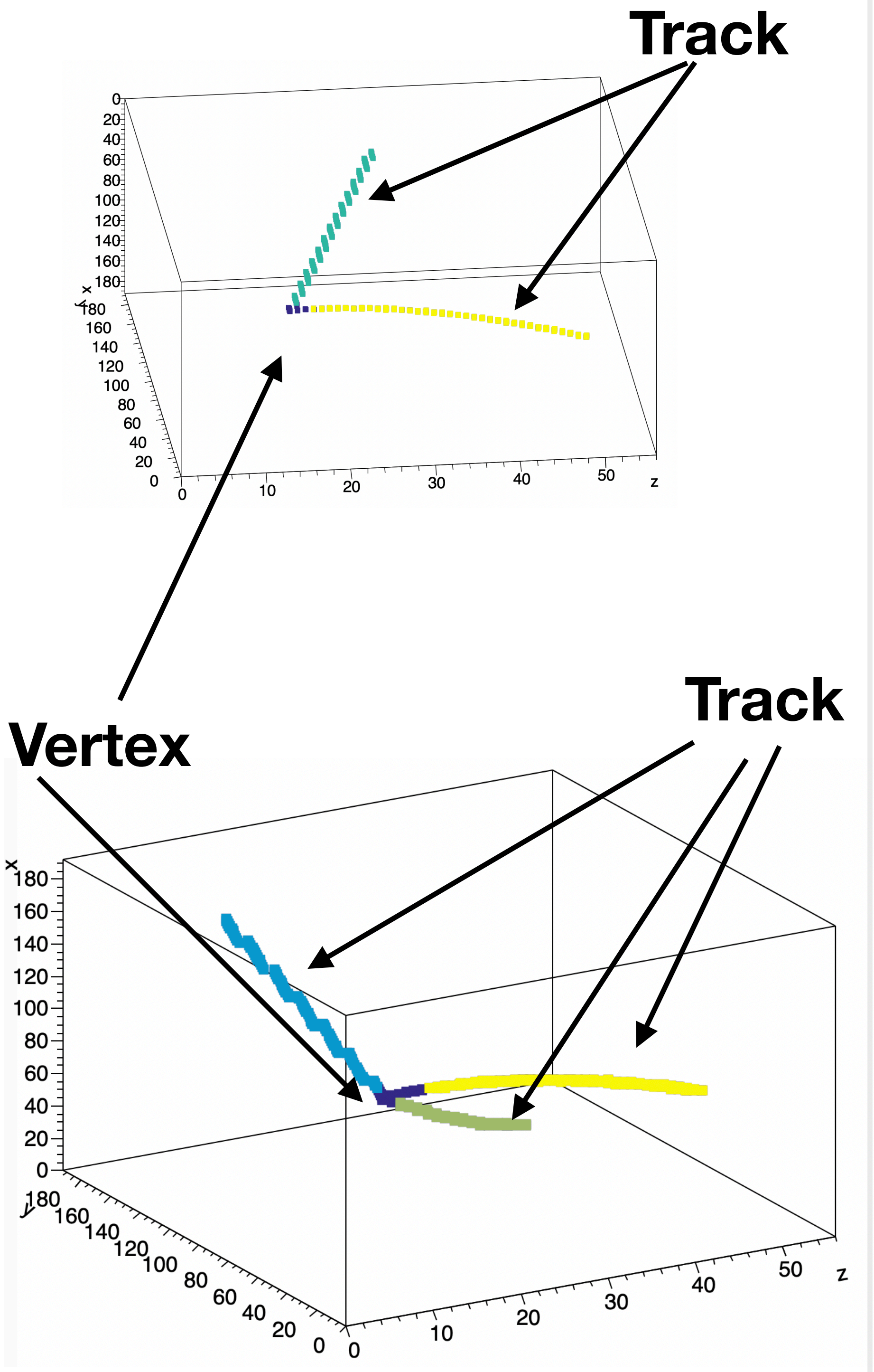


Raw



ProtoTrack

After



Track

Vertex

Track

UNDERLYING IDEAS

The reconstruction algorithm is thought to work **sequentially** in 'independent' steps.

1. Cluster the Voxels in ProtoTracks.
2. Search Vertex in ProtoTracks.
3. Split de ProtoTracks in Tracks.

…

It will of course grow and increase its complexity with time, but it allows to work in a single task.

EXAMPLE:

Imagine that some module can be improved, e.g: Event::SearchVertex().
The algorithms that searches if there is vertex or not and localize them can be tuned without changing anything else. In this way many people can join efforts to build a powerful tool!

CURRENT ALGORITHMS:
I will not enter today in the details of the algorithms, I can do it in a future meeting if you are interested.

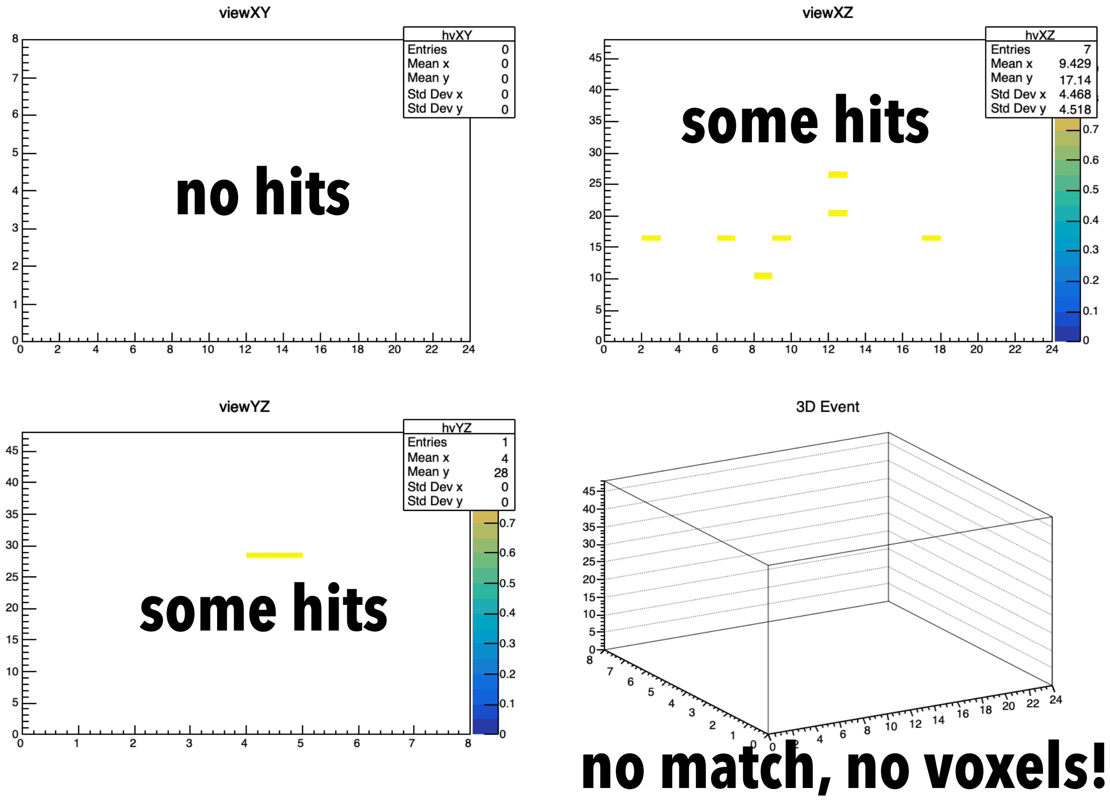
MOVING TO REAL DATA

- The algorithms work in 3D, since it is more powerful to take advantage of all dimensions.
- So far newDataStructure Events had 2D projections.

from 2D to 3D: **with the collaboration of T.Lux**

From the projection XY we look to match 'X' with XZ, from the XY we look to match 'Y' with YZ and from XZ with YZ we look to match 'Z'.

There are many ways to do this! What happens if one MPPC is not working? We loose one data point!
We can also reconstruct points that match in 2 of the 3 projections. The best? Is still not clear . **Suggestions?**



no hits

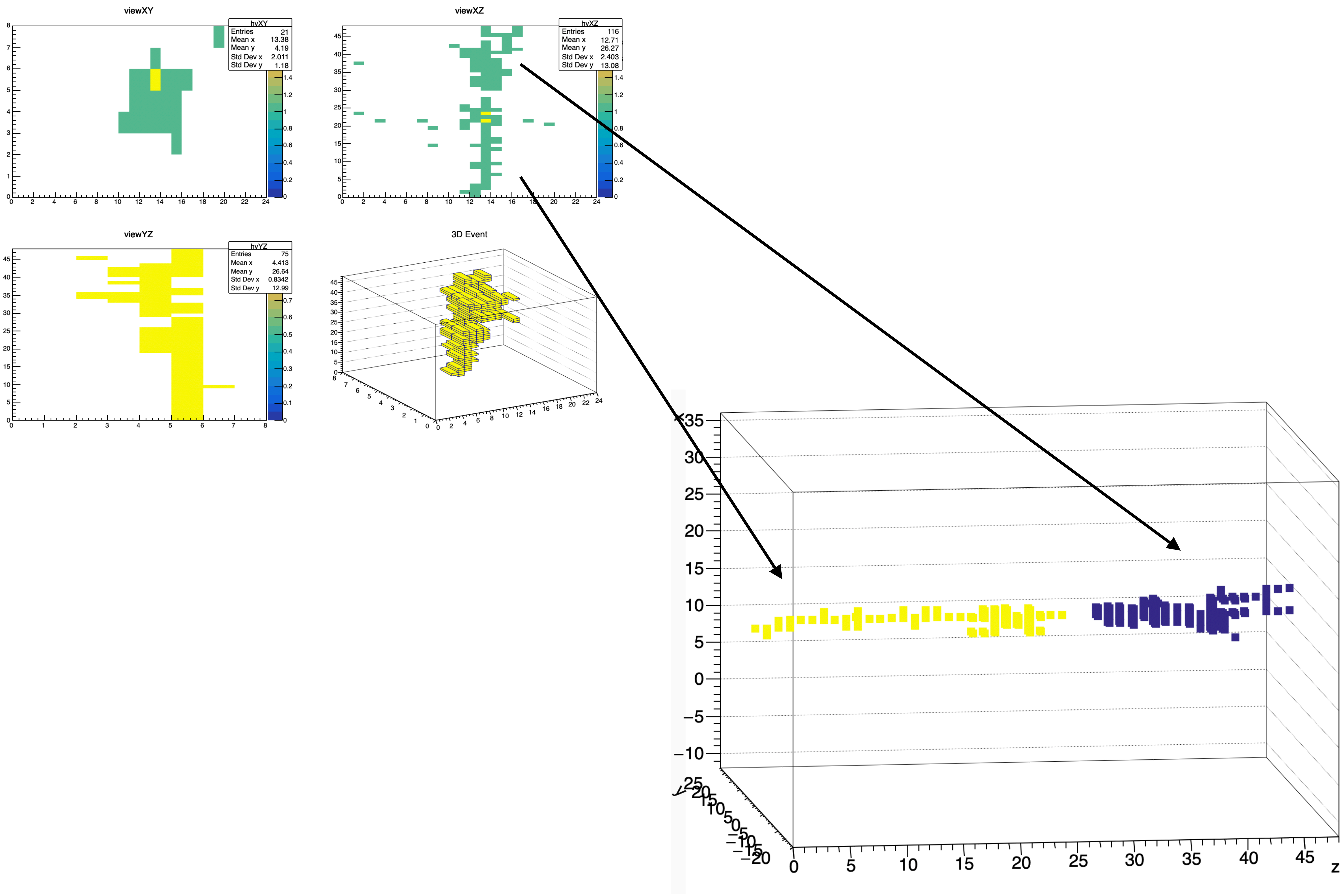
some hits

some hits

no match, no voxels!

Charge reconstruction is not done.
Is one of the next steps…

RECONSTRUCTION ON REAL DATA


viewXY
hvXY
Entries 21
Mean x 13.38
Mean y 4.19
Std Dev x 2.011
Std Dev y 1.18
viewXZ
hvXZ
Entries 116
Mean x 12.71
Mean y 26.27
Std Dev x 2.403
Std Dev y 13.08
viewYZ
hvYZ
Entries 75
Mean x 4.413
Mean y 26.64
Std Dev x 0.8342
Std Dev y 12.99
3D Event
z
y