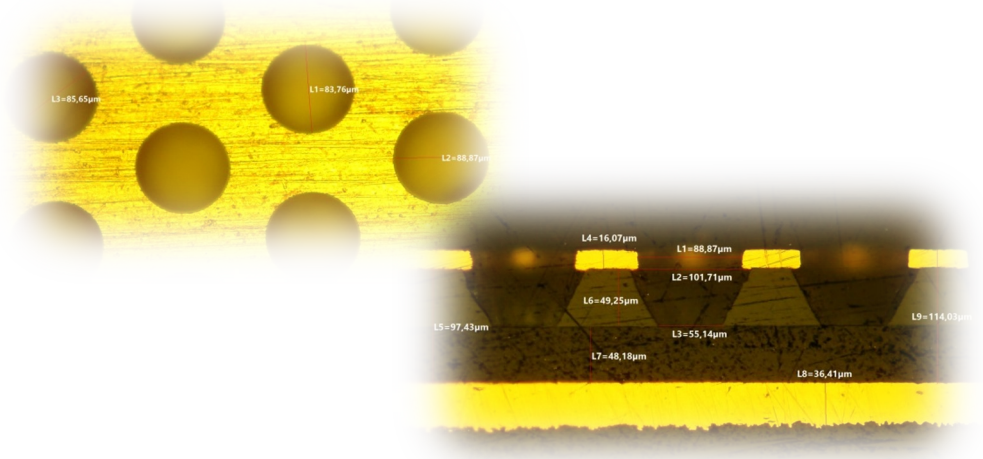
A one-month old Work in Progress,
There's a lot of hope, less results :)

APV25 integration in Corryvreckan



Elena Sidoretti
on behalf of UniRoma 2 & LNF-INFN

Thanks to the Corryvreckan Developers
https://cern.ch/corryvreckan

RD51 Collaboration Meeting, December 7th 2023

INFN LNF - DDG

TOR VERGATA
UNIVERSITÀ
DEGLI STUDI
DI ROMA

12/2023

https://indico.cern.ch/event/1058977/contributions/4924157/

Hands-On: Corryvreckan

The Maelstrom for Your Test Beam Data

10th BTTB Workshop

21th June 2022

Finn Feindt, DESY

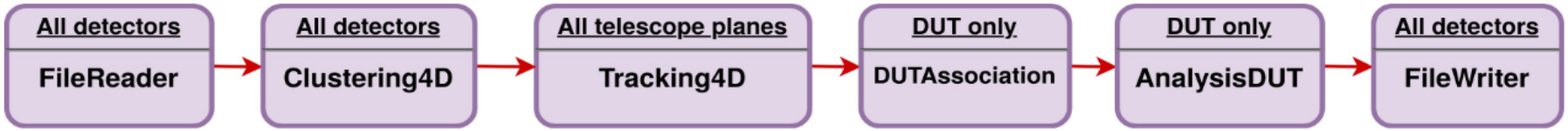
on behalf of the Corryvreckan Developers

Based on the work from: Jens Kröger, formerly Heidelberg University & CERN

The Modular Approach

- modular structure:
  - framework core
  - implementation of algorithms → **`[Modules]`**
- modules:
  (user) algorithms for specific tasks
- objects are stored temporarily:
  - events, pixels, clusters, tracks

- select suitable modules for
  - event building
  - clustering
  - tracking
  - analysis (also multiple DUTs)
  - ...
- quick to set up and easy to configure



The Modular Approach

- modular structure:
  - framework core
  - implementati[illegible]
- modules:
  (user) algorithms
- objects are stored temporarily:
  - events, pixels, clusters, tracks

- select suitable modules for
  - event building
  - clustering

[illegible]tiple DUTs)

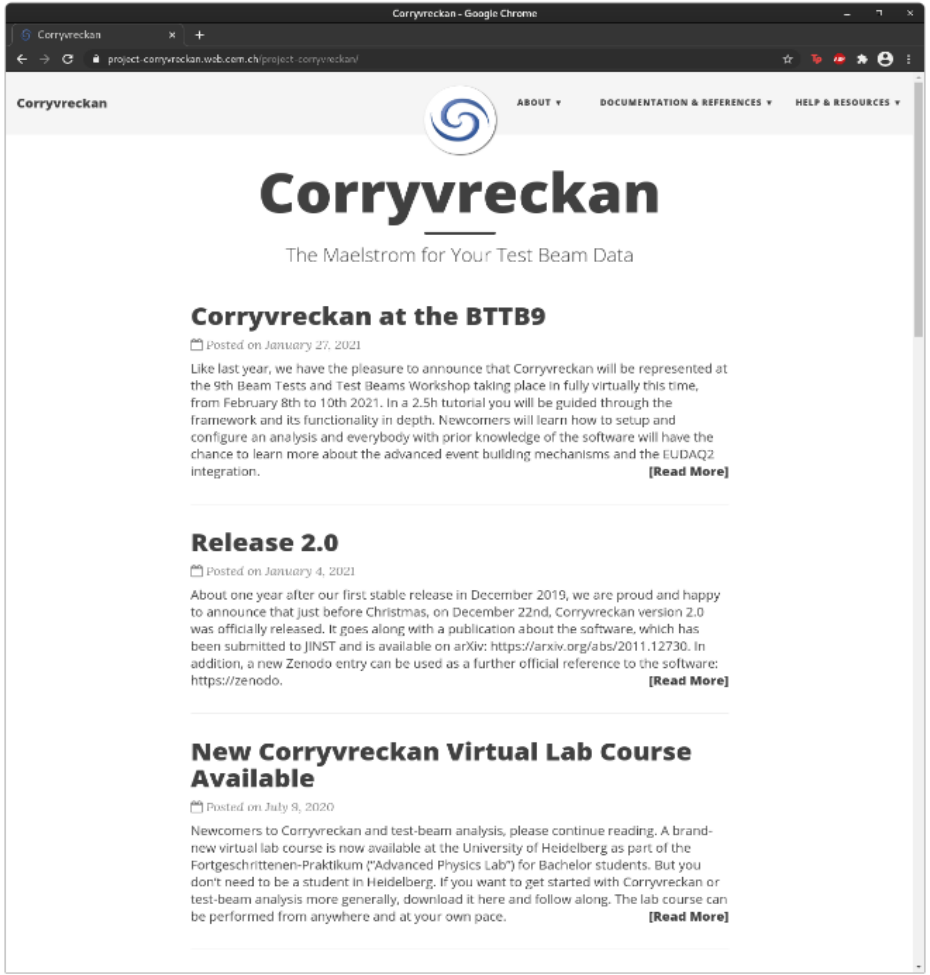
- quick to set up and easy to configure

Project Website

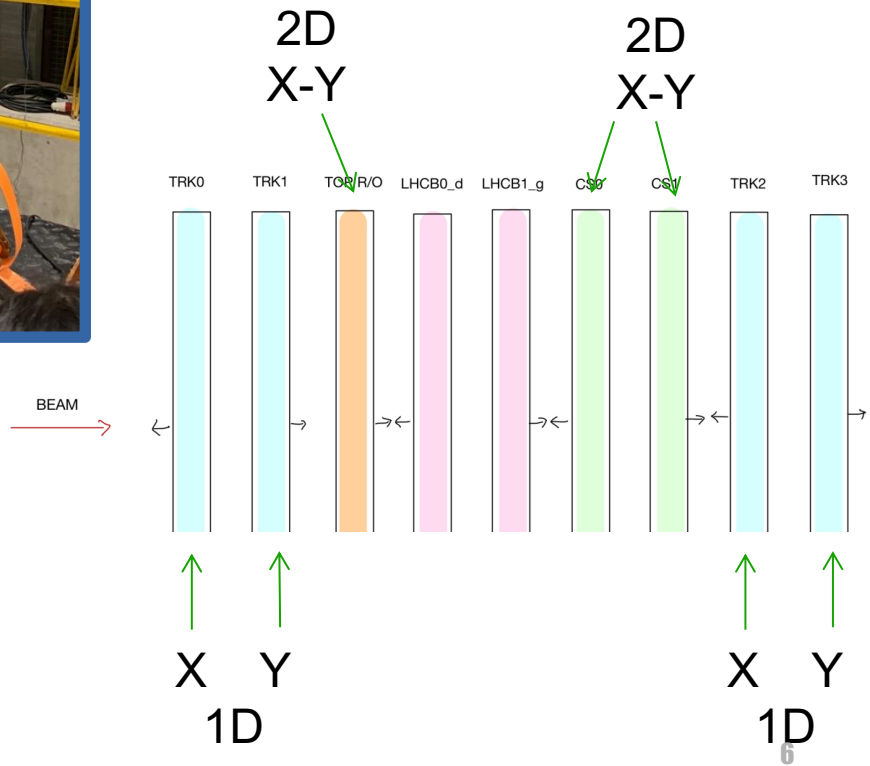
First place to go:

https://cern.ch/corryvreckan

- News on releases
- Installation/Getting Started
- **Links:**
  - code repository
  - issue tracker
  - forum
  - ...



Thanks to the Ferrara, Bologna & Torino INFN groups. See M.P.L. talk→https://indico.cern.ch/event/1327482/contributions/5706126/

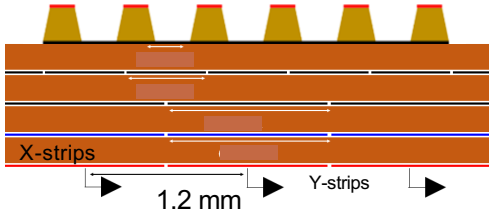


Possible 2D layout

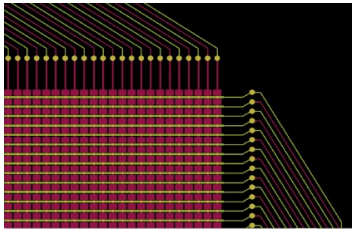
u-RWELL - Charge Sharing r/o (*)

Chatode

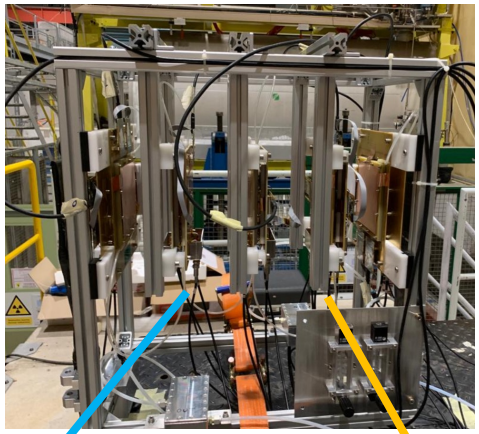
Drift gap



CS Readout board



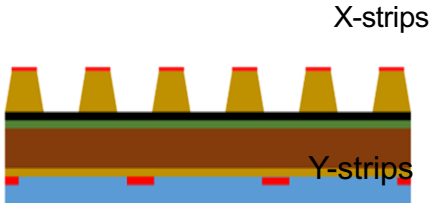
Active area= 100x100 mm2
Resistivity= 50 MΩ/□
Strip pitch= 1.2 mm
Strip width = 1.1 mm
Several layer between DLC and R/out



u-RWELL TOP r/o

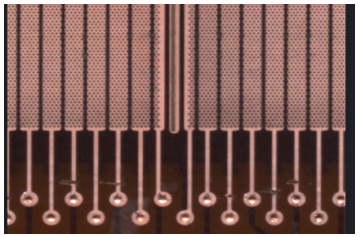
Chatode

Drift gap

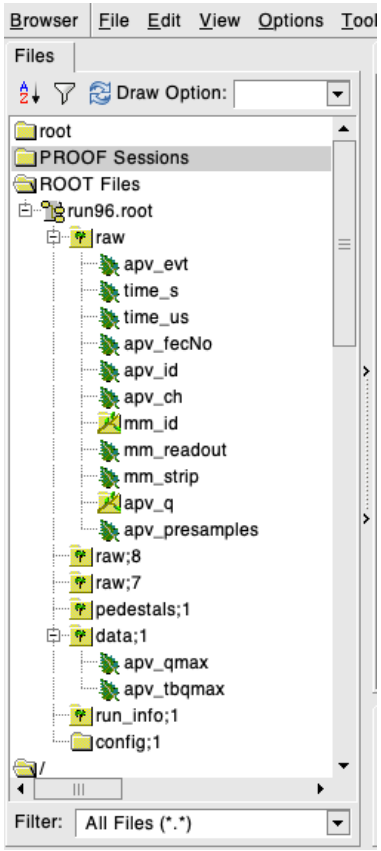


X coordinate on the TOP of the amplification stage

Active area= 100x100 mm2
Resistivity= 50 MΩ/□
Strip pitch= 0.8 mm
Strip width = 0.7 mm
Dead zone (TOP) ~ 15%
Pre-preg thickness= 70 um



What are we dealing with?



Starting from the SRS APV25.

The data are stored in the 'raw' tree:

- apv_fecNo = it is needed only when the APV's number is >16 (not our case so it's always = 1)
- apv_id = the ID for each APV FFE, here goes from 0 to 13
- apv_ch = channel of the APV, goes from 0 to 127
- mm_id = name of the detector
- mm_readou = orientation of the readout (x or other), in our case will be specified in the corryvrekan geometry file
- mm_strip = we don't consider this;
- apv_q = vector containg the waveform of the charge signal (sampled every 25 ns)
- apv_presamples = we don't consider this

SRS version 1.5.4

MMDAQ version 1

APVReader

I created a new Detector type module.

It concernes detector of type: APV25.

The APVReader.h:

- used to initialize the parameters

corryvreckan > src > modules > APVReader > C APVReader.h > ...

```
70
71      private:
72          // bool loadData(const std::shared_ptr<Clipboard>& clipboard, PixelVector&);
73
74          TTree* tree; //! pointer to the analyzed TTree or TChain
75
76          std::shared_ptr<Detector> m_detector;
77          // Member variables
78          int m_eventNumber;
79
80          std::string m_filename;
81          TFile* m_file;
82
83          TTreeReader* reader;
84          TTreeReaderArray<unsigned int>* apv_evt;
85
86          // Declaration of leaf types
87          // UInt_t apv_evt;
88          Int_t time_s;
89          Int_t time_us;
90          std::vector<unsigned int>* apv_fecNo;
91          std::vector<unsigned int>* apv_id;
92          std::vector<unsigned int>* apv_ch;
93          std::vector<std::string>* mm_id;
94          std::vector<unsigned int>* mm_readout;
95          std::vector<unsigned int>* mm_strip;
96          std::vector<std::vector<short>>* apv_q;
97          UInt_t apv_presamples;
98
99          // List of branches
100         // TBranch* b_apv_evt;        //!
101         TBranch* b_time_s;         //!
102         TBranch* b_time_us;        //!
103         TBranch* b_apv_fecNo;      //!
104         TBranch* b_apv_id;         //!
105         TBranch* b_apv_ch;         //!
106         TBranch* b_mm_id;          //!
107         TBranch* b_mm_readout;     //!
```

Ln 1, Col 1 Sp

APVReader

```
12  #include "APVReader.h"
13  // #pragma link C++ class std::vector<vector<short>>
14
15  using namespace corryvreckan;
16  using namespace std;
17
18  APVReader::APVReader(Configuration& config, std::shared_ptr<Detector>
    detector) : Module(config, detector) {
19      m_detector = detector;
20      config_.setDefault<std::string>("input_directory", ".root");
21      m_filename = config_.get<std::string>("input_directory");
22      // Take input directory from global parameters
23  }
24  APVReader::~APVReader() {}
25
```

Introducing:

- Detectors
- Input file

APVReader

Standard

```
26  void APVReader::initialize() {
27
28      // histograms to create
29      std::string title = m_detector->getName() + "hit map";
30      hHitMap = new TH2F("hitMap", title.c_str(), 129, -0.5, 129.5, 129, -0.05, 0.05);
31      title = m_detector->getName() + "charge";
32      hPixelToT = new TH1F("pixelToT", title.c_str(), 400, -0.5, 3000.5);
33      title = m_detector->getName() + "hits per channel";
34      hHits = new TH1F("Hits", title.c_str(), 129, -0.5, 129.5);
35
36      // Checking Geometric file inputs
37      for(auto& detector : get_detectors()) {
38          LOG(DEBUG) << "Initialise for detector " + detector->getName();
39      }
40
41      // Open the data file for later
42      m_file = TFile::Open(m_filename.c_str());
43      m_file->GetObject("raw", tree);
44      reader = new TTreeReader("raw", m_file);
45
46      m_eventNumber = 0;
47
48      // inizializzazione del tree
49      //  Set object pointer
50      apv_fecNo = 0;
51      apv_id = 0;
52      apv_ch = 0;
53      mm_id = 0;
54      mm_readout = 0;
55      mm_strip = 0;
56      apv_q = 0;
57      // Set branch addresses and branch pointers
58      if(!tree) {
59          return;
60      }
61
```

```
62      // tree->SetBranchAddress("apv_evt", &apv_evt, &b_apv_evt);
63      tree->SetBranchAddress("time_s", &time_s, &b_time_s);
64      tree->SetBranchAddress("time_us", &time_us, &b_time_us);
65      tree->SetBranchAddress("apv_fecNo", &apv_fecNo, &b_apv_fecNo);
66      tree->SetBranchAddress("apv_id", &apv_id, &b_apv_id);
67      tree->SetBranchAddress("apv_ch", &apv_ch, &b_apv_ch);
68      tree->SetBranchAddress("mm_id", &mm_id, &b_mm_id);
69      tree->SetBranchAddress("mm_readout", &mm_readout, &b_mm_readout);
70      tree->SetBranchAddress("mm_strip", &mm_strip, &b_mm_strip);
71      tree->SetBranchAddress("apv_q", &apv_q, &b_apv_q);
72      tree->SetBranchAddress("apv_presamples", &apv_presamples, &b_apv_presamples);
73
74      apv_evt = new TTreeReaderArray<unsigned int>(*reader, "apv_evt");
75  }
```

Getting the “raw” tree in the input file

Initialize the pointers (not sure if needed for real)

APVReader

```
26  void APVReader::initialize() {
27
28      // histograms to create
29      std::string title = m_detector->getName() + "hit map";
30      hHitMap = new TH2F("hitMap", title.c_str(), 129, -0.5, 129.5, 129, -0.05, 0.05);
31      title = m_detector->getName() + "charge";
32      hPixelToT = new TH1F("pixelToT", title.c_str(), 400, -0.5, 3000.5);
33      title = m_detector->getName() + "hits per channel";
34      hHits = new TH1F("Hits", title.c_str(), 129, -0.5, 129.5);
35
36      // Checking Geometric file inputs
37      for(auto& detector : get_detectors()) {
38          LOG(DEBUG) << "Initialise for detector " + detector->getName();
39      }
40
41      // Open the data file for later
42      m_file = TFile::Open(m_filename.c_str());
43      m_file->GetObject("raw", tree);
44      reader = new TTreeReader("raw", m_file);
45
46      m_eventNumber = 0;
47
48      // inizializzazione del tree
49      //  Set object pointer
50      apv_fecNo = 0;
51      apv_id = 0;
52      apv_ch = 0;
53      mm_id = 0;
54      mm_readout = 0;
55      mm_strip = 0;
56      apv_q = 0;
57      // Set branch addresses and branch pointers
58      if(!tree) {
59          return;
60      }
61
```

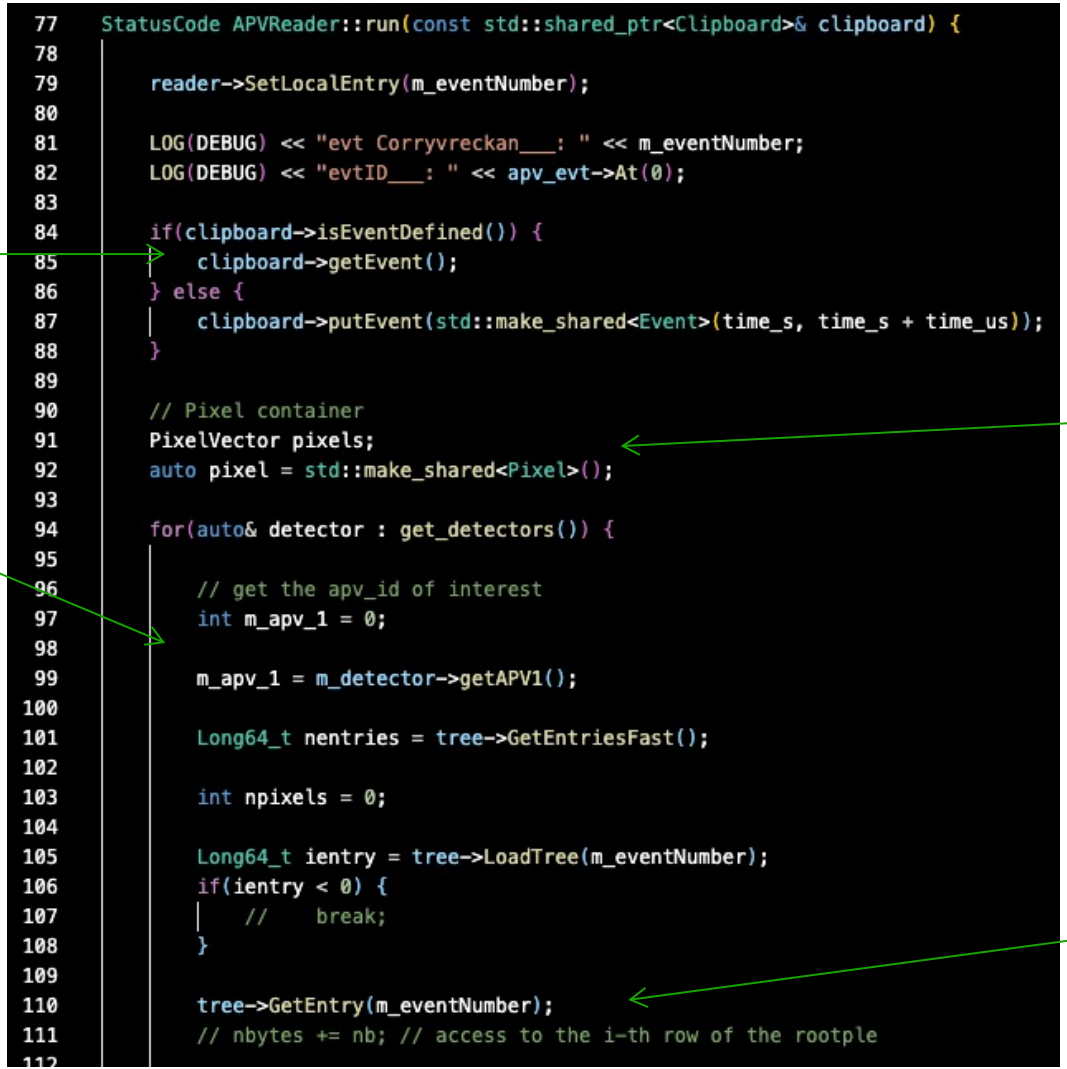
```
62      // tree->SetBranchAddress("apv_evt", &apv_evt, &b_apv_evt);
63      tree->SetBranchAddress("time_s", &time_s, &b_time_s);
64      tree->SetBranchAddress("time_us", &time_us, &b_time_us);
65      tree->SetBranchAddress("apv_fecNo", &apv_fecNo, &b_apv_fecNo);
66      tree->SetBranchAddress("apv_id", &apv_id, &b_apv_id);
67      tree->SetBranchAddress("apv_ch", &apv_ch, &b_apv_ch);
68      tree->SetBranchAddress("mm_id", &mm_id, &b_mm_id);
69      tree->SetBranchAddress("mm_readout", &mm_readout, &b_mm_readout);
70      tree->SetBranchAddress("mm_strip", &mm_strip, &b_mm_strip);
71      tree->SetBranchAddress("apv_q", &apv_q, &b_apv_q);
72      tree->SetBranchAddress("apv_presamples", &apv_presamples, &b_apv_presamples);
73
74      apv_evt = new TTreeReaderArray<unsigned int>(*reader, "apv_evt");
75  }
```

Setting branch addresses for the given tree

APVReader

Getting the event

Getting APV number from the geometry file



```
77  StatusCode APVReader::run(const std::shared_ptr<Clipboard>& clipboard) {
78
79      reader->SetLocalEntry(m_eventNumber);
80
81      LOG(DEBUG) << "evt Corryvreckan___: " << m_eventNumber;
82      LOG(DEBUG) << "evtID___: " << apv_evt->At(0);
83
84      if(clipboard->isEventDefined()) {
85          clipboard->getEvent();
86      } else {
87          clipboard->putEvent(std::make_shared<Event>(time_s, time_s + time_us));
88      }
89
90      // Pixel container
91      PixelVector pixels;
92      auto pixel = std::make_shared<Pixel>();
93
94      for(auto& detector : get_detectors()) {
95
96          // get the apv_id of interest
97          int m_apv_1 = 0;
98
99          m_apv_1 = m_detector->getAPV1();
100
101         Long64_t nentries = tree->GetEntriesFast();
102
103         int npixels = 0;
104
105         Long64_t ientry = tree->LoadTree(m_eventNumber);
106         if(ientry < 0) {
107             //    break;
108         }
109
110         tree->GetEntry(m_eventNumber);
111         // nbytes += nb; // access to the i-th row of the rootple
112
```

Pixel container

Get in the tree

APVReader

For each APV_channel with a signal, we have to assign a pixel

Put it on the clipboard

```
113         for(int j = 0; j < apv_id->size(); j++) {
114
115             if((*apv_id)[j] == m_apv_1) {
116                 int column = ((*apv_ch)[j]);
117                 int max_q = 0;
118                 max_q = *std::max_element(((*apv_q)[j]).begin(), ((*apv_q)[j]).end());
119
120                 /*/ If this pixel is masked, do not save it
121                 if(m_detector->masked(column, 0)) {
122                     continue;
123                 }*/
124                 if(max_q > 50) {
125                     pixel = std::make_shared<Pixel>(m_detector->getName(), column, 0, max_q, max_q,
                        (time_s));
126
127                     hHitMap->Fill(pixel->column(), pixel->row());
128                     hPixelToT->Fill(pixel->raw());
129                     hHits->Fill(pixel->column());
130
131                     pixels.push_back(pixel);
132                 }
133
134                 npixels++;
135             }
136         }
137         clipboard->putData(pixels, m_detector->getName());
138         LOG(TRACE) << "Loaded " << npixels << " pixels";
139     }
140
141     m_eventNumber++;
142
143     if(reader->GetEntries() == m_eventNumber)
144         return StatusCode::EndRun;
145     return StatusCode::Success;
146 }
```

```
148 void APVReader::finalize(const std::shared_ptr<ReadonlyClipboard>&) {
149     LOG(DEBUG) << "Analysed " << m_eventNumber << " events";
150 }
151
```

APVReader – Config and geometry files

TB_1D_APV.conf

```
[Corryvreckan]
#log_level = "DEBUG"

output_directory = "output"
detectors_file = "geom_TB_1D.geo"
histogram_file = "analysis_TB_APV_run96.root"

number_of_events = -1


[APVReader]
input_directory = "/home/elena/Scrivania/corry_TB/run/run96.root"


[Clustering4D]
charge_weighting = true

```

geom_TB_1D.geo

```
[APV_0]
number_of_pixels = 128,1
orientation = 0deg,0deg,90deg
orientation_mode = "xyz"
pixel_pitch = 400um,10cm
position = 0cm,0cm,0mm
spatial_resolution = 100um,100cm
time_resolution = 10ns
apv_1 = 0
role ="reference"
type = "APV25"

[APV_1]
number_of_pixels = 128,1
orientation = 0deg,0deg,90deg
orientation_mode = "xyz"
pixel_pitch = 400um,10cm
position = 0cm,0cm,1mm
spatial_resolution = 100um,100cm
time_resolution = 10ns
apv_1 = 1
type = "APV25"

```

APVReader – Config and geometry files

TB_1D_APV.conf

```
[Corryvreckan]
#log_level = "DEBUG"

output_directory = "output"
detectors_file = "geom_TB_1D.geo"
histogram_file = "analysis_TB_APV_run96.root"

number_of_events = -1


[APVReader]
input_directory = "/home/elena/Scrivania/corry_TB/run/run96.root"


[Clustering4D]
charge_weighting = true

```

geom_TB_1D.geo

```
[APV_0]
number_of_pixels = 128,1
orientation = 0deg,0deg,90deg
orientation_mode = "xyz"
pixel_pitch = 400um,10cm
position = 0cm,0cm,0mm
spatial_resolution = 100um,100cm
time_resolution = 10ns
apv_1 = 0
role ="reference"
type = "APV25"

[APV_1]
number_of_pixels = 128,1
orientation = 0deg,0deg,90deg
orientation_mode = "xyz"
pixel_pitch = 400um,10cm
position = 0cm,0cm,1mm
spatial_resolution = 100um,100cm
time_resolution = 10ns
apv_1 = 1
type = "APV25"

```

Reminder: Corryvreckan reads every R/O as pixels

APVReader – Config and geometry files

TB_1D_APV.conf

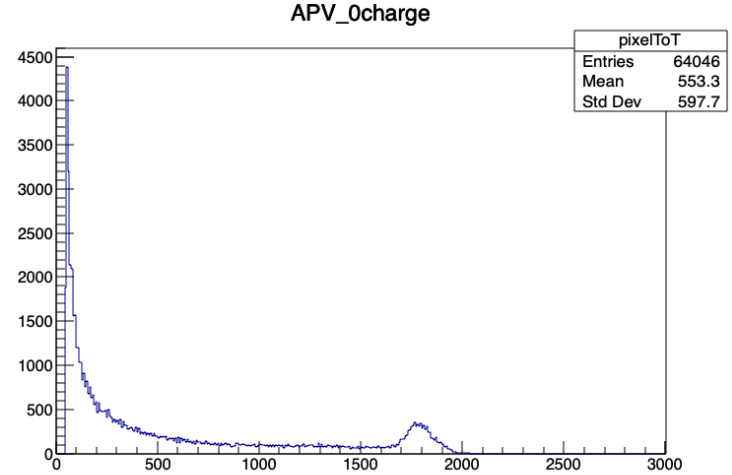
```
[Corryvreckan]
#log_level = "DEBUG"

output_directory = "output"
detectors_file = "geom_TB_1D.geo"
histogram_file = "analysis_TB_APV_run96.root"

number_of_events = -1


[APVReader]
input_directory = "/home/elena/Scrivania/corry_TB/run/run96.root"


[Clustering4D]
charge_weighting = true
```

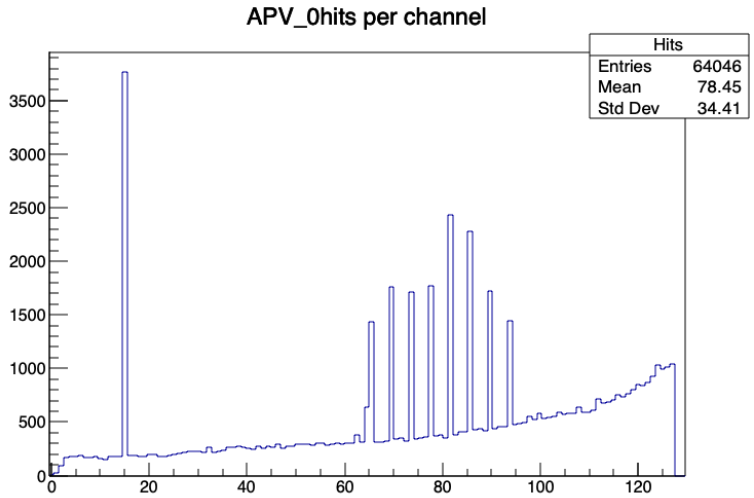
geom_TB_1D.geo

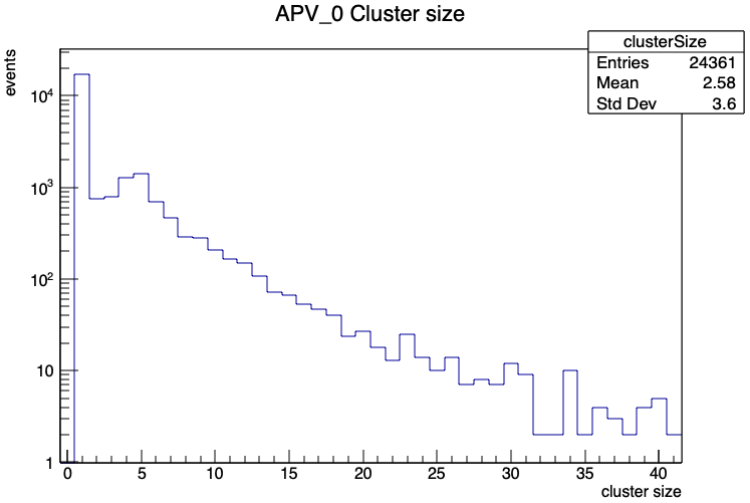
```
[APV_0]
number_of_pixels = 128,1
orientation = 0deg,0deg,90deg
orientation_mode = "xyz"
pixel_pitch = 400um,10cm
position = 0cm,0cm,0mm
spatial_resolution = 100um,100cm
time_resolution = 10ns
apv_1 = 0
type = "APV25"

[APV_1]
number_of_pixels = 128,1
orientation = 0deg,0deg,90deg
orientation_mode = "xyz"
pixel_pitch = 400um,10cm
position = 0cm,0cm,1mm
spatial_resolution = 100um,100cm
time_resolution = 10ns
apv_1 = 1
type = "APV25"
```

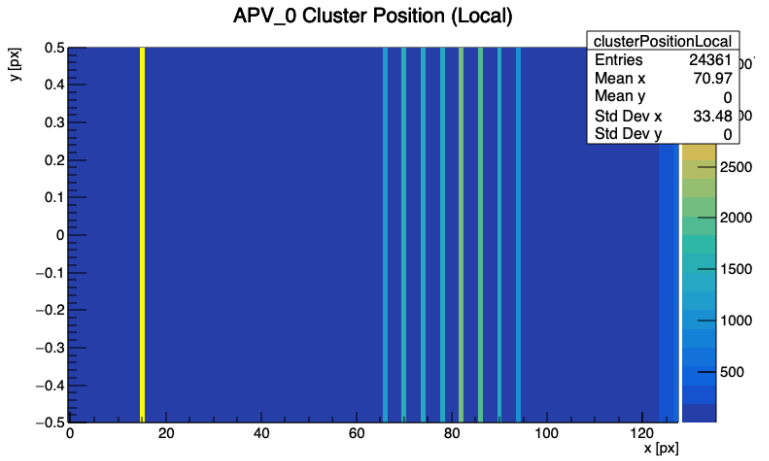
"apv_1" is the parameter indicating the APV reading the detector. If a second APV is used, it will be labelled as "apv_2".

APVReader – preliminary results

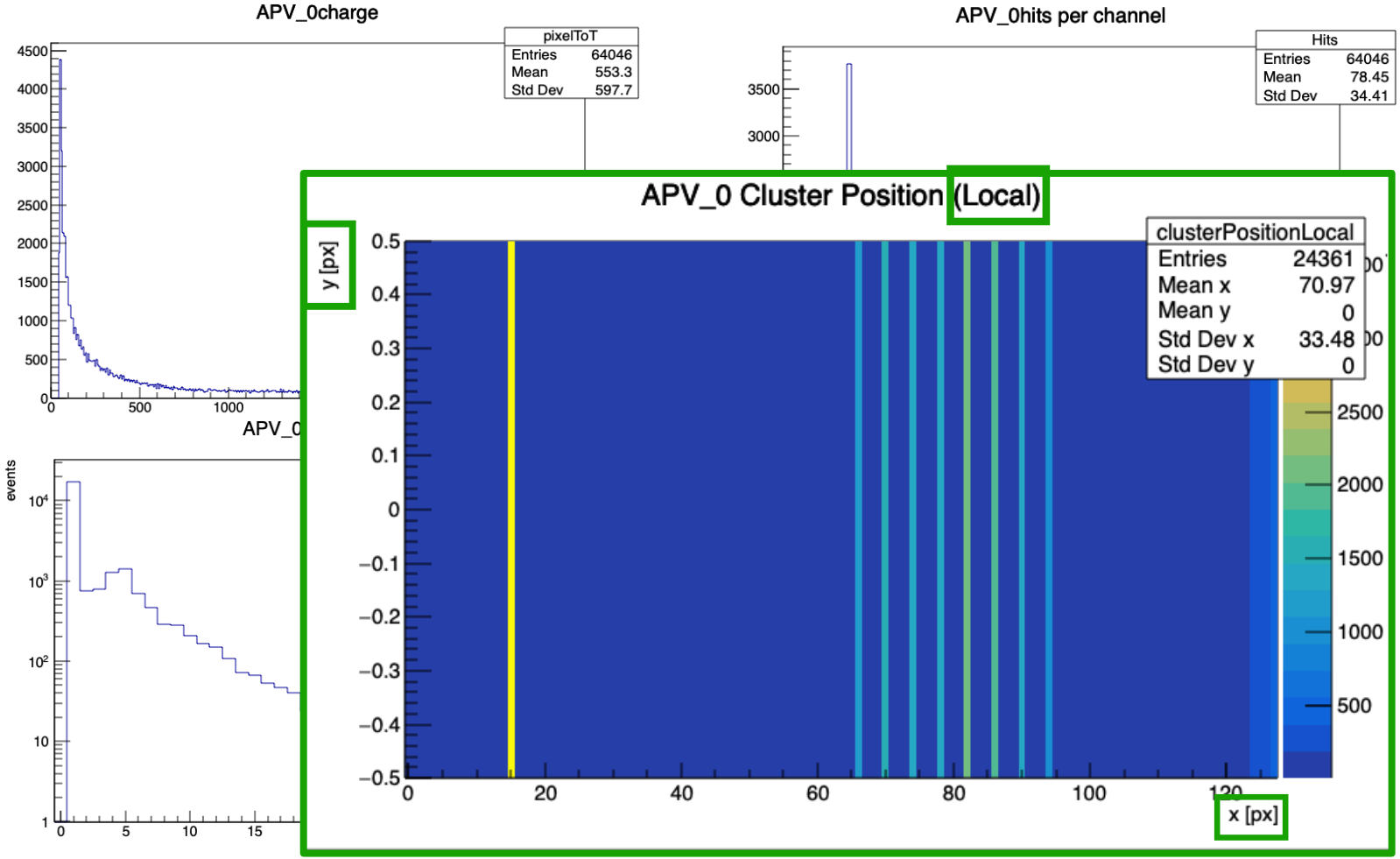








APVReader – preliminary results



APVReader – 2D

```
111     int hits = apv_id->size();
112     int row[hits];
113     int column[hits];
114     int max_q[hits];
115     int d = 0;
116     while(d < hits) {
117         row[d] = -99999;
118         column[d] = -99999;
119         max_q[d] = 0;
120         d++;
121     }
122
123     for(int j = 0; j < hits; j++) {
124
125         if((*apv_id)[j] == m_apv_1 || (*apv_id)[j] == m_apv_2) {
126             if(((*apv_id)[j] == m_apv_1)) {
127                 column[j] = ((*apv_ch)[j]);
128             }
129
130             max_q[j] = *std::max_element(((*apv_q)[j]).begin(), ((*apv_q)[j]).end());
131
132             if(((*apv_id)[j] == (m_apv_2))) {
133                 row[j] = ((*apv_ch)[j]);
134             }
135         }
136     }
```

```
138     for(int l = 0; l < hits; l++) {
139
140         for(int n = 0; n < hits; n++) {
141
142             if(column[l] >= 0 && row[n] >= 0 && max_q[l]>50) {
143                 // If this pixel is masked, do not save it
144                 if(m_detector->masked(column[l], row[n])) {
145                     continue;
146                 }
147                 pixel = std::make_shared<Pixel>(m_detector->getName(), column[l], row[n], max_q[l],
                    max_q[l], (time_s));
148
149                 pixels.push_back(pixel);
150
151                 hHitMap->Fill(pixel->column(), pixel->row());
152                 hPixelToT->Fill(pixel->raw());
153
154                 npixels++;
155             }
156         }
157
158         // }
159     }
```

We get the pixel's column from the APV reading the "x" and the row from the "y" one.

APVReader – 2D

```
111     int hits = apv_id->size();
112     int row[hits];
113     int column[hits];
114     int max_q[hits];
115     int d = 0;
116     while(d < hits) {
117         row[d] = -99999;
118         column[d] = -99999;
119         max_q[d] = 0;
120         d++;
121     }
122
123     for(int j = 0; j < hits; j++) {
124
125         if((*apv_id)[j] == m_apv_1 || (*apv_id)[j] == m_apv_2) {
126             if(((*apv_id)[j] == m_apv_1)) {
127                 column[j] = ((*apv_ch)[j]);
128             }
129
130             max_q[j] = *std::max_element(((*apv_q)[j]).begin(), ((*apv_q)[j]).end());
131
132             if(((*apv_id)[j] == (m_apv_2))) {
133                 row[j] = ((*apv_ch)[j]);
134             }
135         }
136     }
```

```
138     for(int l = 0; l < hits; l++) {
139
140         for(int n = 0; n < hits; n++) {
141
142             if(column[l] >= 0 && row[n] >= 0 && max_q[l]>50) {
143                 // If this pixel is masked, do not save it
144                 if(m_detector->masked(column[l], row[n])) {
145                     continue;
146                 }
147                 pixel = std::make_shared<Pixel>(m_detector->getName(), column[l], row[n], max_q[l],
                    max_q[l], (time_s));
148
149                 pixels.push_back(pixel);
150
151                 hHitMap->Fill(pixel->column(), pixel->row());
152                 hPixelToT->Fill(pixel->raw());
153
154                 npixels++;
155             }
156         }
157
158         // }
159     }
```

The pixel in defined when both “x” and ”y” fire.

APVReader – tracker situation: 1D to 2D

The column of the is defined by the detector with strips along the “x” direction

```
138     for(int j = 0; j < hits; j++) {
139
140         if((*apv_id)[j] == m_apv_1 || (*apv_id)[j] == m_apv_2 ||
141            ((*apv_id)[j] == (m_apv_1 + 2) || (*apv_id)[j] == (m_apv_2 + 2))) {
142
143             if(((*apv_id)[j] == m_apv_1)) {
144                 column[j] = ((*apv_ch)[j]);
145                 max_q_x[j] = *std::max_element(((*apv_q)[j]).begin(), ((*apv_q)[j]).end());
146             } else if(((*apv_id)[j] == m_apv_2)) {
147                 column[j] = (((*apv_ch)[j]) + 128);
148                 max_q_x[j] = *std::max_element(((*apv_q)[j]).begin(), ((*apv_q)[j]).end());
149             }
150
151             if(((*apv_id)[j] == m_apv_1 + 2)) {
152                 row[j] = ((*apv_ch)[j]);
153                 max_q_y[j] = *std::max_element(((*apv_q)[j]).begin(), ((*apv_q)[j]).end());
154             } else if(((*apv_id)[j] == m_apv_2 + 2)) {
155                 row[j] = (((*apv_ch)[j]) + 128);
156                 max_q_y[j] = *std::max_element(((*apv_q)[j]).begin(), ((*apv_q)[j]).end());
157             }
158
159
160             max_q[j] = (max_q_x[j] + max_q_y[j]) / 2.;
161         }
162     }
163
```

The row of the is defined by the detector with strips along the “y” direction

APVReader – tracker situation: 1D to 2D

```
164     for(int l = 0; l < hits; l++) {
165
166         for(int n = 0; n < hits; n++) {
167
168             if(column[l] >= 0 && row[n] >= 0 && max_q[l] > 50) {
169                 // cout << row[n] << endl;
170
171                 // If this pixel is masked, do not save it
172                 if(m_detector->masked(column[l], row[n])) {
173                     continue;
174                 }
175
176                 pixel = std::make_shared<Pixel>(m_detector->getName(), column[l], row[n], max_q[l],
                    max_q[l], (time_s));
177
178                 pixels.push_back(pixel);
179
180                 hHitMap->Fill(pixel->column(), pixel->row());
181                 hPixelToT->Fill(pixel->raw());
182
183                 npixels++;
184             }
185         }
186
187         // }
188     }
189
```

APVReader – Config file

This module will read the “x” and “y” 1D detectors as one “virtual” 2D detector.

TB_analysis.conf

```
[Corryvreckan]

output_directory = "output"
detectors_file = "geom_TB (copia).geo"
histogram_file = "analysis_TB_run96_long.root"

number_of_events = -1

[APV1Dto2D]
input_directory = "/home/elena/Scrivania/corry_TB/run/run96.root"


[APVReader2D]
input_directory = "/home/elena/Scrivania/corry_TB/run/run96.root"

[Clustering4D]
charge_weighting = true


[Tracking4D]
min_hits_on_track = 2
require_detectors = "TRK_IN","TRK_OUT"

```

This module will read the 2D detectors.

APVReader – Config file

TB_analysis.conf

```
[Corryvreckan]

output_directory = "output"
detectors_file = "geom_TB (copia).geo"
histogram_file = "analysis_TB_run96_long.root"

number_of_events = -1

[APV1Dto2D]
input_directory = "/home/elena/Scrivania/corry_TB/run/run96.root"


[APVReader2D]
input_directory = "/home/elena/Scrivania/corry_TB/run/run96.root"

[Clustering4D]
charge_weighting = true


[Tracking4D]
min_hits_on_track = 2
require_detectors = "TRK_IN","TRK_OUT"

```

In this case, we can perform the Tracking!

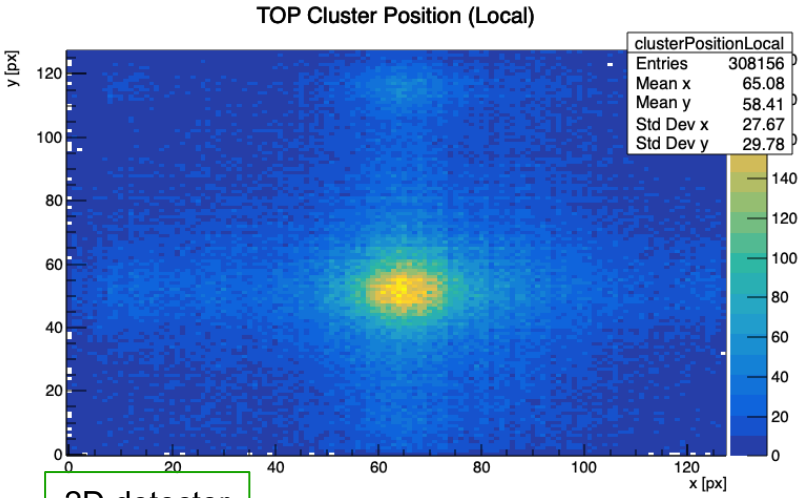
APVReader – Geometry file

“virtual” detector

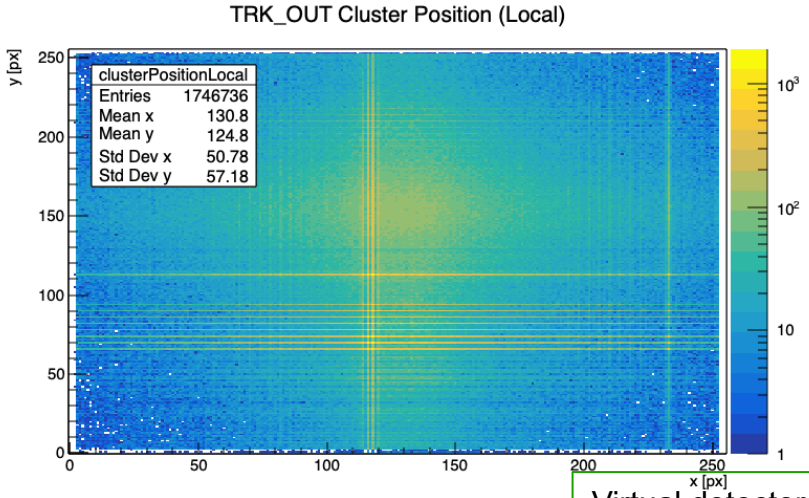
```
14  [TRK_OUT]
15  number_of_pixels = 256,256
16  orientation = 0deg,0deg,0deg
17  orientation_mode = "xyz"
18  pixel_pitch = 400um,400um
19  position = 0cm,0cm,530mm
20  spatial_resolution = 100um,100um
21  time_resolution = 10ns
22  apv_1 = 4
23  apv_2 = 5
24  type = "Virtual"
25
26  [CS0]
27  mask_file = "mask_files/mask_CS.conf"
28  number_of_pixels = 128,128
29  orientation = 0deg,0deg,0deg
30  orientation_mode = "xyz"
31  pixel_pitch = 1.2mm,1.2mm
32  position = 0cm,0cm,408mm
33  spatial_resolution = 100um,100um
34  time_resolution = 10ns
35  apv_1 = 10
36  apv_2 = 11
37  role = "dut"
38  type = "APV252D"
39
```

2D detector

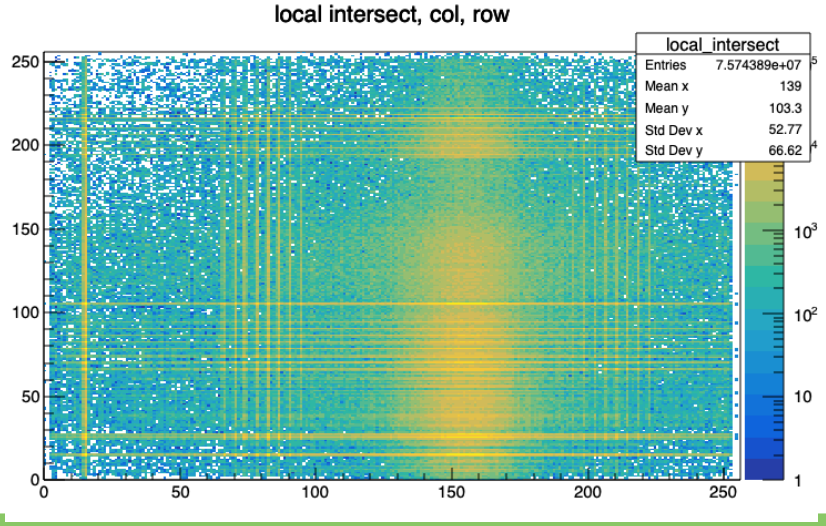
APVReader – preliminary results



2D detector



Virtual detector

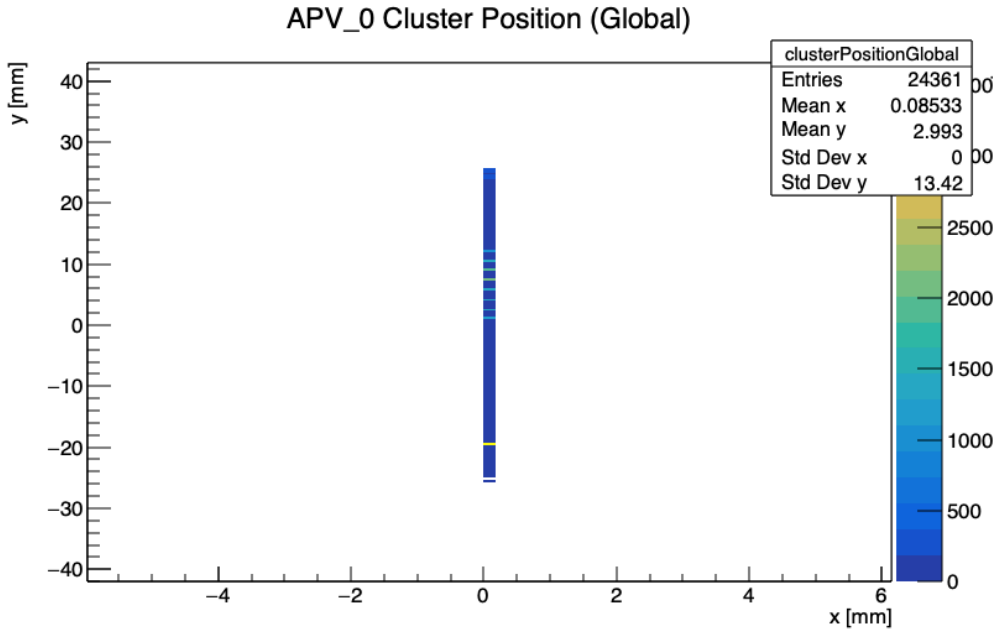


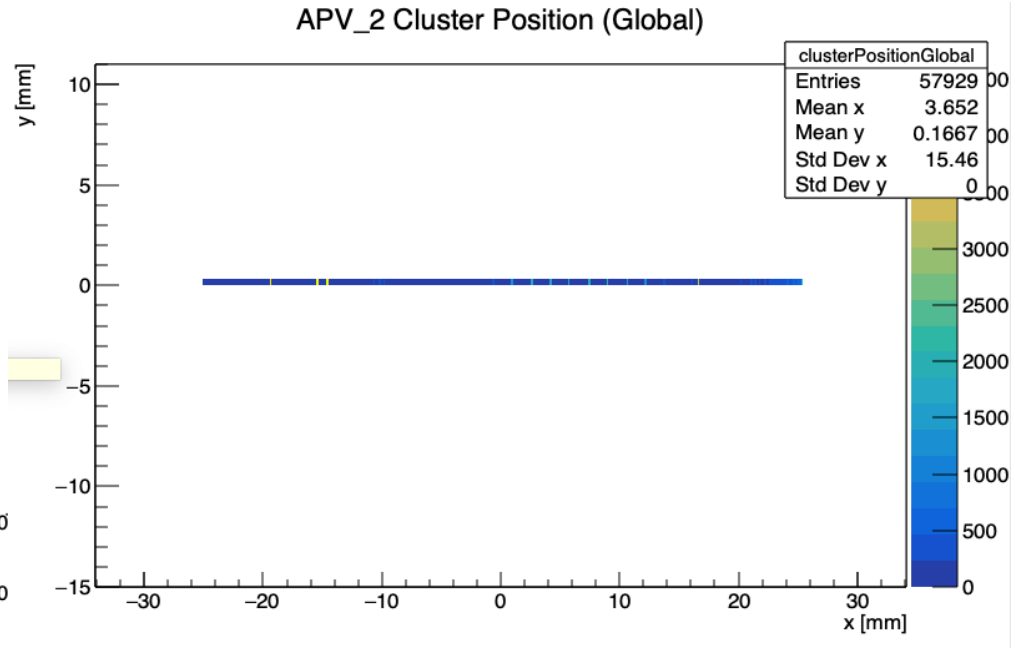
2D detector


"That's all Folks!"
(for now)

Spare


RD51
INFN
LNF - DDG
TOR VERGATA
UNIVERSITÀ
DEGLI STUDI
DI ROMA


APV_0 Cluster Position (Global)
clusterPositionGlobal
Entries 24361
Mean x 0.08533
Mean y 2.993
Std Dev x 0
Std Dev y 13.42
y [mm]
x [mm]


APV_2 Cluster Position (Global)
clusterPositionGlobal
Entries 57929
Mean x 3.652
Mean y 0.1667
Std Dev x 15.46
Std Dev y 0
y [mm]
x [mm]

Corryvreckan – make it APV 25 ready

How I modified the «Detector.cpp» script:

```
m_apv_1 = config.get<int>("apv_1",0); //ELENA
m_apv_2 = config.get<int>("apv_2",0); //ELENA
```

```
int Detector::getAPV1() const { //ELENA
    return m_apv_1;
}
int Detector::getAPV2() const { //ELENA
    return m_apv_2;
}
```

```
config.set("apv_1", m_apv_1); //ELENA
config.set("apv_2", m_apv_2); //ELENA
```

How I modified the «Detector.hpp» script:

```
int getAPV1() const;
    /** ELENA
*/
int getAPV2() const;
```

These will be used in the file defining the geometry of our setup.

Nome	R/O	Pre-preg thickness (um)
TRK0	STRIP (P 0.4 mm – W 0.3 mm)	100
TRK1	STRIP (P 0.4 mm – W 0.3 mm)	100
TOP R/O (TOP_READOUT_3)	STRIP (P 0.76 mm – W 0.3 mm)	25+12+25
LHCB0_d (PEP_DOT_3)	PAD	25
LHCB1_g (PEP_GROOVE_7)	PAD	50
CS0 (CS_4)	STRIP (P 1.2 mm – W 1.1 mm)	4 layers (25+12)
CS1(CS_3)	STRIP (P 1.2 mm – W 1.1 mm)	4 layers (25+12)
TRK2	STRIP (P 0.4 mm – W 0.3 mm)	100
TRK3	STRIP (P 0.4 mm – W 0.3 mm)	100