i2DBSCAN: an iterative density-based clustering algorithm for CYGNO experiment

Igor Abritta Costa on behalf of CYGNO

UFJF

INFN

07-11-2019

Introduction

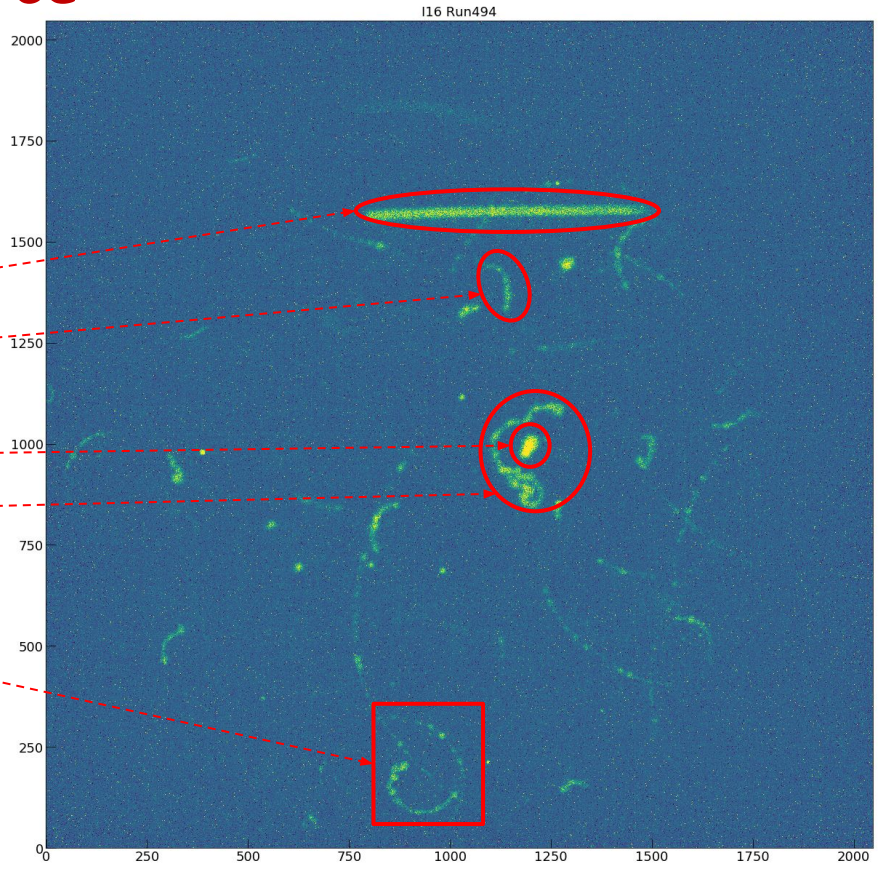
- CYGNO experiment aims to develop a gas TPC instrumented with a triple-GEM structure read-out by a high-resolution CMOS sensor;
- The goal of the experiment is to search for Dark Matter massive particles;
- In order to do that the event reconstruction algorithm should extract the relevant information from the output image and then be able to identify the types of signals.

- This work present the development of the clustering algorithm for CYGNO experiment and some results.

Example of data for AmBe Source

This image is an example of what we can have using a specific source and what we need to identify:

- ❏ Brighter and long tracks;
- ❏ Lighter tracks;
- ❏ Brighter and rounded tracks;
- ❏ Close tracks;
- ❏ Overlapped tracks;
- ❏ etc..



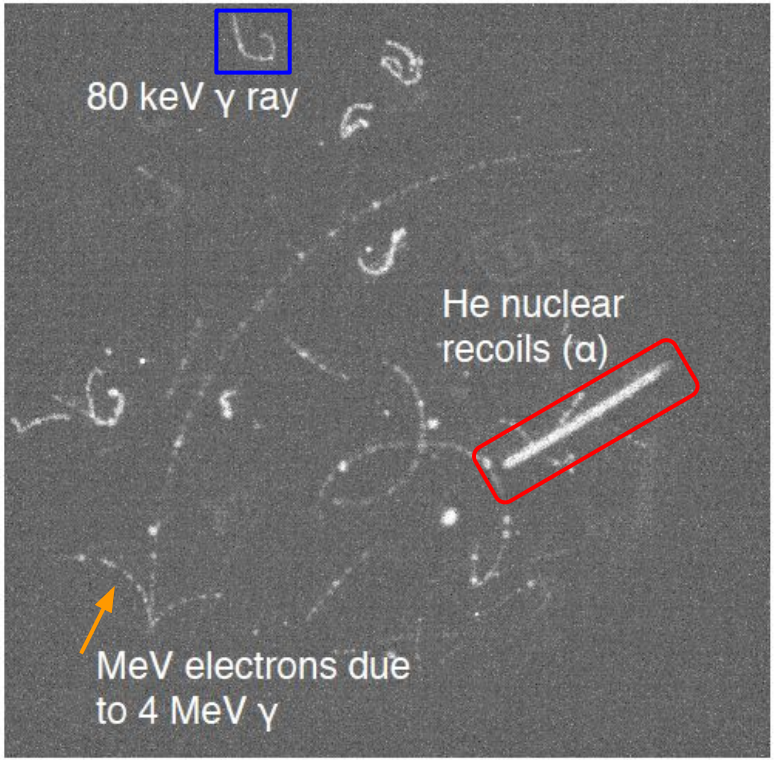
The conditions are very different from the real experiment, where it is expect only few natural radioactivity background and the signal which are **short** and **curvy** tracks.

Setup used to took this data

- ❏ Took using the ORANGE detector;
- ❏ AmBe Neutron source

Using this configuration we expect to see three types of signals:

- He nuclear recoils (α);
- Low energy electrons due to X rays;
- MeV electrons due to 4 MeV γ.



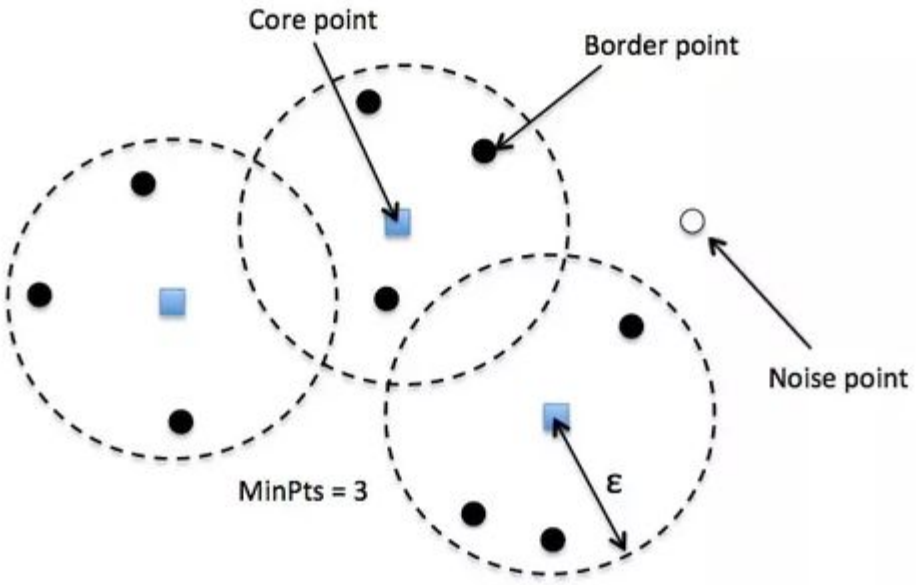
DBSCAN - Density based clustering algorithm

DBSCAN divides a dataset into subgroups of high density regions using two parameters: epsilon (**ε**) and minimum amount of points required to form a cluster (**minPts**).

Core point – a point that has at least a minimum number of other points (minPts) within its ε radius;

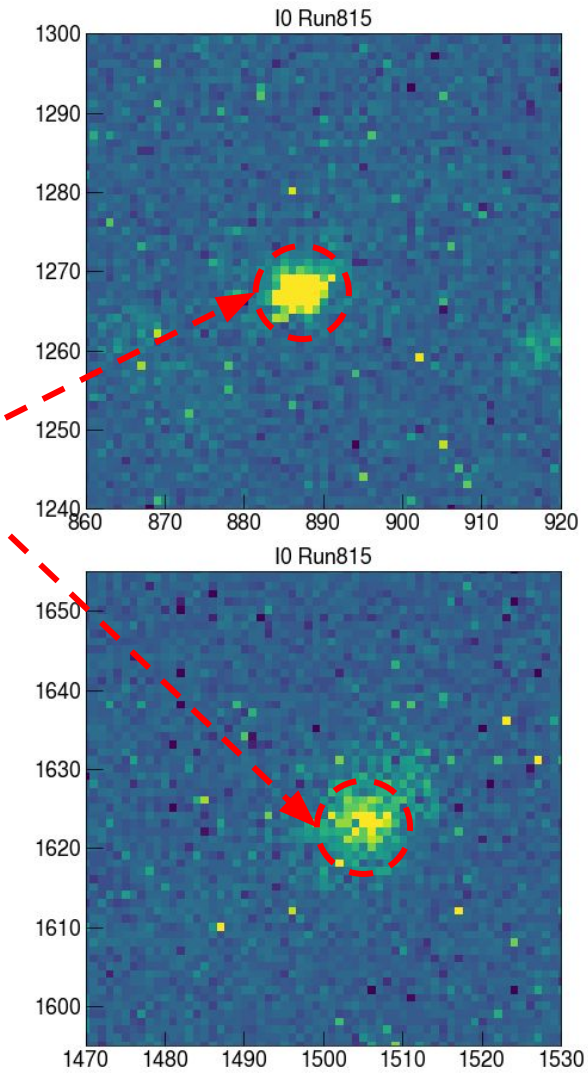
Border point – a point is within the ε radius of a core point BUT has less than the minimum number of other points (minPts) within its own ε radius;

Noise point – a point that is neither a core point or a border point.



Drawbacks of DBSCAN

- It is not simple to set the parameters (**ε**, minPts);
- DBSCAN uses as an input the coordinates of the pixels that it should cluster, so these two clusters (after a threshold and using only the coordinates of X and Y as information) could be seen by the algorithm as similar. And if they are close, the algorithm could cluster them together;
- As DBSCAN looks for proximity on the space, it is knew that increasing the dimensions could lead us to the problem called ‘curse of dimensionality’.



Proposed improvements on the clustering algorithm

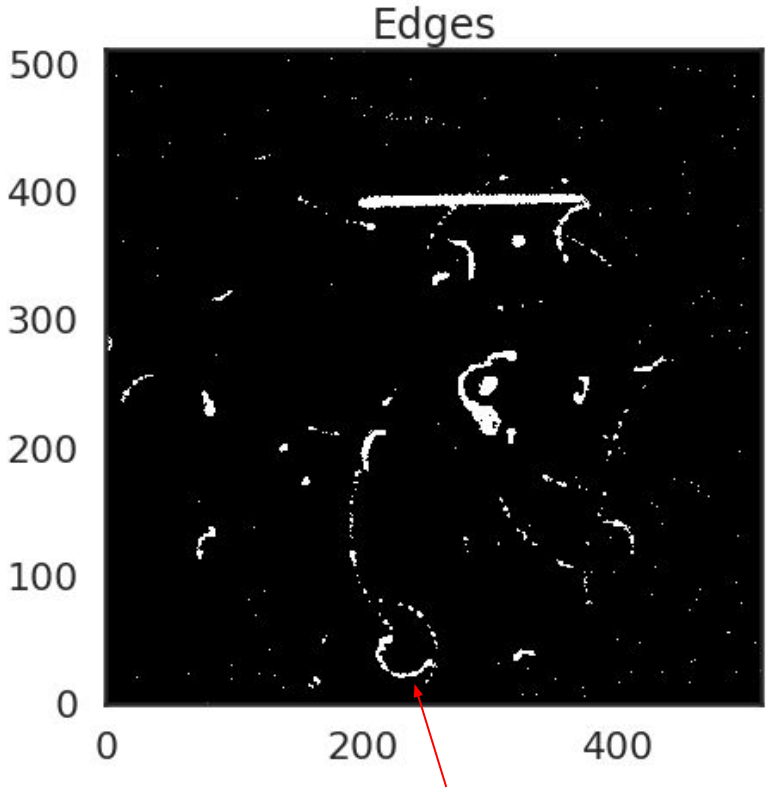
In order to avoid the highlighted drawbacks an modification of DBSCAN called i2DBSCAN is proposed:

- First, the idea is improving the detection of clustering with different sparsity of pixels running DBSCAN more than one time with different set of parameters;
- Second, get the coordinate of the pixel (x,y) that passed throught the threshold and replicate this coordinate by the number of z. In this way will be possible to ‘simulate’ the third dimension without have to lead with the ‘curse of dimensionality’;

Iterative DBSCAN method

1. Run DBSCAN on a image to look to ‘noise’ clusters and remove them from the image;
2. Search first for tracks with **high** density of pixels;
   a. Remove them from the image;
3. Search first for tracks with **medium** density of pixels;
   a. Remove them from the image;
4. Search for others tracks;
5. Go to next image.

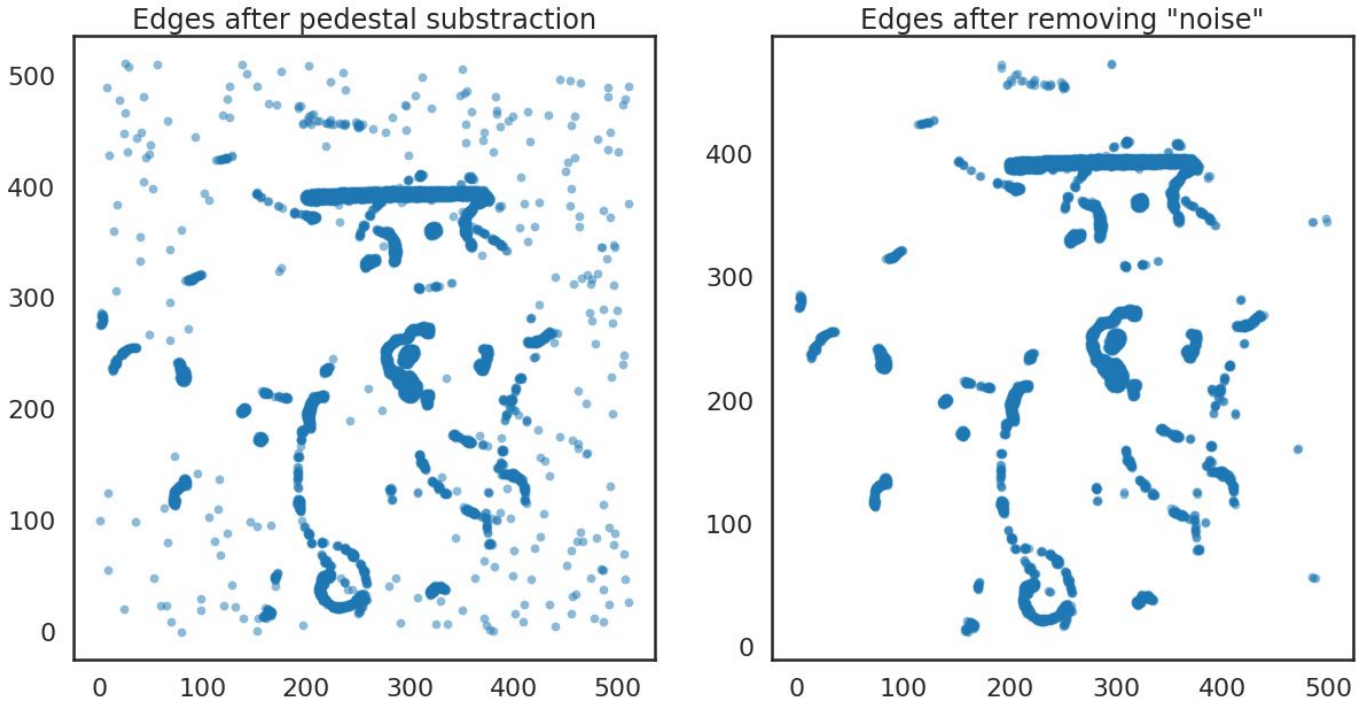
These steps can be done for (x,y) or (x,y,z).



Theses X and Y are the input for the clustering algorithm.

Iterative DBSCAN Method - Step Zero

Remove ‘noise’ clusters (don’t have any near neighbors)



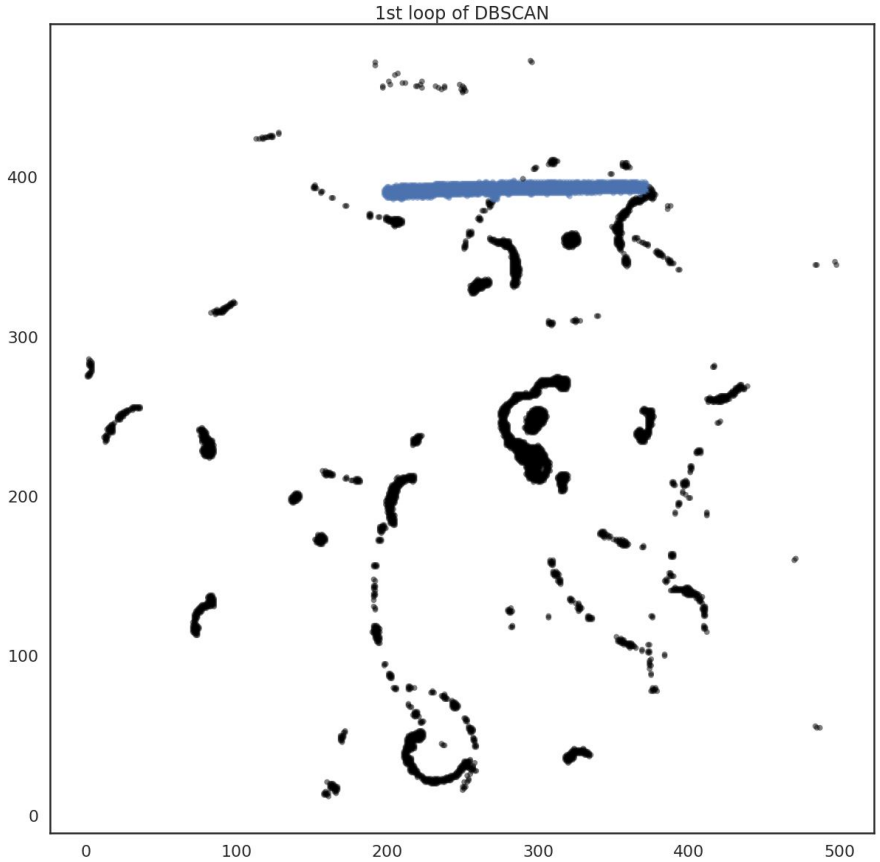
Iterative DBSCAN Method - Step One

In this loop DBSCAN was set to look for groups of pixels that have **high** density.

When the algorithm find a cluster in this step it is labelled as ‘1’.

Then, the found clusters are removed from the image to proceed to the next step.

(In the image at right different colors means different clusters and the black ones represents the not found clusters)

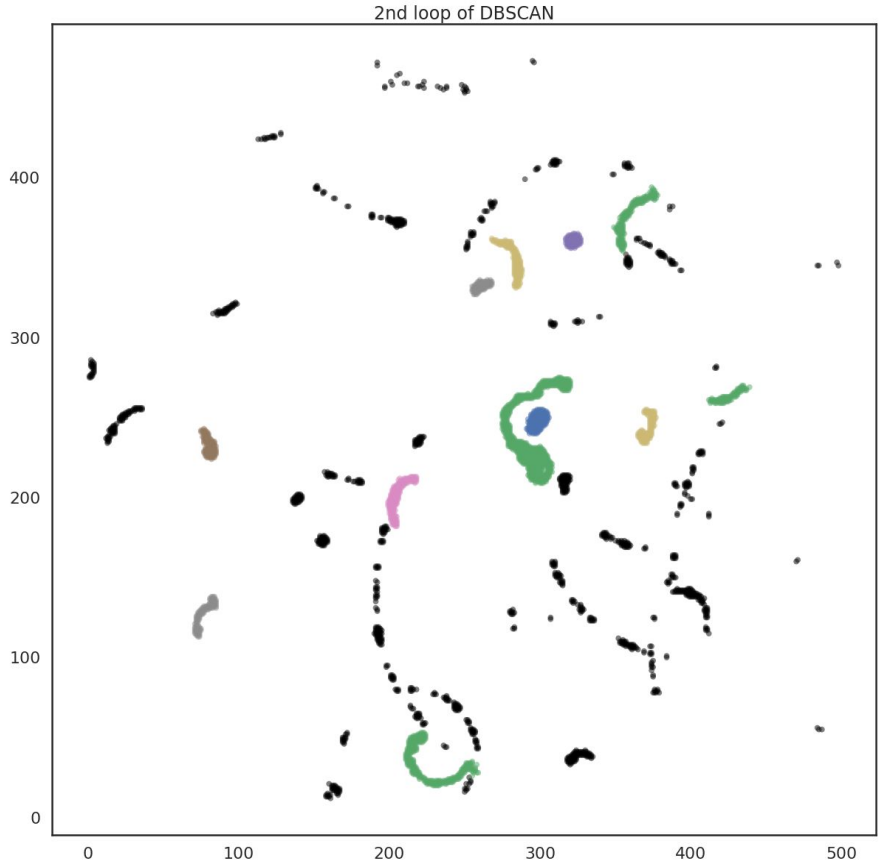


Iterative DBSCAN Method - Step Two

The second loop try to find groups of pixels with not so high density, let's say **medium** density.

And, as in the first step the found clusters are labelled as '2' and removed from the image to proceed to the next step.

(In the image at right different colors means different clusters and the black ones represents the not found clusters)

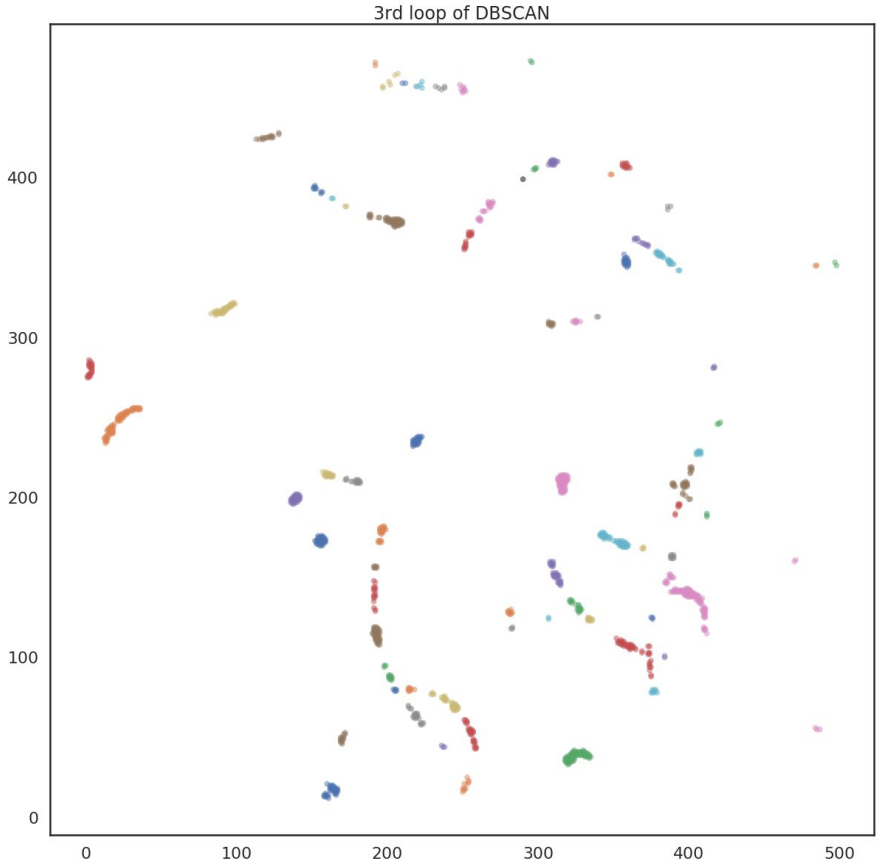


Iterative DBSCAN Method - Step Three

The last one is more flexible and the goal here is find the signals that aren't found yet.

In this case the label is '3' and the output of all steps is save for further analysis.

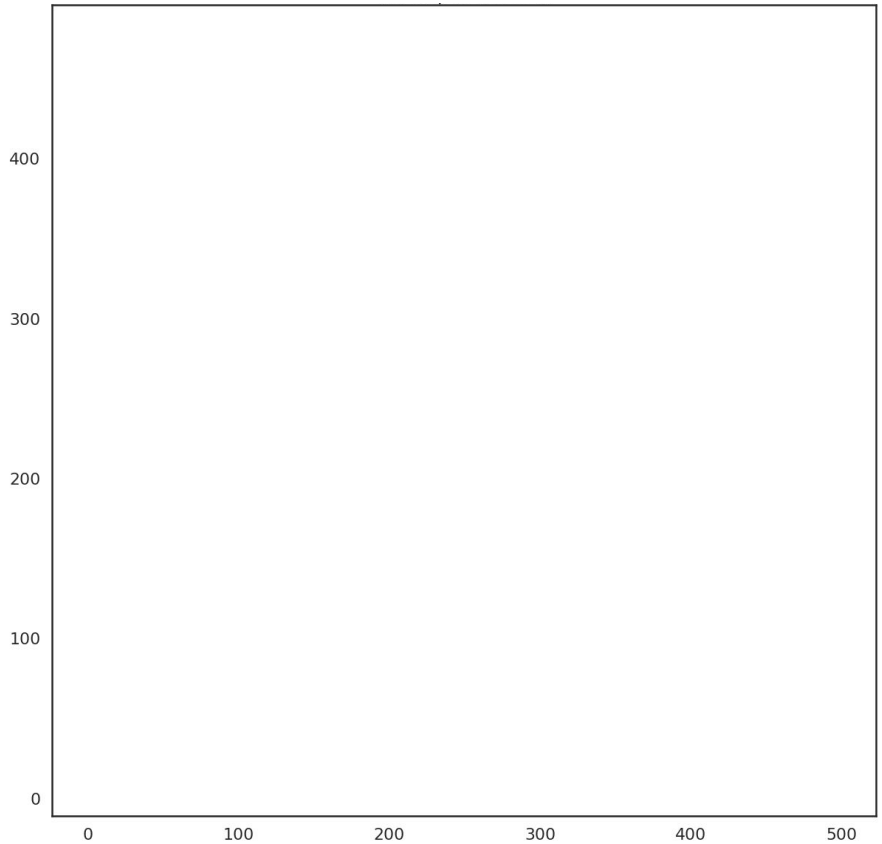
(In the image at right different colors means different clusters and the black ones represents the not found clusters)



Iterative DBSCAN Method - After the third step

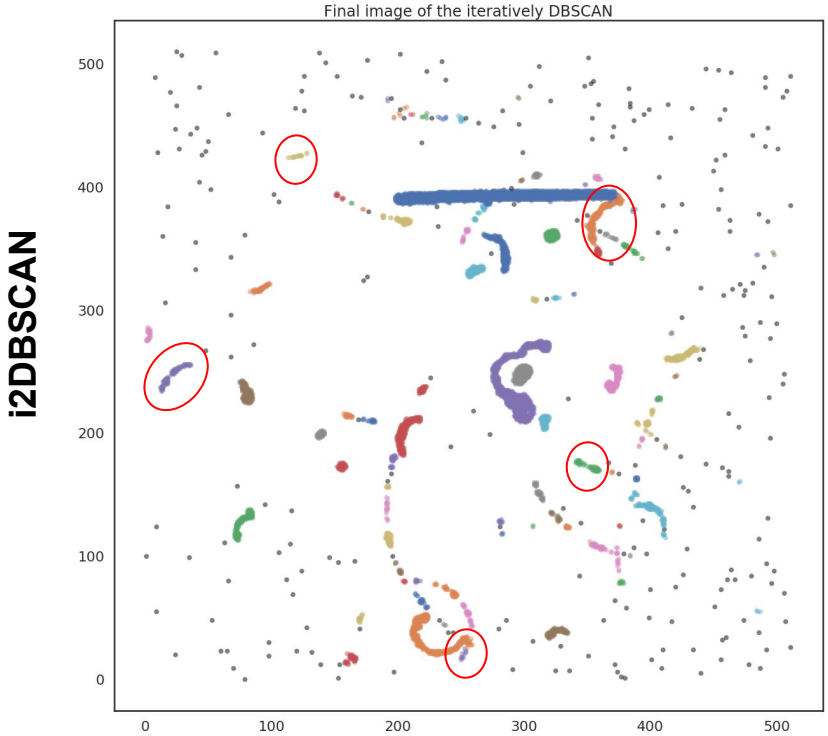
In this particularly case, after the third iteration we don’t have any pixels without being found.

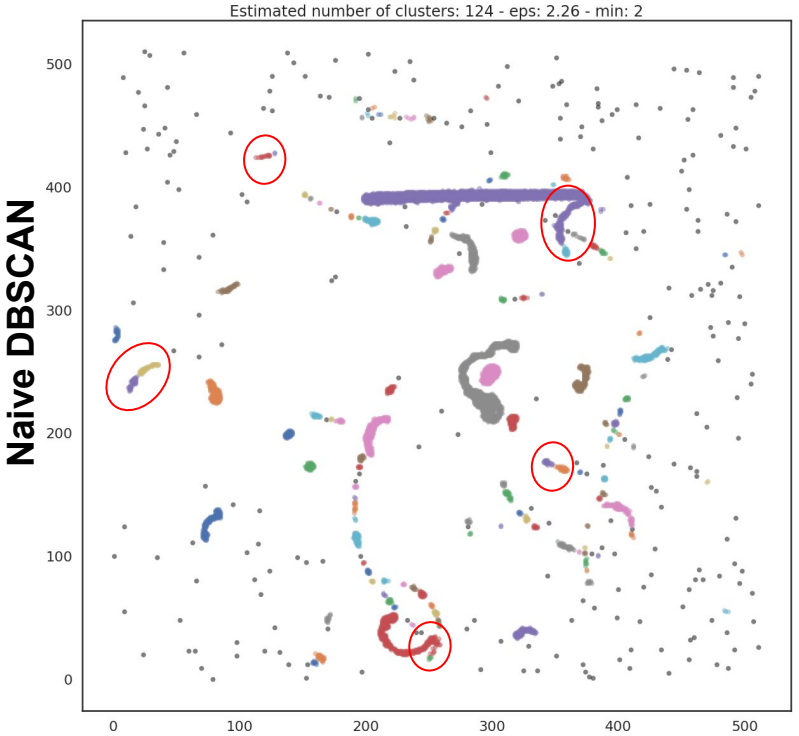
Notice that: This is due to the **Step Zero**, that already removed from the image the ‘noise pixels’.



i2DBSCAN x DBSCAN

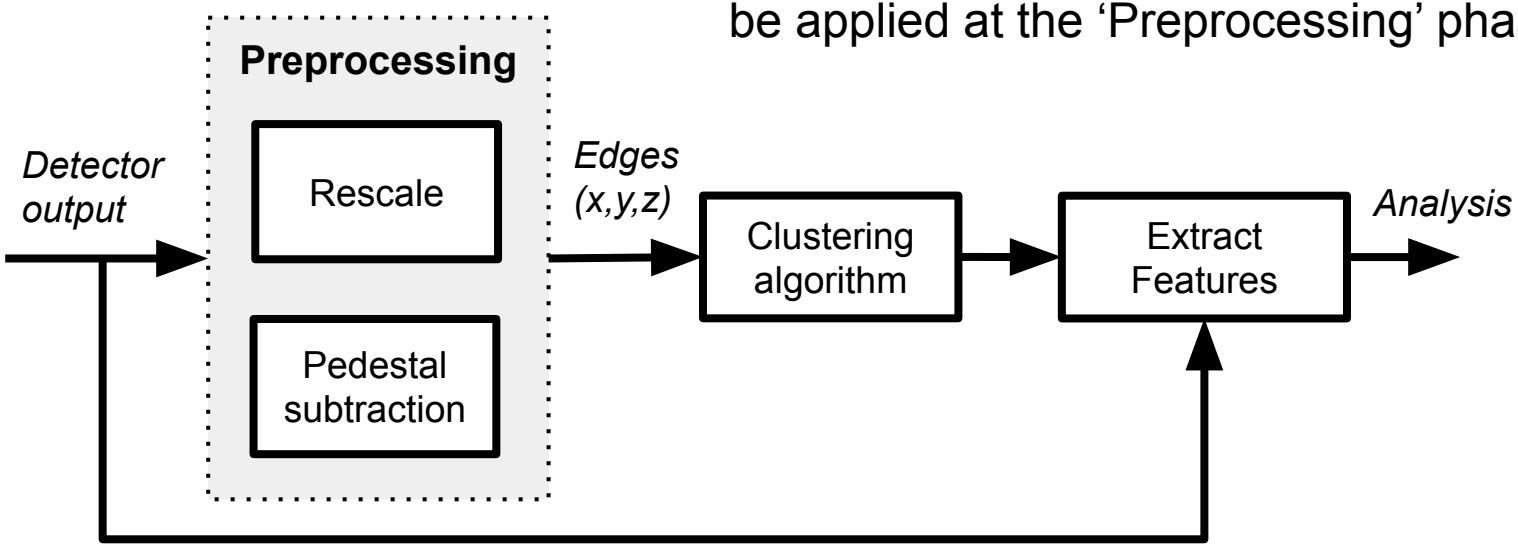
So, the difference between the two methods are not so big by looking at one image. However, it can be an improvement considering a hundred of events.





Workflow of the algorithm

Also, filtering methods are under study to be applied at the 'Preprocessing' phase.



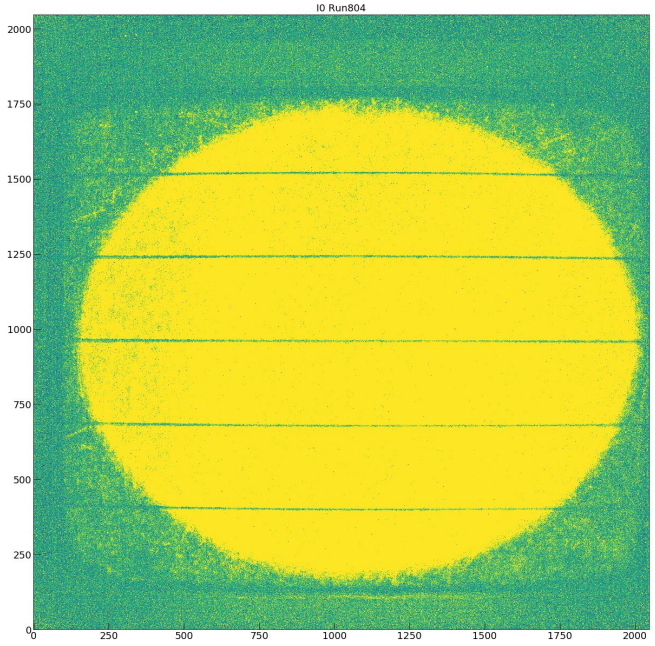
Note that the algorithm developed is such to reconstruct efficiently most of the visible clusters, because without a simulation is difficult to say what really is signal and background.

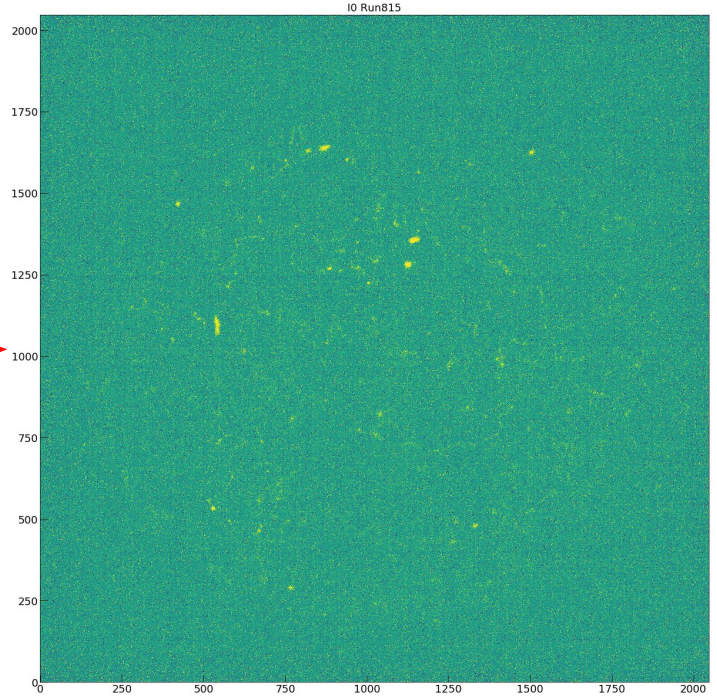
Preliminary Results on FNG data

About the FNG data

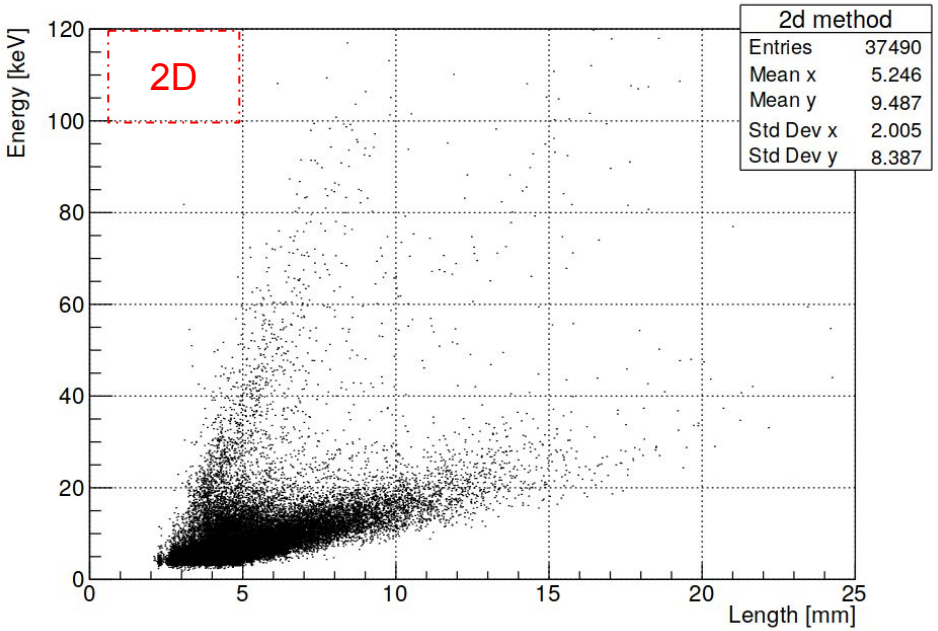
This image is using an exposure time of 10s, in order to show the region within the "field cage".

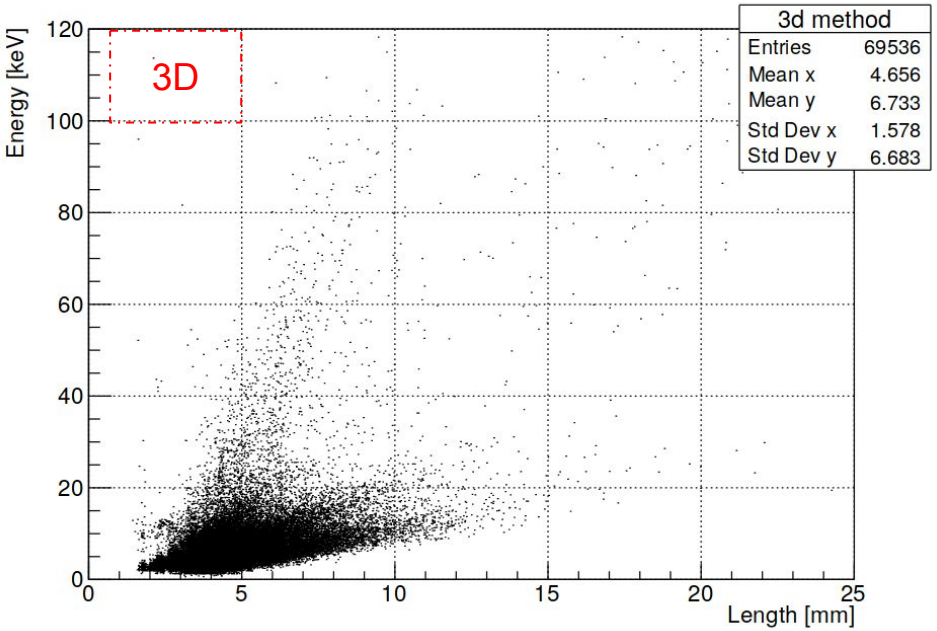
GEM Voltage (V)	He:CF4	Transfer field strength (kV/cm)	CMOS Exposure Time (ms)	Nominal Flux cubic cm/min	Effective flux cubic cm/min	Acquisition Number (# Events)	Up Voltage (V) (fixed)
440	60/40 premix	2	100	300	218.4	300	2120



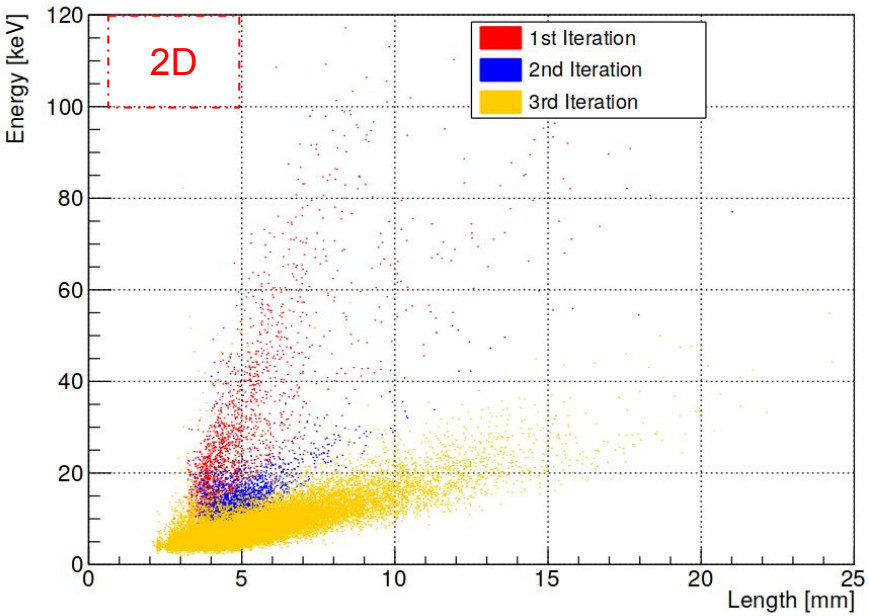


Difference between using (x, y) and (x, y, z)

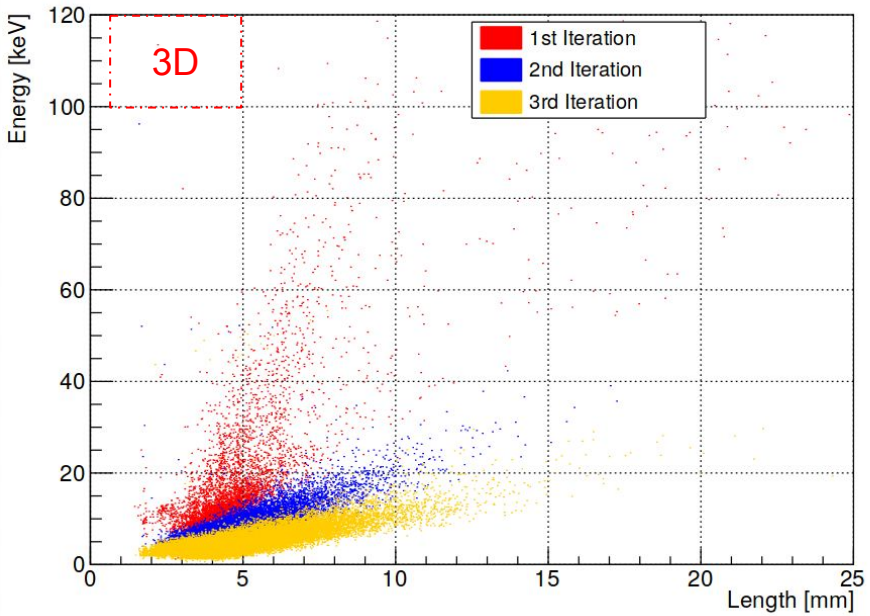




Difference between using (x, y) and (x, y, z)

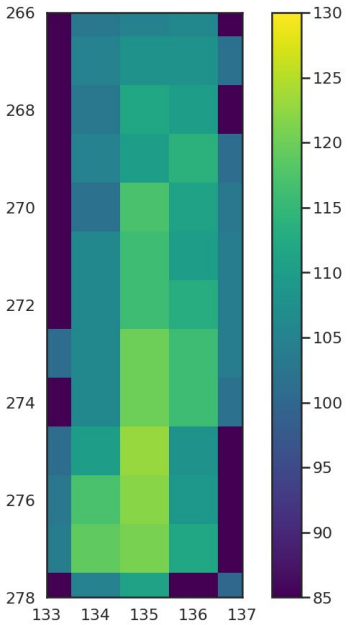


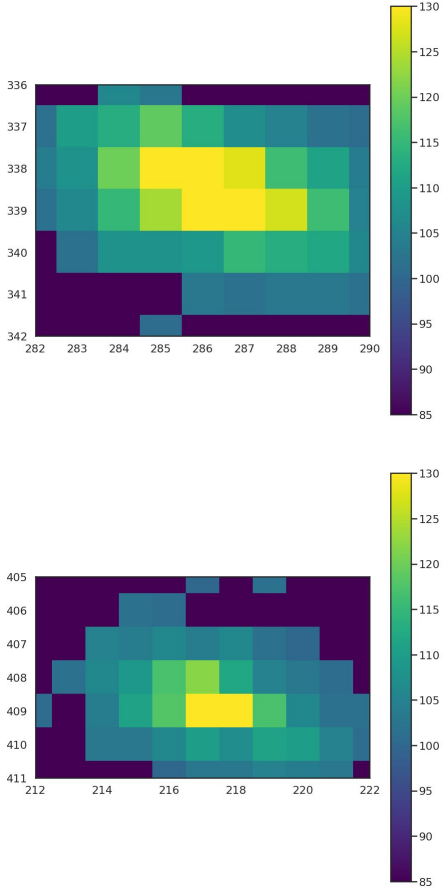
Looking at the images, what we can observe is that with 3D the clustering algorithm is using the 'energy' to select the tracks.

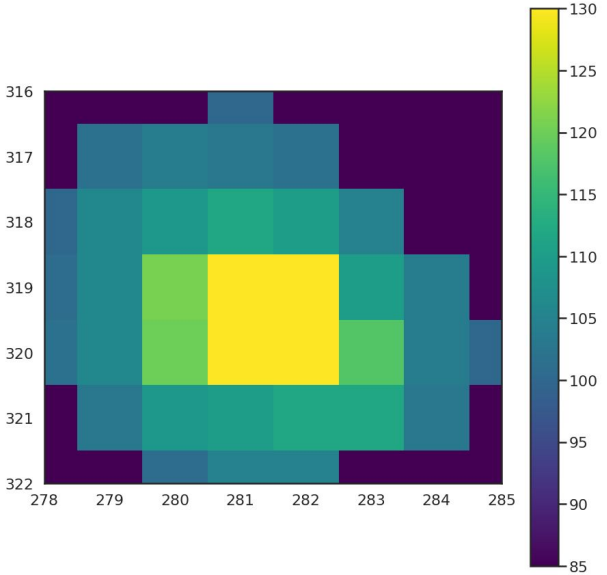


It seems that 3D approach helps in not selecting random noise, because it weights hits with larger energy.

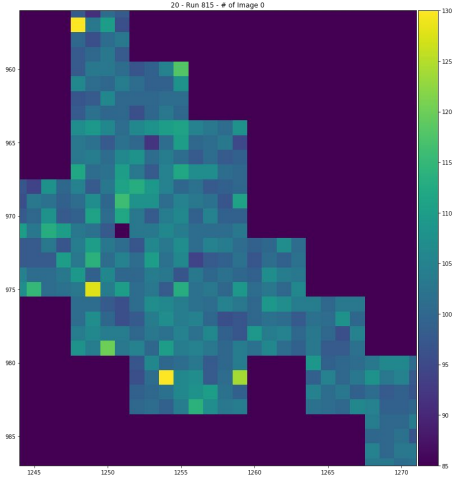
Example of clusters found at iteration 1 (Red)

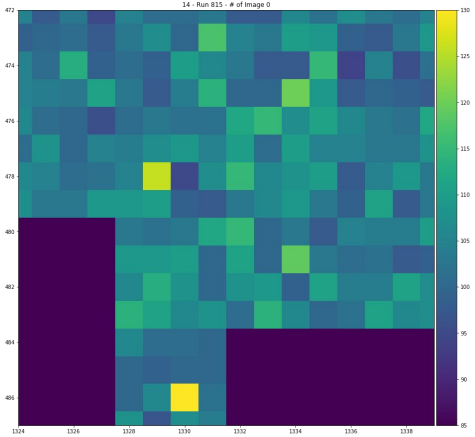


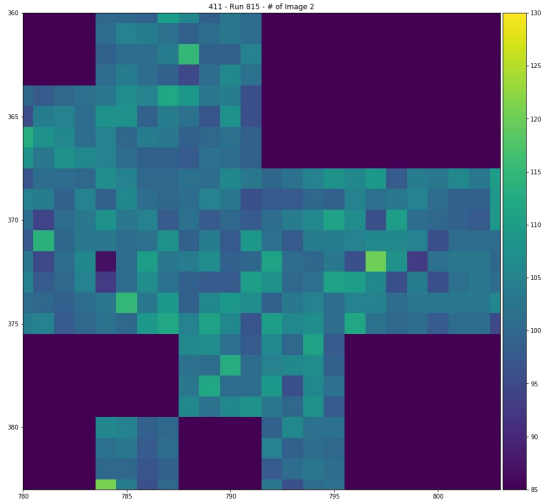


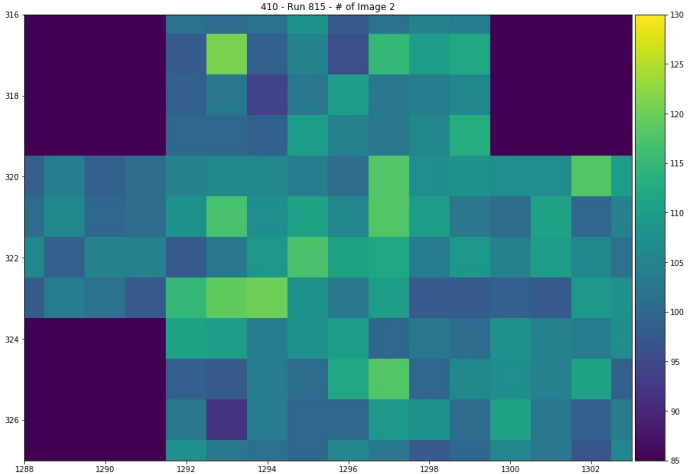


Example of clusters found at iteration 2 (Blue)

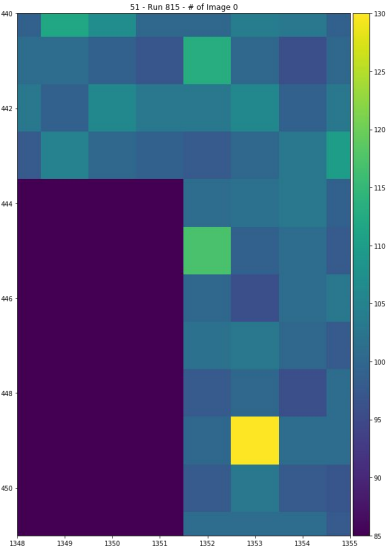


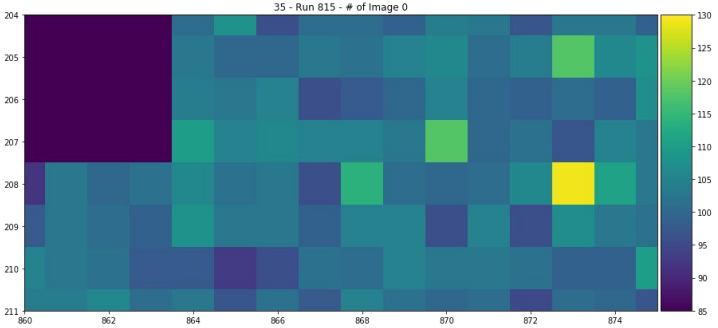


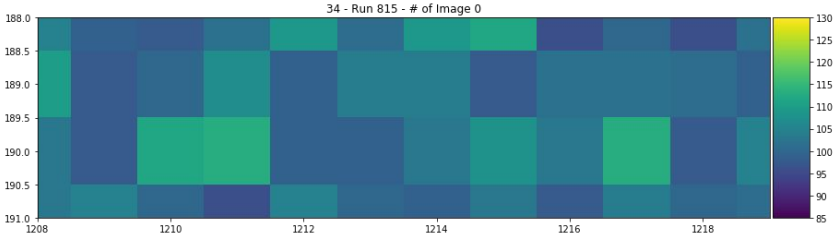


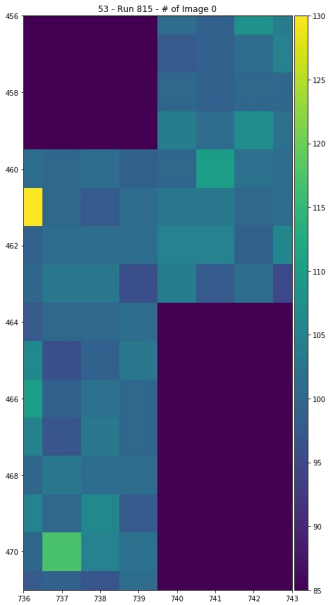


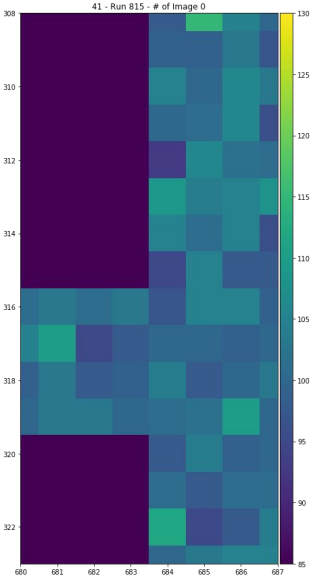
Example of clusters found at iteration 3 (Yellow)



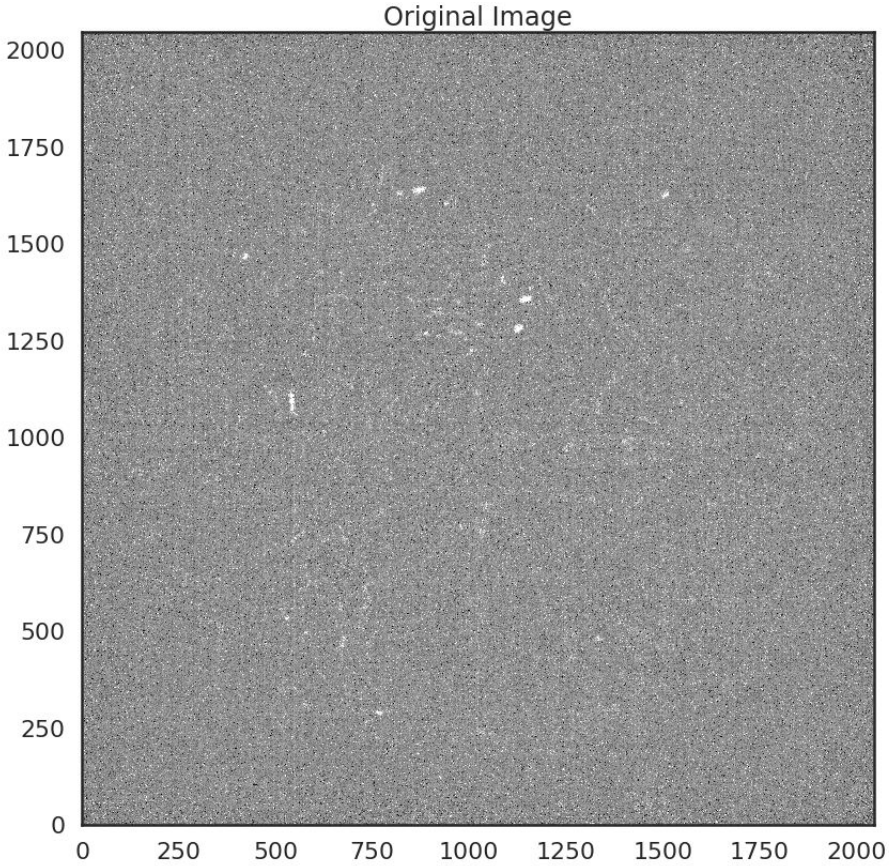


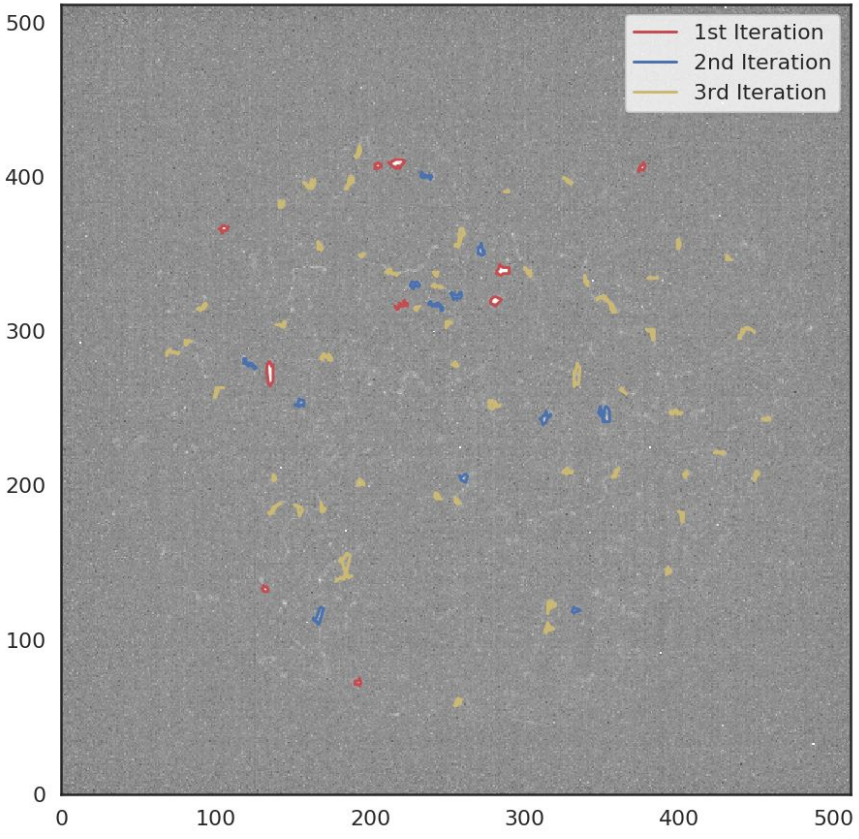






Example of clusters found





Conclusions and further work

- In this work we show that i2DBSCAN could lead us to an improvement when comparing against the naive approach;

- The next steps are:
  - Characterize the background noise of the experiment;
  - Develop a machine learning algorithm to classify the clusters;
  - Look at the discriminant variables (e.g. cluster shapes) for each iteration separately because each iteration may have different Signal/Background.
- Next-next step: we pretend to optimize the algorithm using the simulation + digitalization data.

Thank you!

Backup

RUN 815 - BEAM ON - 440 V - 100ms