UNIVERSIDADE FEDERAL DE JUIZ DE FORA

PPEE
Programa de Pós-Graduação em Engenharia Elétrica UFJF

Directional-iDBSCAN

a proposal to CYGNO

Igor Pains

Igor Abritta and Rafael A Nobrega

Last presentation

- An implementation of the iDDBSCAN with its iteration part written in cython was made aiming to reduce the duration of the algorithm.

- The speed boost was not so high as other implementations found in the literature.

- New ways to accelerate the algorithm were researched.

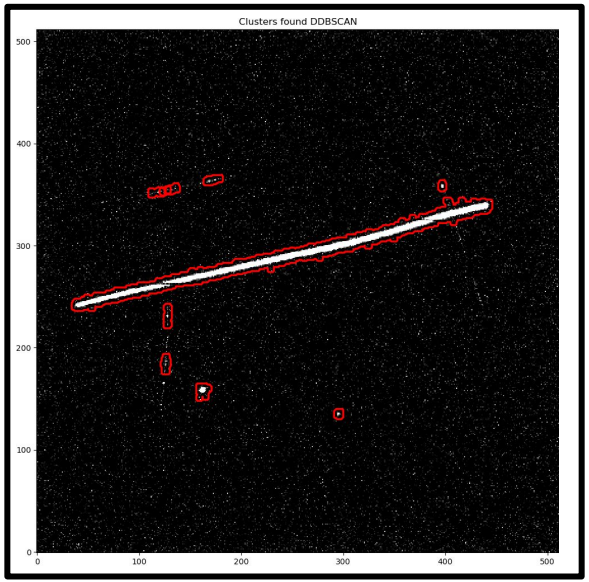
Problems with the 3D

- There was a problem with the duration of the 3D simulation in some runs of the lime data.

- During the search for the solution for this problem we found about the parameter “*sample_weight”* of the “*fit( )”* method of the DBSCAN, that allows the possibility to assign weights to each point.

3D simulation vs Sample Weight

Run 2320 - Event 135



3D

Replicate the points based on the intensity.
(from 3888 to 39442 here)

run time: 40.4 milliseconds

Send the new data array to the DBSCAN to find clusters.

run time: 1.28 seconds

Create the sample_weight array based on the intensity.
(3888 elements here)

run time: 0.33 milliseconds

Send the “old” data array and the sample_weight to the DBSCAN to find cluster.

run time: 0.032 seconds

Get the clusters thought the labels attribute.

run time: 1.32 seconds

run time: 0.0323 seconds

Sample_Weight parameter

Sample weight parameter

- Since it is already available on the *scikit-learn* library, no modification directly in the DBSCAN was necessary.

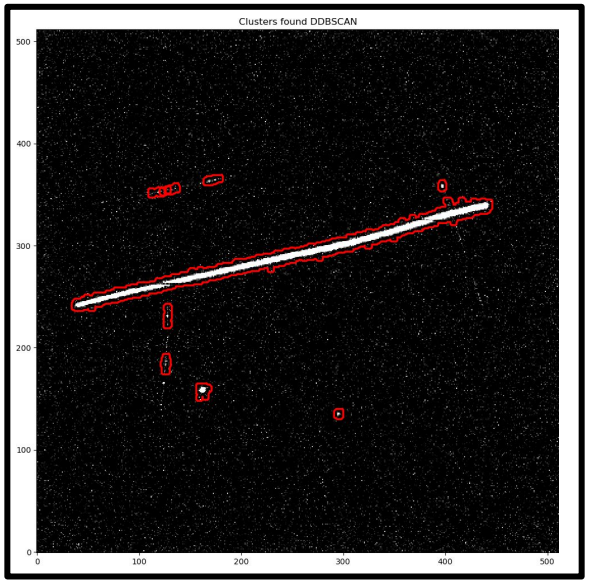
- The difference between the old and new algorithm is that now the *sample_weight* array is built instead of the larger data array with replicated points, allowing the possibility to submit fewer points to the iterative part of the DBSCAN and maintain the 3D results.

Sample weight parameter - DDBSCAN

- The DDBSCAN algorithm had to be slightly changed in order to use the "*sample_weight".*
  - The "false cluster" (*halos*) remotion done by the *min_samples* (input of the DBSCAN) was removed. It is now done by the length of the smallest cluster found in the DBSCAN part (DBSCAN seeding).
  - The parts of the algorithm that used to eliminate replicated points to compare to the *dir_minsamples* parameter or to send the points to the RANSAC are not necessary any longer.
- This parameter also solves the problem of the high amount of time used looping through replicated points.

Sample weight parameter - DDBSCAN

Run 2320 - Event 135



- This example was one of the “outliers” in the histogram shown on a previous presentation.

- If no control parameter is used with the DDBSCAN (*time_threshold* or *max_attempts*), the gain of speed in this image is 160x (from 790 to 4.9 seconds).

Conclusions

- The *sample_weight* parameter is a great addition both to DBSCAN and DDBSCAN, conciliating results and time efficiency.

- The DDBSCAN specifically can still be optimized, especially the RANSAC part, which now is probably the most demanding part of the algorithm.