Short-term identification of gamma-ray transients inside Cherenkov Telescope Array sky maps

Nicola Severini - 0000851091
Matteo Galletti - 0000851080

https://github.com/Seve95/Gamma-Ray-Transient-Analysis

1. Dataset Generation

For this step we used *ctools* version 1.5.2, a software package developed for the scientific analysis.
Ctools are based on *GammaLib*, a versatile toolbox for the scientific analysis of astronomical gamma-ray data. It consists of a C++ library and a Python module that exposes the full functionality of the library to Python which will be the language for the project. The library provides an abstract data analysis framework that is independent of any specific gamma-ray telescope.

First step of the project is the simulation of event data with *ctobssim* command specifying a set of parameter that define the characteristics of the simulation. One of this parameter is the input model, an xml file which specifies the distribution of source and background of gamma ray flares.
There is an approximate value indicating the signal to noise ratio, called sigma:

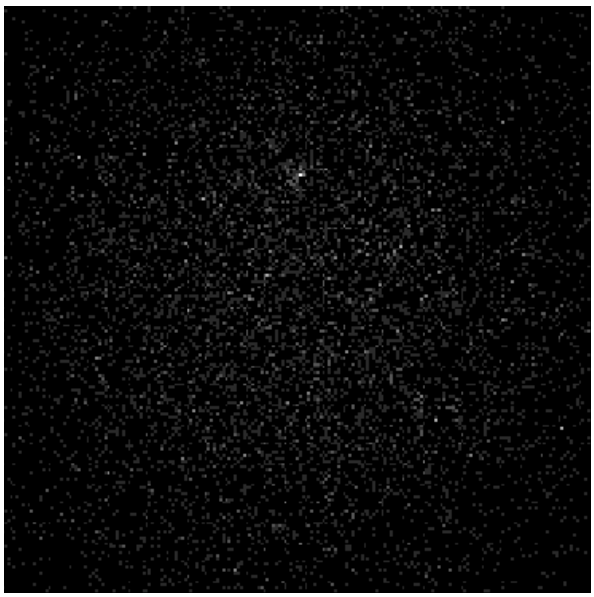
$$\sigma = \frac{S}{\sqrt{S+B}}$$

the value of S can be found in the *ctobssim* log file, in the *MC source events* field, i.e the number of photons register as source events, while the *S+B* must be verified in the ctselect log file in the *number of observed events* fields, i.e the sum of both source and background events registered.
ctselect command select source and background events without distinction, within the event list using a search radius, 0.2 in our case because it corresponds to the PSF of this instrument (a system's impulse response) at these energies.
The value of sigma used in the test phase is 3,7 obtained by setting a time interval of 15 minutes.

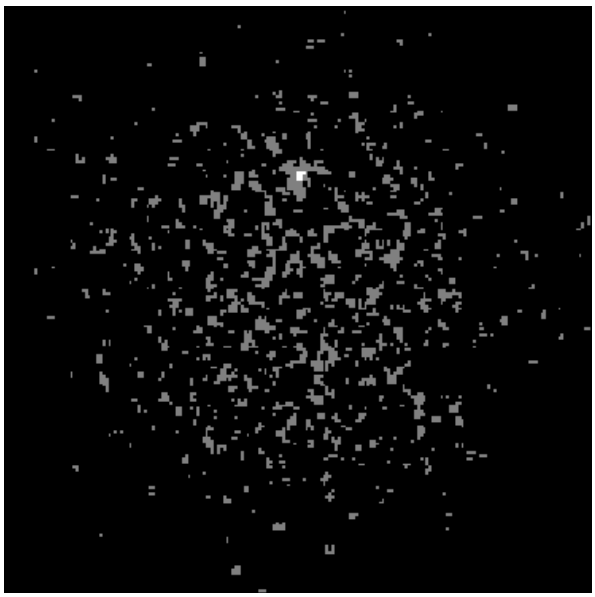
Once created the event data, the second step is to generate skymap using ctskymap command. The tool produces the file *skymap.fits* which contains a sky map of the events in FITS format. The sky map is centred on a fixed location and consists of *200* **x** *200* spatial pixels of *0.02* **x** *0.02* degrees in size, covering an area of *4 deg* **x** *4 deg*.
To generate a dataset of skymaps we have defined a python script called *generate.py* which allows to create a parametric number of skymaps with a single source, whose position varies by a random value in range [-1 ,1] degrees of both celestial coordinates (RA, DEC), or without any source.

2. Pre-processing of the Skymap

The first step of the pre-processing process is the application of a smoothed filter 3x3 used for simplify the following step.

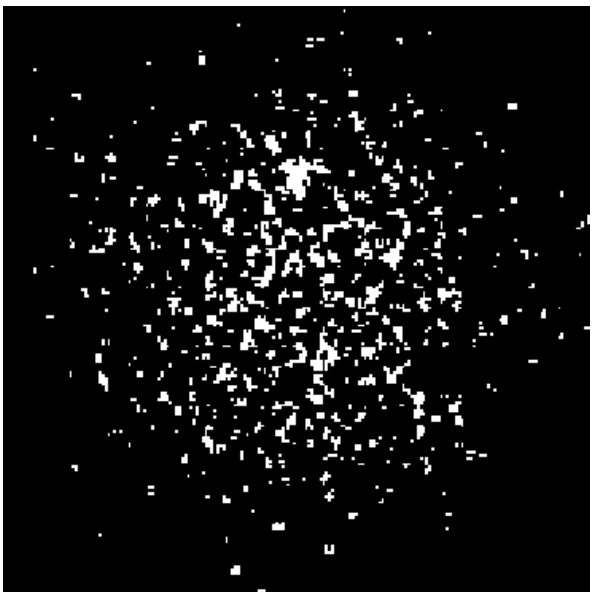


ORIGINAL MAP



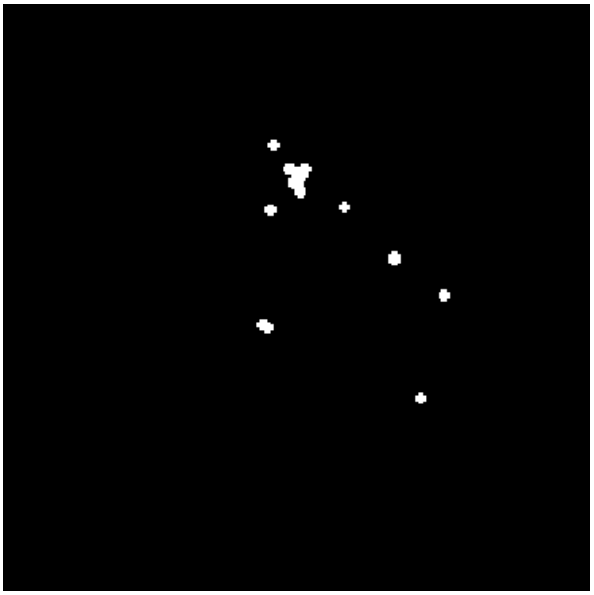
SMOOTHED MAP

Second step is a binarization to keep only the brighter sections of the skymap. Initially a threshold equal to 1, allowing a greater skimming of the map so as to be able to easily manage the cases with a very high sigma. If this value leads to zero blobs, try with a threshold of zero, thus keeping all the pixels greater than zero on the map.



MAP AFTER BINARIZATION STEP

Third step is the application of a binary morphology operator, the opening, with a 4x4 circular matrix, which highlights only the most density regions of the binerized skymap.

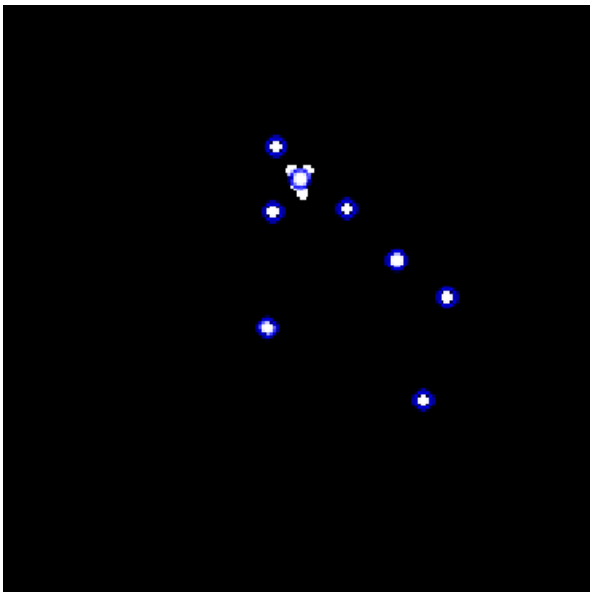


MAP AFTER OPENING STEP

3. Blob Analysis

The next step is blob analysis provided by *opencv* object called *SimpleBlobDetector_create* which allows to detect the list of blobs and eventually to found the source, if there is one.

If the number of blobs is equal to zero, the value of the binarization threshold is decreased and the detector is recalculated with the new result of the opening.



DETECTED BLOBS

If the number of blobs is one before decreasing the threshold, we can assume we are in the case of a high sigma and the single blob is the source.

When more blobs are found, we calculate the average of the areas of the various blobs and if the ratio between the area of the major blob and the average is not greater than a certain threshold, we exclude the possibility of having a source in the map, otherwise the blob with the largest area will be the source. This threshold was found after a training phase in which the average ratio was calculated in the maps without source and in those with source and the average of these two values was chosen as the threshold.

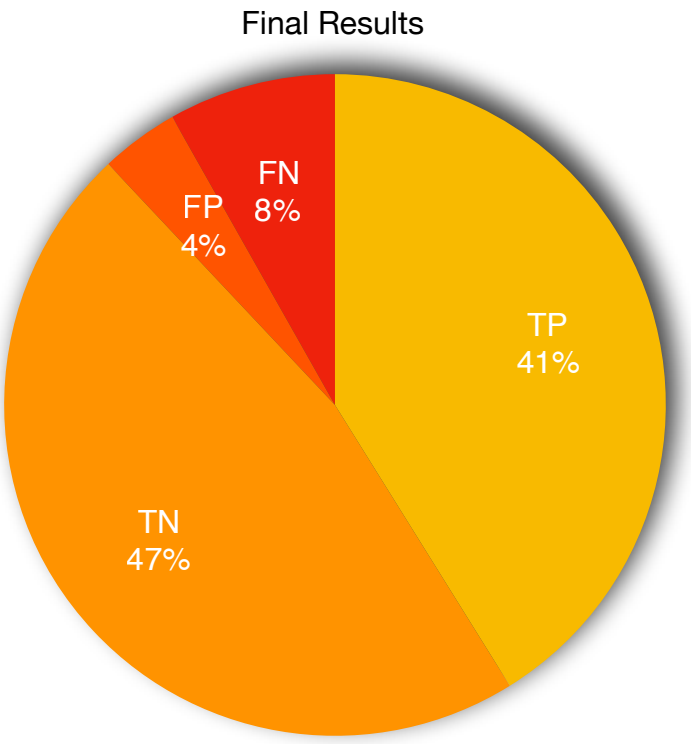
if a source has been found, the next step is to extract the image coordinates of the blob and transform it into celestial coordinates through the appropriate astropy function; if these coordinates coincide with those of the model except for an error given by the approximation of the image, we have a true positive, otherwise a false positive.

4. Conclusions

Even if the optimal value is obtained using the threshold found by the training phase, it is possible to modify the value of this threshold to have a different ratio between false positives and false negatives; for example, if we want to minimize the number of false positives as much as possible even at the cost of having a worse overall result, it is possible to do so by increasing the threshold value, vice versa for false negatives.

Possible improvements can be obtained by using a neural network trained with a much larger dataset than that used to test our algorithm.

Below are the results of the application of the algorithm on 6000 maps, 3000 with source and 3000 without source.



Confusion Matrix

Total population: 6000	**Condition positive: 3000**	**Condition negative: 3000**	**Prevalence: 0,45** $\frac{predicted\ positive}{total\ population}$	**Accuracy: 0,88** $\frac{True\ positive + True\ negative}{total\ population}$
Predicted positive: 2698	True positive: 2468	False positive: 230	Precision: 0,91 $\frac{true\ positive}{predicted\ positive}$	False discovery rate: 0,09 $\frac{false\ positive}{predicted\ positive}$
Predicted negative: 3302	False negative: 491	True negative: 2811	false omission rate: 0,15 $\frac{false\ negative}{predicted\ negative}$	negative predicted value: 0,85 $\frac{true\ negative}{predicted\ negative}$
	True positive rate: 0,82 $\frac{true\ positive}{condition\ positive}$	False positive rate: 0,08 $\frac{false\ positive}{condition\ negative}$	Positive likelihood ratio: 10,25 $\frac{TPR}{FPR}$	
	False negative rate: 0,16 $\frac{false\ negative}{condition\ positive}$	True negative rate: 0,94 $\frac{true\ negative}{condition\ negative}$	Negative likelihood ratio: 0,17 $\frac{FNR}{TNR}$	