Gamma Ray Sources Detection

1. Introduction

This project aims at detecting gamma-ray transients observed by a gamma-ray observatory. A gamma-ray transient phenomenon occurs when a significant change in the level of gamma-ray emission is detected in a specific region of the sky.
Gamma-ray emissions are the most energetic form of electromagnetic radiation, they are usually associated with catastrophic events involving compact objects (e.g. black holes, neutron stars, gamma-ray bursts and more).

The context is applicable, among other projects, to the CTA, Cherenkov Telescope Array, Observatory (2), an array of more than one hundred ground-based telescopes (Image 1) that will be built in the Atacama Desert in Chile and in La Palma.

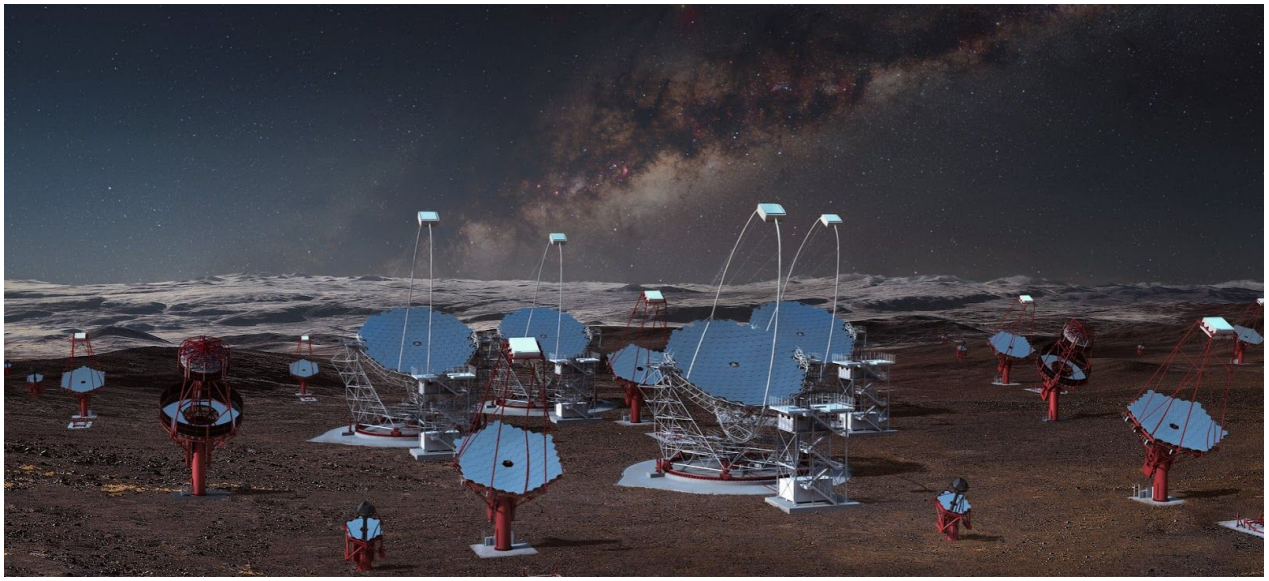


Image 1: Rendering of the three classes of telescopes planned for the southern hemisphere.
Credit: Gabriel Pérez Diaz (IAC)/Marc-André Besel (CTAO)/ESO/ N. Risinger (skysurvey.org)

CTA will be able to detect the cascades of subatomic particles created when gamma rays interact with the earth's atmosphere (Sequence shown in Image 2). These particles travel faster than light in air and each cascade only lasts a few billionths of a second and can't be seen by the human eye.

This work will simulate these events by creating different sets of skymaps, using CTA's ctools (1), and will use them in order to try to find a way to detect gamma-ray transients automatically.

Image 2: From left to right, cascade of subatomic particles created by interaction of gamma rays and Earth's atmosphere.
Credits: CTAO, https://www.cta-observatory.org/

A typical skymap is represented at the left in image 3. Each square pixel represents the number of photons for the sky region covered by the area of the pixel, while to the right there's an example of an elaborated image, obtained by the same skymap through ds9 (4), an application for astronomical imaging and data visualization.

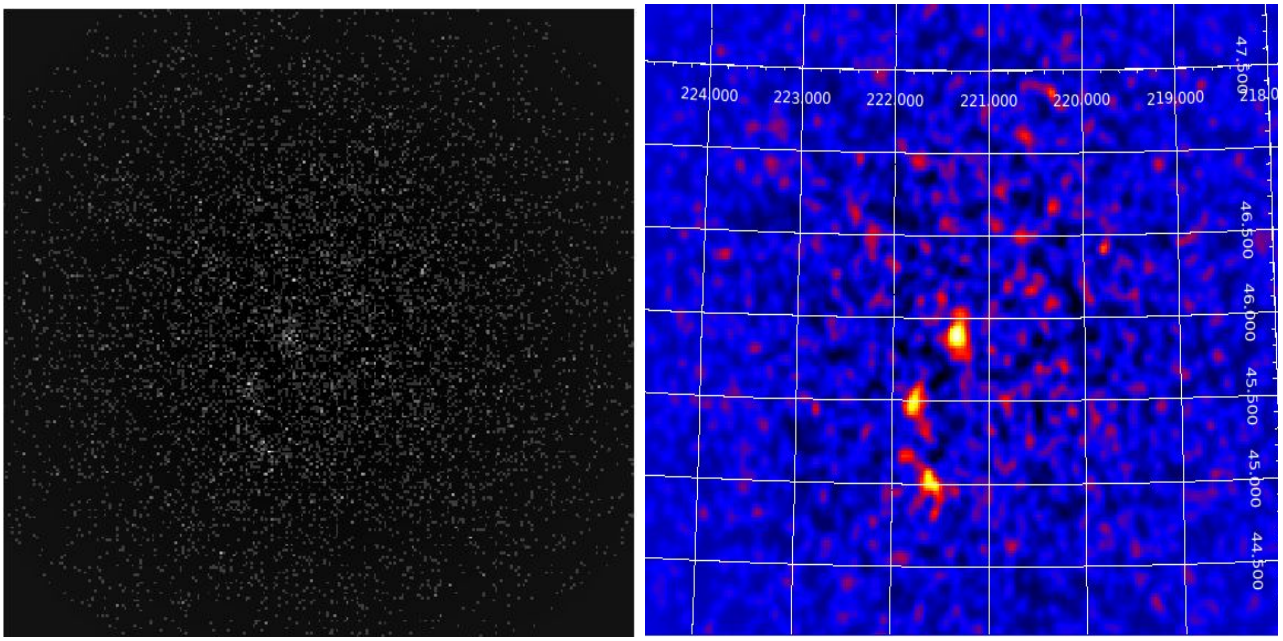


Image 3: Skymap with 3 point sources with the same flow, to the left we have the original skymap, while to the right the same skymap elaborated with ds9, by applying gaussian smoothing with kernel of dimension 6 and sigma 1.5, coloured and with a superimposed coordinates grid.

Python has been chosen as a programming language, the library Astropy (3) implements a useful set of methods that can be used to read a skymap as a python matrix and to compute distances between points in celestial coordinates. Consequently it has been used for this project, along with numpy (6) and scipy (5).

2. Dataset creation

In order to generate a skymap one needs to create an xml file that will be taken in input by “ctobssim”, one of the three ctools’ commands used, and that specifies the position of each point source, their numbers, and the signal to noise ratio, along with other parameters which have not been considered by this project.

Xml files are created by “generateXml.py”, specifying how many xml files have to be generated and how many point sources each one will have.

The position of each point source is given in celestial coordinates, and they are randomly picked in a circular region around (221, 46), of radius 1.

The python script can also change the flow of each point source randomly, in the range [1, 2]. This feature has been used to understand the behaviour of the algorithm as the ratio between noise and signal changes.

The results reported in paragraphs “Initial results” are obtained with a noise to signal ratio equal to $sigma = S/sqrt(S + B) = 4$.

Where S is the value of ‘MS source events’ in ctobssim’s log and S+B is the value of ‘number of observed events' in ctselect’s log when selecting with radius 0.2 degrees around the position of the point source.

The xml files will be saved, along with a csv file which contains the position of each point source for each file, to the folder “data/xml_files/<number of point sources>

Subsequently, through “generateSkymaps.py” one can use the xml files to generate skymaps, specifying the type of background subtraction has to be used, the number of point source per skymaps that should be present, the maximum change for the flow of each point source and the minimum distance between each point source.

Based on these parameters the xml files will be read from “data/xml_files_/<number of point source>” and the skymaps will be saved to “data/skymaps/<background subtraction>/<number of point source>” with name “skymap_<background>_<num_file>.fits”, where “num_file” is a unique integer.

“generateSkymaps.py” uses “ctobssim”, “ctselect” and ctskymaps” to, respectively, simulate an observation, with the parameters found in the xml files, select a specific area of the observation and ultimately to create a skymap.

These three methods could also be used from the command line by writing all the required parameters by hand. The python script is used to speed-up the process.

Along with the xml file in python, the position of the centre of the observation, the radius of the area to select and the type of background subtraction these commands take in input also other parameters, such as the type of coordinate system (“CEL” as celestial for this project), the interval of time considered by the observation (15 minutes in our case), the instrument response function and the calibration database (respectively, South_0.5h and prod2)

1) python3 generateXml.py --num_files=<int> --num_sources=<int> --change_flow=<float> --min_dist=<float>
2) python3 generateSkymaps.py --backsub=<NONE/IRF> --num_sources=<int>

3. Image processing

Each skymap is read as a numpy matrix, and it is then smoothed with a gaussian kernel 5x5 with sigma = 1.5. The result is used to estimate a threshold value as reported in each “Result” paragraph. An example of a smoothed image can be seen in image 4.

```
kernel = gaussianKernel(kSize, kSigma)
for i, mat in enumerate(matrices):
     if i >= len(matrices_smoothed):
     matrices_smoothed.append(gaussianBlur(mat[0].data, kernel))
```

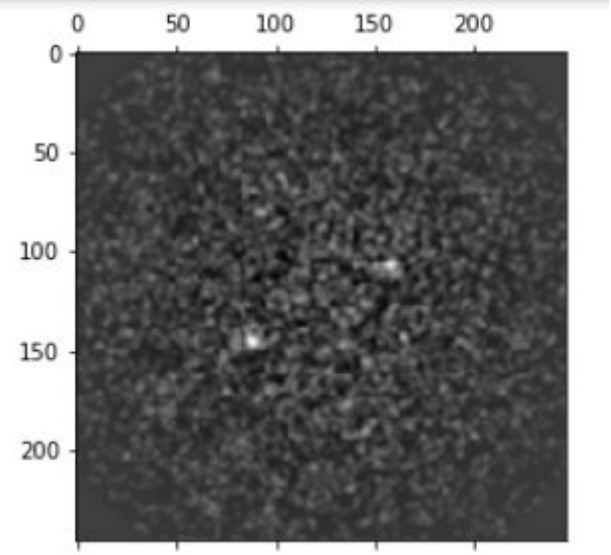


Image 4: Smoothed skymap with two point sources, kernel dimension is 5 and sigma is 1.5.

The threshold value is used to binarize each image, and the result is then eroded to remove noise (only in the case of IRF background subtraction, no erosion in case of NONE), unfortunately in the process also some point sources are lost or made too small to be recognized by the blob detector. An example of binarized skymap is in image 5, while image 6 represents an eroded map.

```
matrices_bin = []
for i, mat in enumerate(matrices_smoothed):
     matrices_bin.append(np.zeros((dimx, dimy)))
     matrices_bin[i][mat > thresh] = 1
```

```
for i, mat in enumerate(matrices_bin):
     if i >= len(matrices_eroded):
     matrices_eroded.append(binErosion(copy.deepcopy(mat),
np.array([[1,1], [1, 1]])))
```

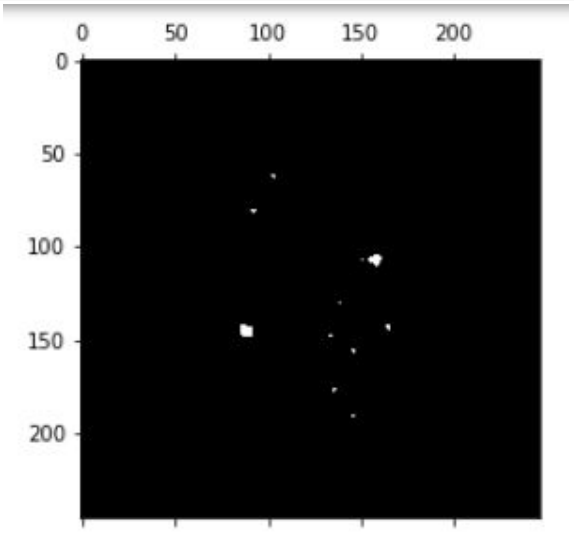


Image 5: Binarized image, this has been obtained by the image represented in image 4.

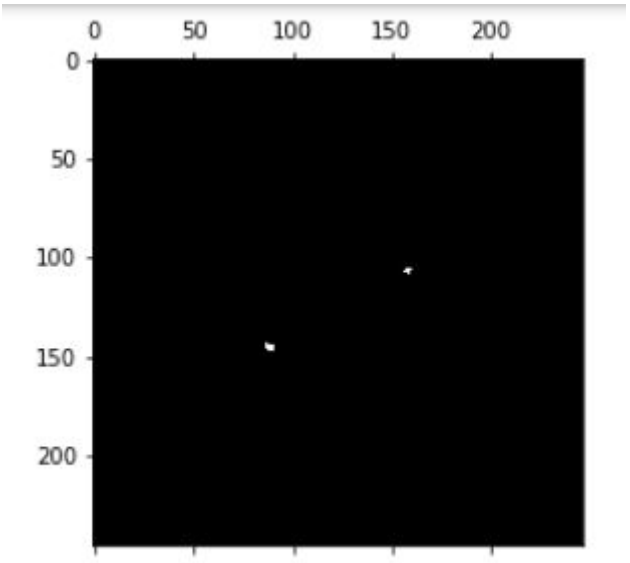


Image 6: Skymap in image 5 after erosion. It can be noticed that the smallest white points (noise) have been removed from the image.

One way to try to avoid getting sources too small to be detected is to perform opening instead of erosion, that is perform erosion and dilation, this however could cause the presence of false positives.

In order to save time, it is possible to save smoothed or eroded maps in binary format and then to load them if needed.

```
save("data/skymaps/IRF/"+str(SOURCES_NUMBER)+"/matrices_smoothed.npy",
asarray(matrices_smoothed))
```

```
# load numpy array from npy file
matrices_smoothed =
load("data/skymaps/IRF/"+str(SOURCES_NUMBER)+"/matrices_smoothed.npy")
```

The eroded matrix is subsequently analyzed by the blob detector “blob_doh” of scipy (5), and the resulting blobs are matched with the closest point source existing in the given image (the closest centroid is chosen as the one with the smallest angle between celestial coordinates).

```
blobs = dict()
mean_error_pix = []
mean_error_angle = []
data_taken = 0
for i, mat in enumerate(matrices_eroded):
      blobs[i] = []
      cont = 0
      blobs_doh = blob_doh(mat, max_sigma=20, threshold=.01)
      for blob in blobs_doh:
      y, x, r = blob
```

```
        if r > 1:
            cont += 1
            blobs[i].append((int(x), int(y)))
        if cont == SOURCES_NUMBER:
        data_taken += 1
        else:
        blobs[i] = []
```

```
for blob in blobs_key:
    to_be_saved.append(blob)
    closer_blob_dist = 10000
    sky_obs = wcs.pixel_to_world(blob[0], blob[1])
    for i, blob_true in enumerate(blob_pos[key]):
        sky_true = SkyCoord(float(tmp_rows[key][2*i+1]),
float(tmp_rows[key][2*i+2]), unit='deg')
        tmp_dist = (sky_true.separation(sky_obs)).degree
        if tmp_dist < closer_blob_dist:
            closer_blob_dist = tmp_dist
            closer_blob = blob_true
    mean_error_angle.append(closer_blob_dist)
    mean_error_pix.append(np.sqrt((closer_blob[0]-blob[0])**2 +
(closer_blob[1]-blob[1])**2))
```

The skymaps in which the correct number of point sources has been found are saved in the folder “data/dataset_<none/irf>”, together with a csv file with the coordinates of the found centroids, for future analyses.

```
origin = src_folder+"/skymap_IRF_"+str(key)+".fits"
dest = "data/dataset_irf/skymap_IRF_"+str(key)+"_"
       +str(SOURCES_NUMBER)+".fits"
shutil.copyfile(origin, dest)
# Add contents of list as last row in the csv file
csv_writer.writerow(to_be_saved)
```

The notebooks used to perform this operations are “irf.nypb” and “none.nypb”, in the second cell of each one is possible to change the number of point source and the gaussian kernel.

```
SOURCES_NUMBER = 3  # Should be equal for all data used
kSize = 5  # Size of smoothing kernel
kSigma = 1.5  # Sigma for smoothing
```

4. Introduction to the results

In the following paragraphs are shown the results obtained with different sets of parameters, they are represented in tables, where for each set of background subtraction, number of sources and, only for some of the results, maximum signal-to-noise ratio, are reported:

- # blob found: Percentage of skymaps in which the right number of pixels has been found;
- Mean pix/angle: Mean error in pixel or mean angle between celestial coordinates;
- Max pix/angle: Maximum error in pixel or max angle between celestial coordinates;
- Min pix/angle: Minimum error in pixel or minimum angle between celestial coordinates.

Columns “Back sub” and “# sources” are used as identifiers and represent, respectively, the type of background subtraction used and the number of sources present in each skymap.

Plots have also been used: firstly, in order to better show the differences between different sets of parameters; secondly, in order to represent in a cleaner way the robustness of the algorithm with respect to decreasing signal-to-noise ratio.

The title of each plot will report both the information represented and the type of background subtraction used.

In some of the plots the range of the y-axis doesn’t start from 0, this choice has been made in order to make the differences easily visible.

5. Initial Results

5.1. 3x3 Kernel and 8 mean background level

Using 1000 samples for each number of point sources.
Threshold value used:

For NONE background subtraction: 8*mean background level
For IRF background subtraction: histogram of intensities

The kernel used for erosion (only in IRF) is the following:

1 1 1
1 1 1
1 1 1

Back sub	# source	# blob found	Mean pix	Max pix	Min pix	Mean angle	Max angle	Min angle
None	1	89.40%	3.97	5.83	1.41	0.08	0.11	0.03
None	2	53.90%	4.11	6.71	2.24	0.08	0.13	0.04
None	3	38.80%	4.54	7.81	1.0	0.09	0.15	0.02
None	4	37.4'%	4.56	7.28	0.0	0.09	0.15	0.01
Irf	1	98.80%	3.58	6.71	1.41	0.07	0.14	0.03
Irf	2	92.30%	4.01	6.71	1.01	0.08	0.14	0.03
Irf	3	74.60%	4.40	7.81	1.0	0.09	0.17	0.01
Irf	4	69.80%	4.46	10.0	0.0	0.09	0.21	0.01

5.2. 2x2 Kernel and 7 mean background level

Using 1000 samples for each number of point sources.
Threshold value used:

For NONE background subtraction: 7*mean background level
For IRF background subtraction: histogram of intensities

The kernel used for erosion (only in IRF) is the following:

1 1
1 1

Back sub	# source	# blob found	Mean pix	Max pix	Min pix	Mean angle	Max angle	Min angle
None	1	98.30%	3.85	5.83	1.41	0.08	0.12	0.02
None	2	88.10%	4.09	6.41	1.41	0.08	0.13	0.03
None	3	68.80%	4.53	7.81	2.0	0.09	0.15	0.04
None	4	66.90%	4.53	8.25	1.0	0.09	0.17	0.01

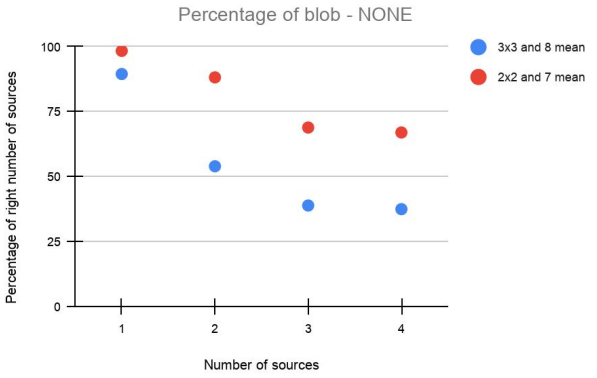
Back sub	# source	# blob found	Mean pix	Max pix	Min pix	Mean angle	Max angle	Min angle
Irf	1	97.60%	2.95	5.01	1.01	0.06	0.09	0.01
Irf	2	92.30%	3.29	6.71	1.01	0.07	0.13	0.03
Irf	3	79.40%	3.71	8.11	0.0	0.07	0.15	0.00
Irf	4	67.00%	3.76	12.0	1.0	0.07	0.25	0.02

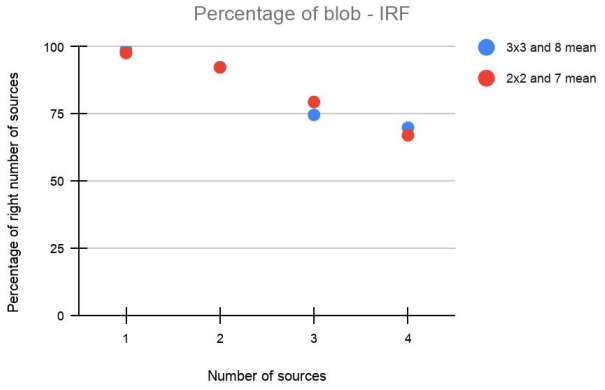
5.3. Comparing results

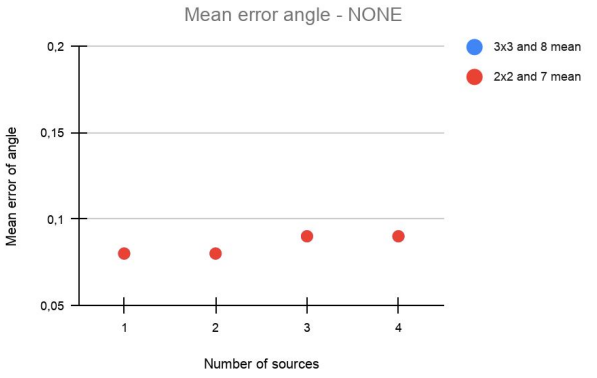
The plots below represent the results obtained in the previous paragraphs and have been used to extract the best parameters found.

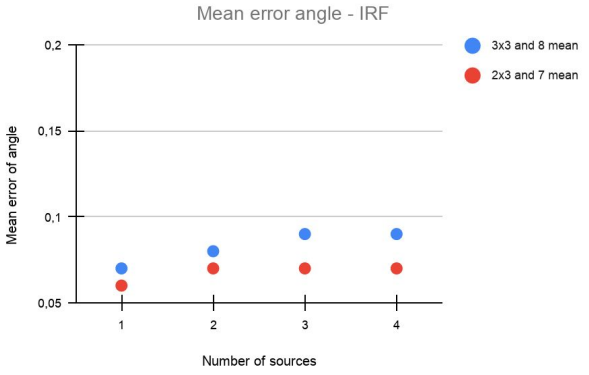
In the x-axis of each plot are shown the number of sources present in each skymap, while in the y-axis is shown the information described by the title of the plot.

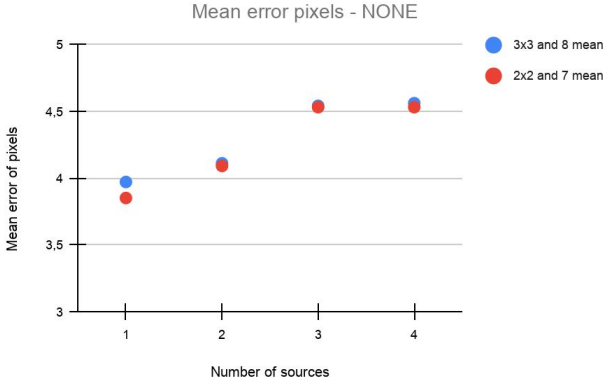
As can be seen, the parameters used in “2x2 Kernel and 7 mean background level” led to better results, with respect to those found in “3x3 Kernel and 8 mean background level”, increasing the percentage of the correct number of blob found, in the case of NONE background subtraction, and decreasing the mean error in the position of each point source, in the case of IRF background subtraction.

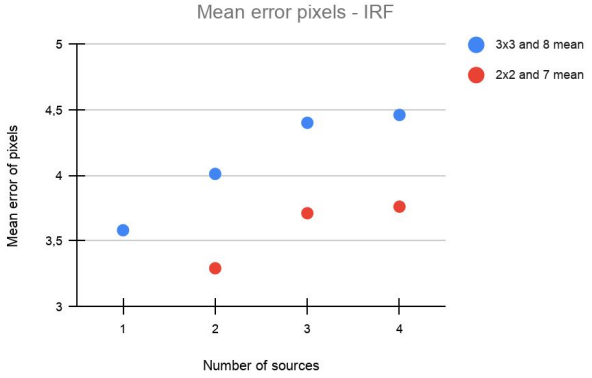












6. Changing sigma

Evaluating the algorithm as signal-noise ratio changes, using 100 samples for each number of point sources.

The values for sigma are computed with one point source.

Threshold value used:

For NONE background subtraction: 7*mean background level
For IRF background subtraction: 0.35 (obtained by precedent results)

The kernel used for erosion (only in IRF) is the following:

1 1
1 1

Max Sigma: 3.89

Back sub	# source	# blob found	Mean pix	Max pix	Min pix	Mean angle	Max angle	Min angle
None	2	85.00%	4.01	6.71	1.01	0.08	0.13	0.02
None	4	39.00%	6.63	7.81	0.0	0.09	0.14	0.01
Irf	2	86.00%	3.22	5.39	1.01	0.06	0.10	0.01
Irf	4	65.00%	3.82	7.07	1.41	0.08	0.13	0.03

Max Sigma: 3.63

Back sub	# source	# blob found	Mean pix	Max pix	Min pix	Mean angle	Max angle	Min angle
None	2	66.00%	4.16	6.40	1.41	0.08	0.13	0.03
None	4	28.00%	4.58	7.28	2.0	0.09	0.15	0.04
Irf	2	86.00%	3.39	5.0	0.0	0.07	0.10	0.01
Irf	4	68.00%	3.81	7.28	1.41	0.08	0.15	0.03

Max Sigma: 2.68

Back sub	# source	# blob found	Mean pix	Max pix	Min pix	Mean angle	Max angle	Min angle
None	2	6.00%	4.32	5.39	2.83	0.09	0.11	0.05
None	4	1.00%	4.38	5.66	3.16	0.08	0.11	0.06
Irf	2	57.00%	3.32	6.08	0.0	0.07	0.12	0.01
Irf	4	47.00%	3.56	7.07	1.0	0.07	0.14	0.02

Max Sigma: 2.25

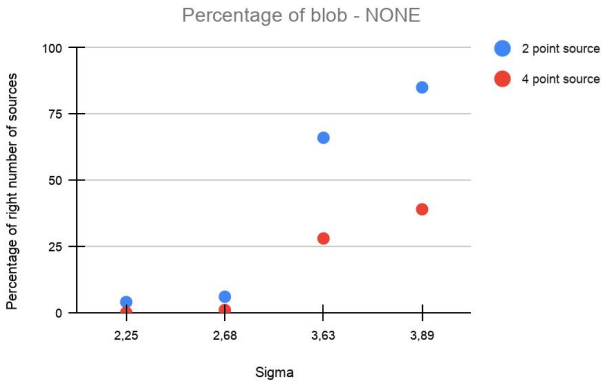
Back sub	# source	# blob found	Mean pix	Max pix	Min pix	Mean angle	Max angle	Min angle
None	2	4.00%	3.75	5.0	2.24	0.08	0.1	0.6
None	4	0%	Na	Na	Na	Na	Na	Na
Irf	2	21.00%	3.56	6.40	0.0	0.07	0.12	0.01
Irf	4	20.00%	3.79	7.62	1.0	0.07	0.15	0.02

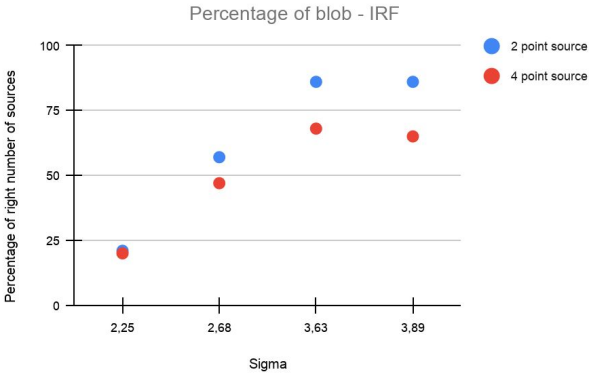
6.1. Comparing results

The plots below represent the results obtained in the previous paragraph. They represent the percentage of files in which the correct number of blobs has been detected with respect to the value of the signal-to-noise ratio, sigma.

As can be seen with values below 3.63 there's a sharp decrease in the number of blobs detected in the case of NONE background subtraction, reaching almost 0%.
While when using IRF background subtraction the percentage decreases significantly, but doesn't go below 20%, reaching values comparable to those found for values of sigma above 3.63 without IRF background subtraction.

Consequently, the tests below have been conducted with 3.63 as the minimum value for sigma.





7. Randomly changing flow of sources

Using 1000 samples for each number of point sources.
Threshold value used:

- For NONE background subtraction: 7*mean background level
- For IRF background subtraction: histogram of intensities

The kernel used for erosion (only in IRF) is the following:

1 1
1 1

Subtracted random number between 0 and 1 from the flow of every source, thus lowering the signal to noise ratio, starting value 2 (signal-noise ratio = 4).

Back sub	# source	# blob found	Mean pix	Max pix	Min pix	Mean angle	Max angle	Min angle
None	1	67.60%	4.19	7.21	2.0	0.08	0.14	0.04
None	2	46.5%	4.12	7.28	0.0	0.08	0.14	0.01
None	3	27.8%	4.26	7.28	1.0	0.09	0.15	0.02
None	4	19.2%	4.49	9.22	0.0	0.09	0.19	0.01
Irf	1	88.20%	3.28	6.39	1.01	0.07	0.13	0.01
Irf	2	70.80%	3.38	7.0	0.0	0.07	0.14	0.01
Irf	3	64.40%	3.56	9.49	0.0	0.07	0.18	0.00
Irf	4	54.50%	3.72	10.0	0.0	0.07	0.19	0.01

8. Testing parameters with 5000 samples

Threshold value used:

For NONE background subtraction: 7*mean background level
For IRF background subtraction: 0.35 (Taken from previous results)

The kernel used for erosion (only in IRF) is the following:

1 1
1 1

Subtracted random number between 0 and 0.50 from the flow of every source, thus lowering the signal to noise ratio, starting value 2 (signal-noise ratio = 4).

Back sub	# source	# blob found	Mean pix	Max pix	Min pix	Mean angle	Max angle	Min angle
None	2	76.06%	4.08	7.21	0.0	0.08	0.15	0.01
None	4	39.5%	4.58	9.06	0.0	0.09	0.18	0.0
Irf	2	84.82%	3.33	10.05	0.0	0.07	0.20	0.00
Irf	4	66.40%	3.77	12.17	0.0	0.08	0.24	0.01

9. Bibliography

1. http://cta.irap.omp.eu/ctools/.

2. https://www.cta-observatory.org/.

3. https://www.astropy.org/.

4. https://sites.google.com/cfa.harvard.edu/saoimageds9/home.

5. https://www.scipy.org/.

6. https://numpy.org/.