Satellite Tracks Detection in Astronomical Images

Wajdi IMLIKI
wajdi.imliki@epfl.ch

January 15, 2021

Abstract

We present a systematic approach to detect and extract information about satellite tracks in the fields of a high-cadence lensed quasar monitoring project with OMEGACAM on the VLT Survey Telescope, without the use of reference images. A pixel collapse method is carried out for every possible direction α in order to detect the peaks in brightness indicating the presence of satellite or space debris tracks. We use a script called PREDICTSAT to link the satellites or space debris that we detect to those present in the SPACETRACK database in order to access the information of the object responsible for the track, such as its name, its nature, the parameters of its orbit. We show the efficiency of our methods and the limit of detection using hand-added tracks by varying the length and brightness. We also compare the experimental results obtained with those expected statistically by assuming a uniform distribution of the satellites on the images.

Contents

Chapter 1

Introduction

Artificial satellites and space debris around the Earth are a new form of light pollution. Those on low-earth orbits can be especially bright. ESA estimates the total number of Space objects in Earth orbit, today, around 29,000 for sizes larger than 10 cm, and 670,000 for sizes larger than 1 cm. The situation is only getting worse with the launch of the new 5G communications satellite constellations, threatening the Dark Sky, due to the substantial increase in relatively bright satellite tracks in wide-field astronomical observations.

Determining as precisely as possible the orbital parameters of satellites as well as space debris has also become increasingly crucial, as the large number of satellites put into orbit, increasing day by day, can trigger a phenomenon called *Kessler syndrome*, a theoretical scenario in which the density of objects in low earth orbit (LEO) due to space pollution is high enough that collisions between objects can cause a cascade in which each collision generates space debris which increases the likelihood of further collisions, hence the increased need to know with great precision these evolving orbital parameters of satellites and debris in order to avoid this kind of scenario.

Due to the high velocities of satellites, as well as space debris, and in a case of a relatively short time of exposure, the trajectories of these objects are almost linear. The luminosity and the length of the observed tracks depend essentially on the speed of the object, on its height at the time of observation as well as on the way in which its outer surface reflects light.

This report presents an image processing method, using pixel collapsing, to

detect satellites tracks in astronomical images, as well as Near-Earth-Orbit Objects, using astronomical archival dataset, with the objective of setting up a pipeline which automatically analyzes a set of given astronomical images, detecting satellites and Near-Earth-Orbit Objects then characterizing these detected objects based on astronomical archival data-set.

Actual methods for detecting such objects in images are based on difference image analysis using a reference image. Using difference image analysis for this problematic implies many exposures in order to observe the objects which we are looking for. The goal of our method is to detect satellites without difference imaging.

Chapter 2

Pixel Collapse

2.1 Inputs

The input of our process is a $16k \times 16k$ pixels FITS image composed of 32 blocks ($2k \times 4k$) separated by bands of NaN values. Each block will be independently analyzed.

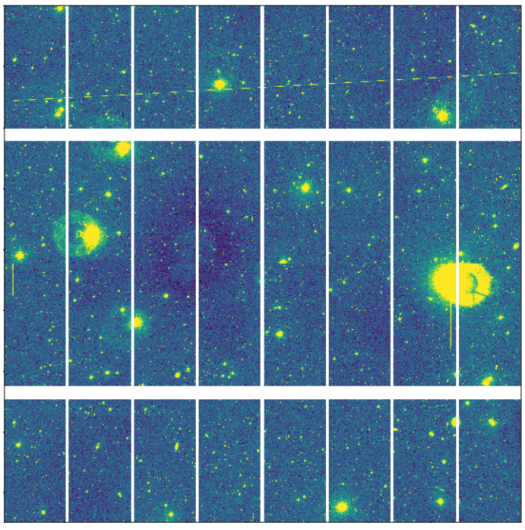


Figure 2.1: FITS Image : Mosaic of 32 blocks

The following table summarizes the different characteristics of the astronom-

ical images studied:

Parameter	Value
Height H	4040 pixels per bloc
Width W	2020 pixels per bloc
Exposure time τ	320 secondes

2.1.1 Predictsat

Predictsat is a script written by Alexis Schizas that allows a user to predict which satellites cross the region of the sky they observe using a list of TLE files obtained from any database. A TLE is a two-line element ans it is a way of encoding data of a satellite such that it fits in two lines. The most important information that can be extracted from the TLE of a satellite are the 6 Keplerian elements of its orbit :

Parameter	Signification
Eccentricity e	Shape of the ellipse, describing how much it is elongated compared to a circle.
Semimajor axis a	The distance between the centers of the bodies.
Inclination i	vertical tilt of the ellipse with respect to the reference plane, measured at the ascending node (where the orbit passes upward through the reference plane).
Longitude of the ascending node Ω	horizontally orients the ascending node of the ellipse (where the orbit passes upward through the reference plane)
Argument of periapsis ω	angle measured from the ascending node to the periapsis (the closest point the satellite object comes to the primary object around which it orbits.
True anomaly ν **at epoch** t_0	the position of the orbiting body along the ellipse at a specific time.

The excentricity and the semimajor axis define the shape of the ellipse. The inclination and longitude of the ascending node define the orientation of the plane of the ellipse, Another important element of data contained in the TLE is the satellite's epoch.

The manipulation of TLEs will be done using the *Skyfield* library of python.

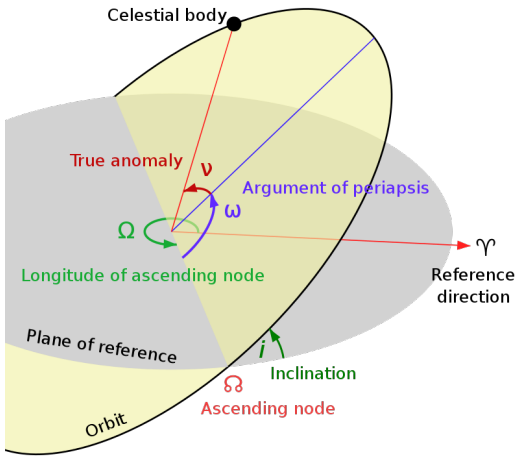


Figure 2.2: Orbit parameters

2.1.2 SpaceTrack dataset

The website used to obtain the list of TLEs of all the satellite tracked is *space-track.org*. The query builder feature allows you to enter two dates, and you get the TLEs of all satellites that were tracked in between. The file has been processed in order to get rid of all the duplicates, since a same satellite can be tracked multiple times.

2.2 Pixel Collapse Method

The figure above shows an example of satellite tracks on an observation. The aim of the method is to detect all the tracks that would appear during this kind of observations.

To do this:

1. We divide the image into small blocks ($2k \times 4k$) that we will convert into RGB matrices. Each block will be then independently analyzed.

2. A mask is applied to replace bright stars with NONE if the considered image contains very bright sources.

3. Now that we have an image with few bright objects left, we set an angle α, and we project the average brightness according to this direction for all possible lines. Each line is defined by an angle α and an intersection with the y-axis which we will call c. The equation of a line in Cartesian coordinates reads:

$$y = \tan(\alpha)x + c$$

This constant varies from $-W\tan(\alpha)$ up to L with W being the width of the bloc (number of columns of the RGB matrix) and L its height (number rows of the RGB matrix). x and y are the cartesian coordinates, α ranges from $-\pi/2$ to $+\pi/2$ and the step $\Delta\alpha$ is equal to $1°$.

Collapsing pixels for every α and c and therefore for every possible direction, will give us an idea of how the bright pixels are distributed over the image. Indeed, a peak in luminosity for a given direction corresponds to a bright line, i.e. a satellite track. The importance of this peak will depend on the brightness of the satellite, as well as the length of its track.

4. The detection of the peak is done using the function of the scipy library *signal.find_peaks*. A study will be carried out to determine the exact parameters to choose, in the general case. The results obtained for each block will be compared with the results of the other blocks in order to determine if a same satellite passes through more than a single block (which is often the case).

5. For each satellite detected, we will try to estimate its speed of rotation by looking at how the brightness varies along its trajectory. We will use the satellite database to determine if we have a match with a pre-existing satellite and therefore determine the various data of this satellite: its name, function, orbit etc.

The diagram below summarizes the process and specifies the different steps necessary for the detection of satellites and space debris from a single image:

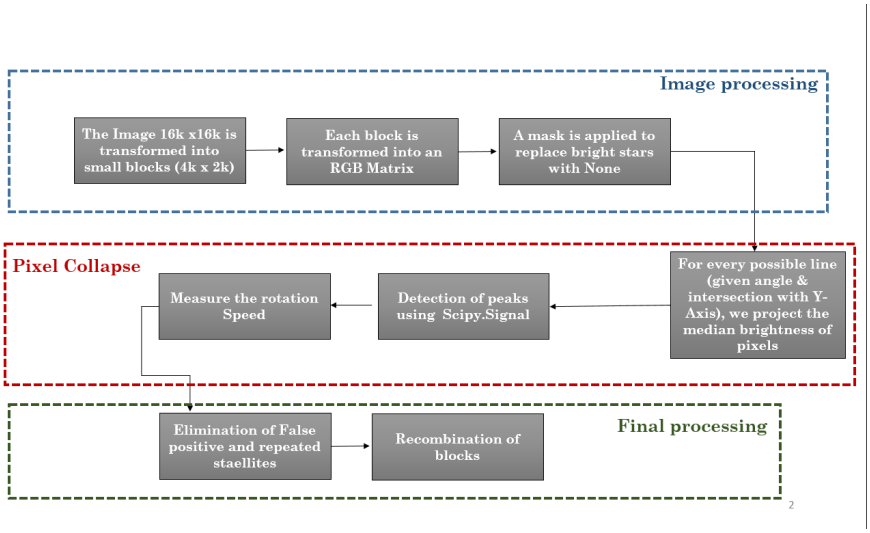
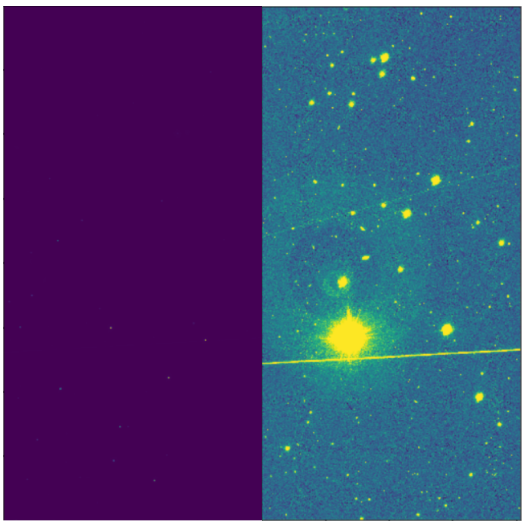


Figure 2.3: The pixel collapse method.

2.3 Z-Scale algorithm

In each block, the intensity of pixels sweep a wide range of values. So in order to be able to process the image in the classical grayscale, we perform a rescaling of the values of each pixel of the image using iraf's zscale algorithm which consists on displaying the values near the median image value without the time consuming process of computing a full image histogram. A sample of pixels of the image is taken and the intensity of these pixels are observed to define a new intensity scale. This intensity scale is defined by two intensity values z_1 and z_2 such that any pixel with an intensity lower than z_1 (resp. higher than z_2) takes the value of z_1 (resp. z_2). As a consequence we obtain an image that shows us the streaks we want to automatically detect through our process.



(a) Unscaled Block (b) ZScaled Block

2.4 Peak detection

Now that we have the evolution of the median brightness in a given direction, we will be able to detect the presence of satellites tracks by looking for the peaks of brightness.

To do this, we will use the *signal.find_peaks* function of the python *scipy* library. This function takes a $1-D$ array and finds all local maxima by simple comparison of neighboring values and by specifying conditions for a peak's properties. Optionally, a subset of these peaks can be selected by specifying conditions for a peak's properties.The function returns the indices of peaks in x that satisfy all given conditions and a dictionary containing the properties of the returned peaks which were calculated as intermediate results during evaluation of the specified conditions : peak heights,left thresholds, right thresholds, prominences, right bases, left bases...

Parameter	Definition	Value required
Prominence p	It describes the importance of the peak. The function will detect all peaks with a prominence higher than the chosen value.	0.02
Width at half maximum w	It describes the required width of the peak curve measured between the two points of having half the maximum amplitude.	Between 5 pixels and 30 pixels

These chosen values were determined by analyzing the satellite tracks in our astronomical images so that the *signal.find_peaks* function detects in the most optimal way possible the tracks with the fewest possible false positives.

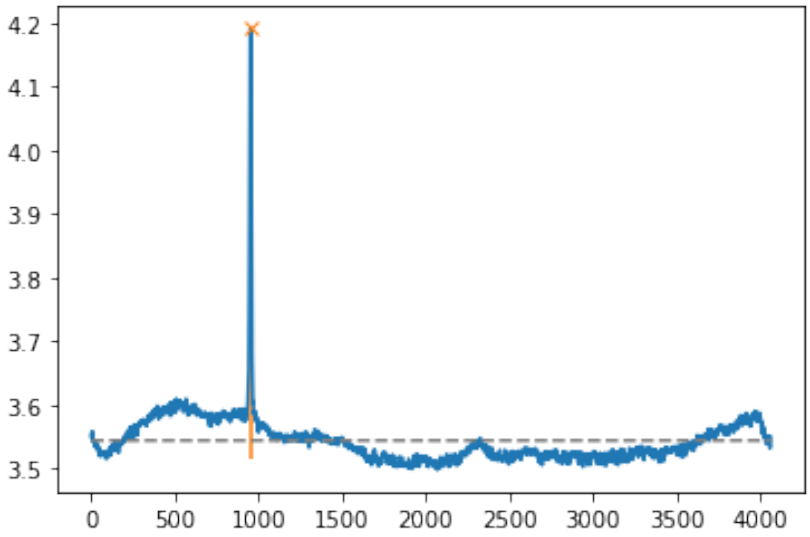


Figure 2.5: The peak detected is marked with an orange "x". For a given direction α, we measure the mean brightness as a function of the intersection with the y-axis, that defines every line in the direction α.

We also notice that in most cases the satellite tracks extend over several blocks. The comparison between the results obtained for each block makes it possible to eliminate more false positives and to reduce the uncertainty on the satellites detected. Indeed, a satellite detected in exactly the same

direction in two adjacent blocks has a good chance of actually being a real satellite.

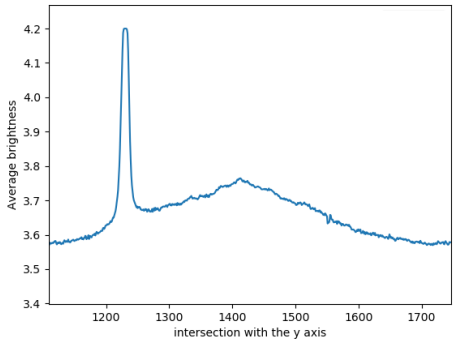
2.4.1 Thresholding

Thresholding consists at eliminating the brightest pixels, in a given direction, due to the presence of very bright light sources such as stars, galaxies or supenovae, affecting, mostly, the detection of faint satellites. In our process, we reject pixels that are much brighter than the median brightness in a given direction α.

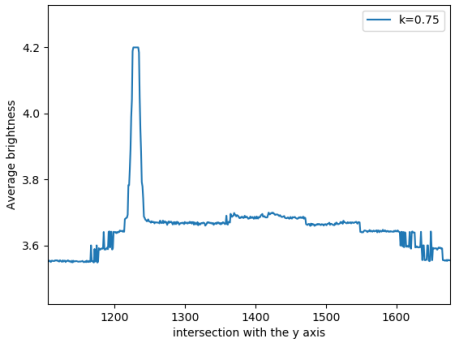
The idea is to take into account only pixels such as :

$$|Brightness - \mu| < k \times \sigma$$

in a given direction, where μ Mean brightness in a given direction, σ the standard deviation in a given direction and k an integer (here we fix $k = 2$). The values of μ and σ are determined using *np.mean* and *np.var* and considering all pixels in a given direction and for a given intersection with the $y-$axis.



(a) Pixel collapse without rejection of bright pixels



(b) Pixel collapse with rejection of bright pixels

We note that the bump in the middle, mainly due to the presence of a large bright star in the middle of the image, disappears by rejecting the bright pixels. In what follows, we will therefore keep the rejection of this kind of pixels.

2.4.2 Mean or Median brightness?

During the performance analysis of our process, we noticed that the pixel collapse method better detects the shortest lines when we measure the median brightness of the pixels instead of the average brightness, in a given direction. Indeed, the median value is less sensitive to noise than the average value and allows us to access satellite tracks much shorter than the limit obtained by measuring the average brightness.

In what follows, the median value is used for the brightness calculation.

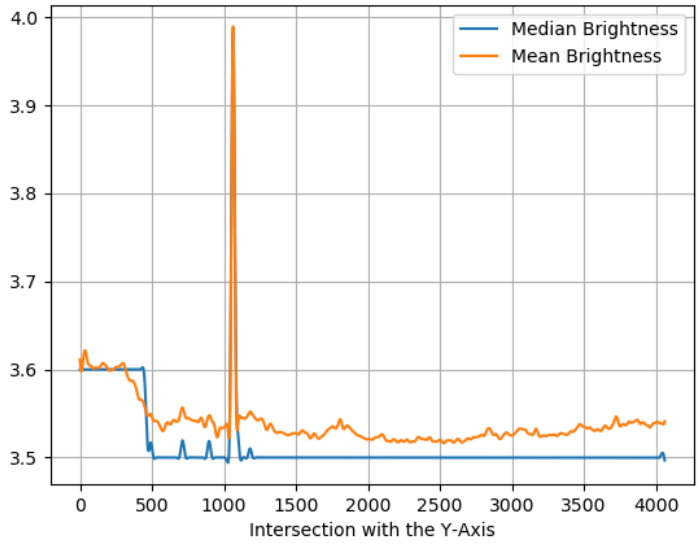


Figure 2.7: Mean (in orange) vs Median (in blue) brightness: Here we used the same image and the same dirction α to do the pixel collapse. The median brightness in a given direction α is less sensitive to small variations in brightness. the peaks are much more pronounced and easier to identify

It is also possible to use the third quartile in order to detect shorter satellites tracks. However, the number of false positives grows exponentially with q. Therfore, in all that follows, the median value is used for the brightness calculation.

2.5 False positives

Please note that the method does not detect the tracks themselves but rather detects their effects, in particular any significant local variation in luminosity. As a result, it may be that various causes not necessarily linked to tracks of objects such as an increase in brightness due to the malfunction of the sensors or alignment of stars etc., can consequently generate false positives.

There are several causes that generate false positives. The main reasons are:

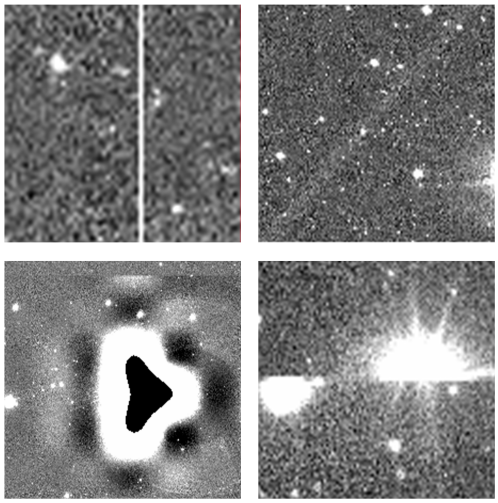


Figure 2.8: Main causes of false positives. (a) Bad columns. (b) Lens flare effect. (c) and (d) Malfunction of sensors.

2.5.1 Bad columns

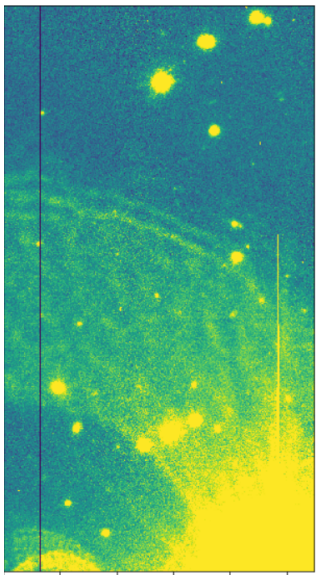


Figure 2.9: On the right, vertical white (yellow) line and on the left, vertical black line

The bad columns correspond to a very homogeneous vertical line in the image. There are two types of vertical lines: black lines and white lines.

Black lines have zero intensity on a gray scale and are therefore easily detectable. The white lines, on the other hand, have a similar intensity to that of the satellite tracks we are looking for. Finding these dead pixels by thresholding alone, even after performing a z-scale is not at all effective.

However, we notice that these white lines have a width of a few pixels which will later allow them to be differentiated from the tracks of real satellites having about ten to twenty pixels in width on average.

2.5.2 Lens flare

Lens flare refers to a phenomenon where light is scattered or flared in a lens system, often in response to a bright source, producing a sometimes undesirable artifact within the image. This happens for example through internal reflection and forward scatter from material imperfections in the

lens. Flare is particularly caused by very bright light sources like bright stars or supernovae. In some cases, the generated lines can be confused with satellite tracks by pixel collapse

2.5.3 Sensors' imperfections

Defective pixel, or a time between two images shorter than the time required for a saturated sensor to return to its initial state... may be in some cases the cause behind the detection of false positives.

2.6 Satellite identification

We use predictsat in order to identify the detected satellite and to access its name, its orbits parameters etc in the case of a match.. The *astropy* library is used in the process to convert $x - y$ coordinates to equatorial coordinates (right ascension α, declination δ).

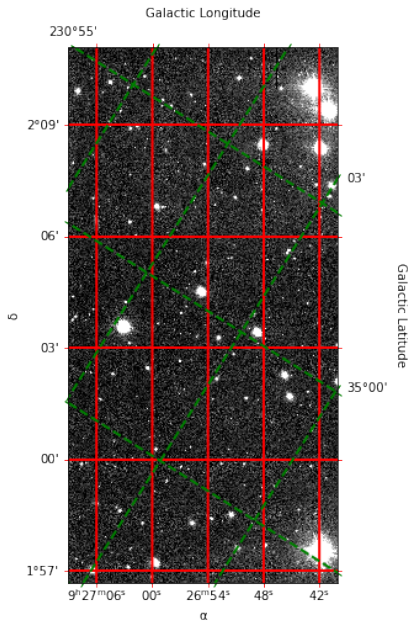
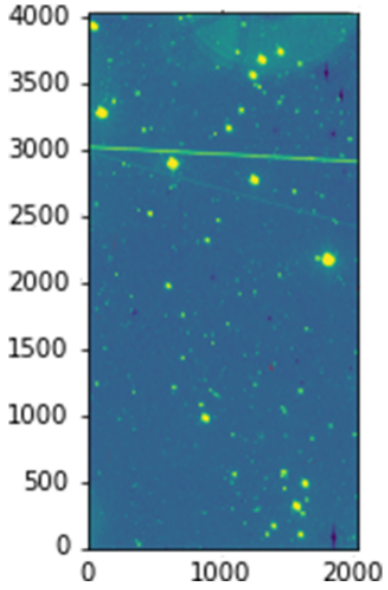


Figure 2.10: equatorial coordinates

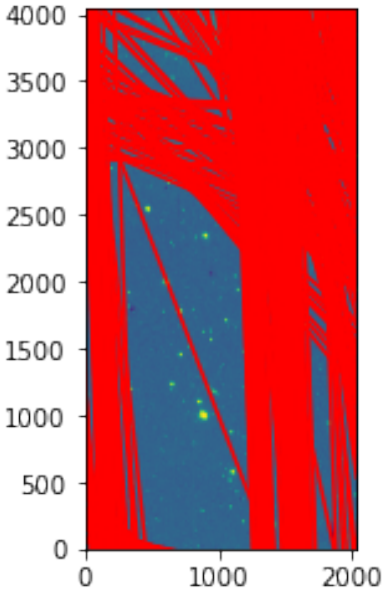
Chapter 3

Results

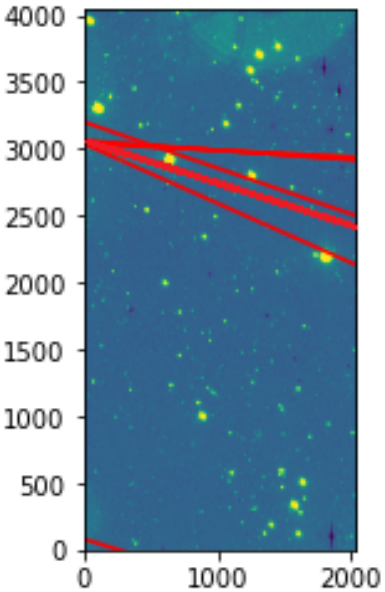
3.1 Results



(a) Step 1: Generate the image.



(b) Step 2: Detect every peak in brightness.



(c) Step 3: Adjust the parameters and compare with the rest of the image.

The results obtained are summarized in this table:

Number of images ($16k \times 16k$) analyzed	63
Number of blocs ($2k \times 4k$)	1134
number of observation days	15
Number of visible satellites detected	33
Number of visible satellites detected with a match	7 ($\approx 21\%$)
Number of visible satellites / false positives detected	45
Runtime per image	107 min

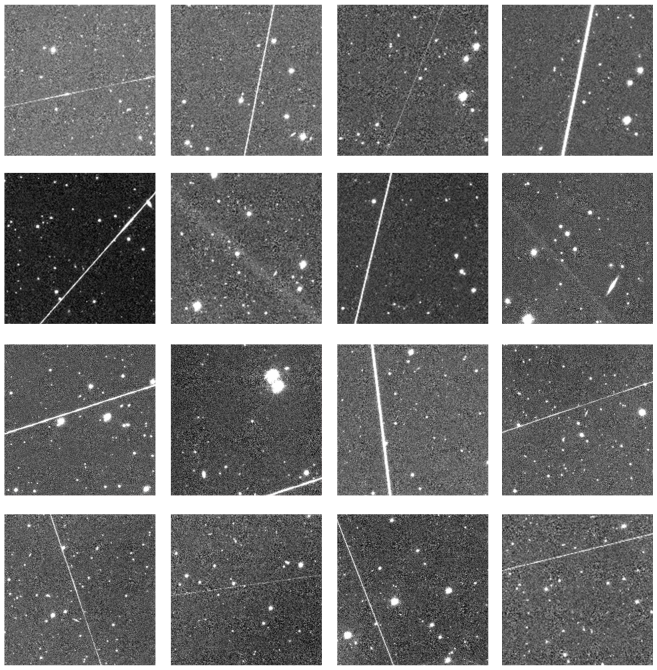


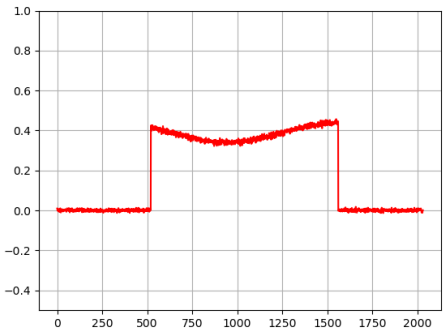
Figure 3.2: Example of detected satellite tracks

We notice the rate of detected satellites which match exactly with the predict-sat predictions is quite low (22%). Also, the tracks predicted are sometimes not perfectly aligned with the tracks visible on the image. The error can be related to the fact that the epoch of the satellite's TLE is too old. The TLEs

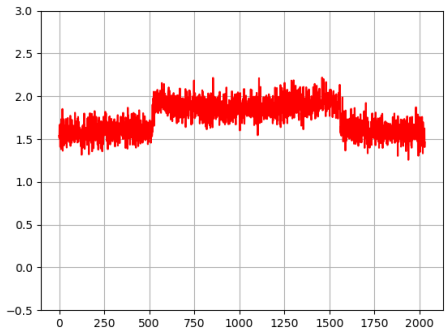
were obtained from *space-track.org*, which is a website that tracks satellites. Not all objects are tracked at the same rate, therefore some objects might have slightly older epochs, leading to error if the satellite's trajectory has changed recently. Also, TLEs themselves have limited precision and only give you an estimation of the position of the satellite

3.2 Rotation speed

By analyzing the evolution of the brightness along the track of the satellite, we are able, in theory, to determine the speed of rotation of the satellite around itself by performing a Fourier series decomposition of the detected signal, since the brightness of the satellite is directly related to the nature of its outer surface. However, due to the non-negligible noise, it is impossible to detect small variations in brightness without using a reference image capable of reducing the noise considerably.



(a) Evolution of the brightness of the satellite along its track using a reference image



(b) Evolution of the brightness of the satellite along its track without using a reference image

3.3 Detection limit

In order to test the performance of our process, we will seek to determine the limit values of brightness and length of satellite tracks so that they are correctly detected. To do this, we add satellite tracks by hand and we vary the length in pixels of these lines, as well as their opacity.

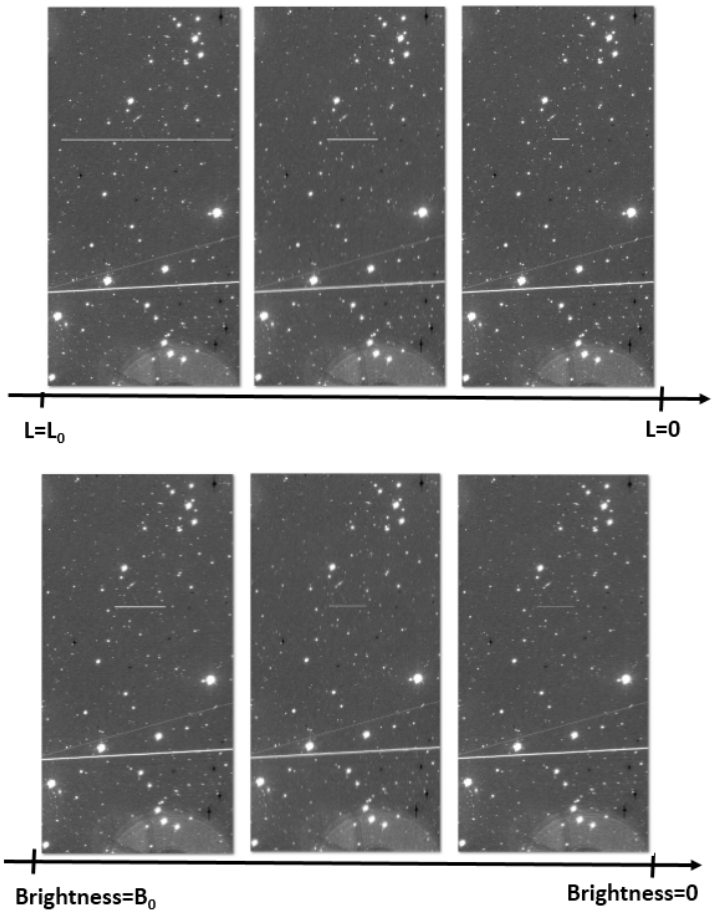


Figure 3.4: We add by hand a satellite track (in a middle, $\alpha = 0$). We vary its length and its brightness.

Let $B(i, j)$ be the brightness of pixel (i, j) after adding the satellite track and $S(i, j)$ be the brightness of pixel (i, j) before adding the satellite track, the opacity δ is defined as:

$$B(i, j) = \delta \times 1 + (1 - \delta) \times S(i, j)$$

δ is therefore a cursor between the value of the brightness of the image before the addition of the trace and the maximum value 1 (that of a saturated pixel). δ is proportional to the relative brightness compared to the sky $\Delta S(i, j) =$

$B(i,j) - S(i,j)$:

$$\delta = \frac{\Delta S(i,j)}{1 - S(i,j)} \propto \Delta S(i,j)$$

We plot the minimal length of the satellite required in order to be detected using our pixel collapse method as a function of the opacity:

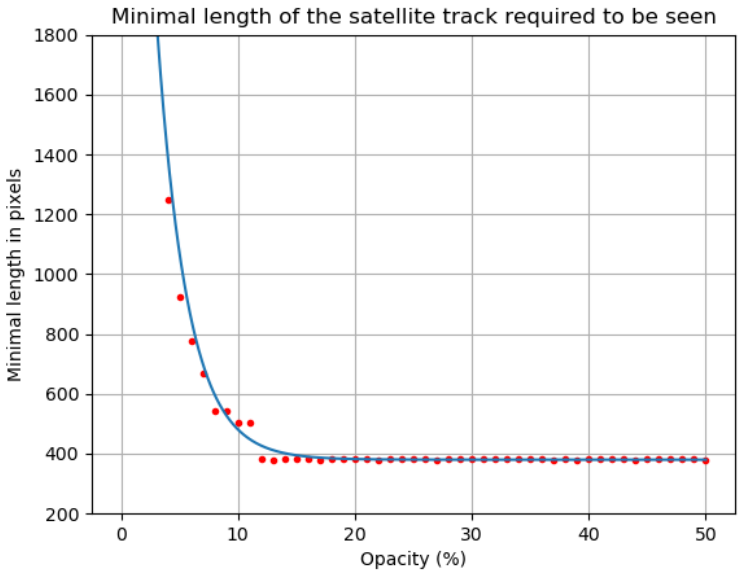


Figure 3.5: The minimal length of the satellite required in order to be detected as a function of its opacity, and therefore its brightness. The brighter the satellite, the greater its opacity and the smaller the minimum length for it to be detected.

We notice that:

1. If the satellite is bright enough ($\delta > 15\%$), the minimum length required does not vary since the satellite is already visible enough. The calculation of the variation of median brightness takes into account just the fact that the luminosity is above a well determined value (that of the median brightness of the sky) and makes absolutely no difference between all the values of brightness greater than this median.

2. For opacities of less than 15%, the fainter the satellite, the smaller its opacity and the smaller the minimum length to be detected. If $\delta < 3\%$, the minimum length required is greater than the image size, which is impossible. It follows that the method detects only the satellites having an opacity greater than 3%, which gives us a lower bound for the brightness of the satellites detected by the method:

$$\delta \geq 3\% \iff \Delta S(i,j) \geq 0.03(1 - S(i,j)) \sim 0.03(1 - \mu)$$

Where μ is the average brightness of our image (about ~ 0.35 on a grayscale). Therefore, this method is able to detect variations of approximately 0.02 in brightness.

3.4 Statistics

3.4.1 Same satellite, multiple blocs

In the case where we have an image composed of n blocks, a segment is defined by the choice of two points M_1 and M_2 randomly on the image. The probability that the two points belong to the same block and therefore that the segment is contained in a single block B_i is:

$$p = \sum_{i=1}^{n} p(M_1 \in B_i) p_{M_1 \in B_i}(M_2 \in B_i)$$

The choice of the first point is independent of the choice of the second point, therefore:

$$p = \sum_{i=1}^{n} (p(M_1 \in B_i))^2 = \sum_{i=1}^{n} \frac{1}{n^2} = \frac{1}{n}$$

Therefore, for $n = 32$, it is expected that only $1/32 \approx 3\%$ of the detected satellites belong only to a single block. Experimentally, we find a fairly close result: 6%. It is therefore interesting to compare the results obtained for each block, since the satellites tend to cross several blocks at the same time. This will make it possible to eliminate several false positives, in particular most of the tracks which only belong to a single block.

3.4.2 Orientation and length of detected tracks

Length of detected tracks:

Figure 3.6: distribution of satellites according to the size of the track in pixels

The average distance ℓ between two points randomly chosen in a square with side a is:

$$\ell = \frac{2 + \sqrt{2} + 5\ln(1 + \sqrt{2})}{15} a$$

In our case, $a = 16k$ pixels, so We expect that the average distance of the tracks, if we assume that the size of the satellite tracks is completely random, is:

$$\ell \approx 8.3k$$

According to the results obtained with our pixel collapse method:

$$\ell \approx 12.4k$$

The small deviation can be explained by the fact that satellites with short tracks are difficult to detect compared to satellites with longer tracks. This asymmetry in probability therefore increases the expectation of the size of the satellites.

Orientation of detected tracks:

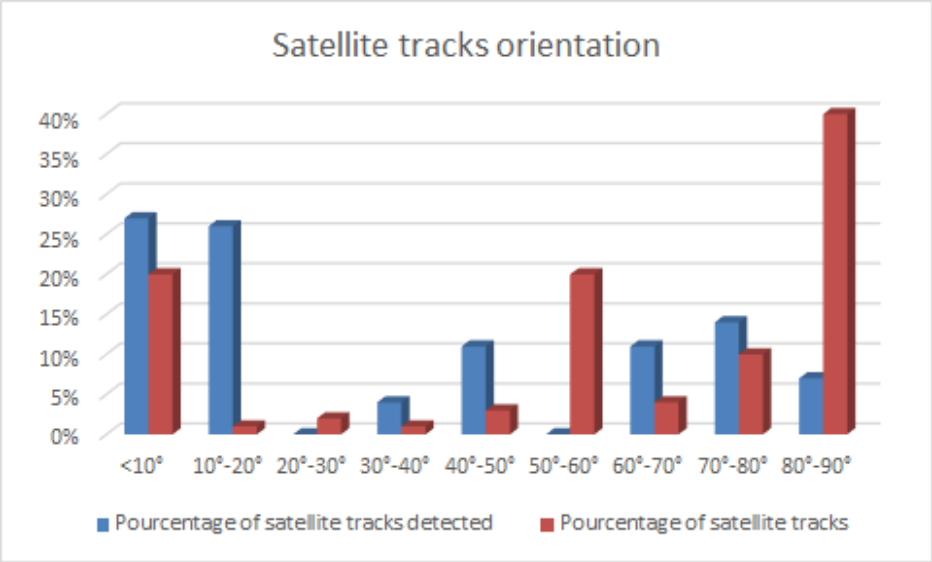


Figure 3.7: distribution of satellites detected according to the track orientation (in blue) vs distribution of satellites according to the track orientation (in red)

By analyzing the inclinations of about 3000 satellites appearing in the database that we have, we conclude that we expect to have most of the satellites with geostationary (angle close to 0 degree) or polar (angle close to 90 degrees) orbits. The results obtained are quite close although they do not coincide exactly with what we expect. This deviation can be explained by the fact that the tracks that we detect do not often correspond to satellites but also space debris sometimes even uncategorized. It is also possible to say that the whole of the analyzed images constitutes a fairly small sample (63 images).

Please denote that here we have reported all the angles over the interval $[-90, 90]$ and we only considered the absolute values of these angles.

3.4.3 Track width

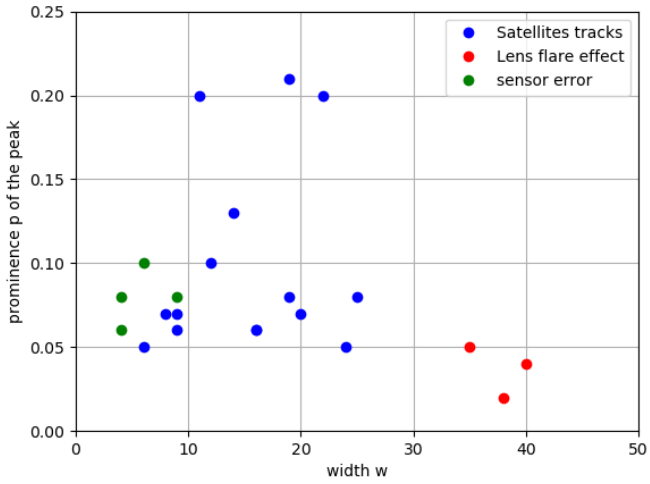


Figure 3.8: distribution of satellites according to the track orientation

Each type of tracks (Satellites, Lens flare, bad columns) has a characteristic width. We can reduce the number of false positives by eliminating tracks with a width $w < 5$ pixels ot $w > 30$ pixels.

3.5 Possible improvements

Detecting short satellites tracks
The pixel collapse method as I built it in this project detects very well, if not perfectly, the satellites having a rather long track ($> 8k$ pixels). When the track is short enough, and especially when the signal to noise ratio SNR is quite low, the algorithm has difficulty in detecting these traces and distinguishing between them, and the stars not far enough from each other. Improving the efficiency of the algorithm on short track satellites will reduce the average size of the detected traces from $12.4k$ pixels (currently) to $8.3k$ pixels (in the perfect case).

Estimate correctly the pourcentage of false positives
How much is the percentage of false positives among the satellites detected?

As explained above, we detect 33 satellites that can be seen with the naked eye or having a match with the predictions of PREDICTSAT, but 45 satellites of which we are not 100% sure of their existence. All we can say that the upper bound of the false positive rate $\eta_{f,+}$ is therefore

$$\eta_{f,+} = \frac{45}{45 + 23} = 66\%$$

Complexity

The script's algorithm is far from being perfectly efficient. The method seems to give fairly accurate results. However, by analyzing and projecting the pixels more efficiently, it seems to me that it would be possible to reduce the computation time a little more.

Chapter 4

Conclusion

Since satellites or space debris travel very fast, their trajectories on an astronomical image are almost linear, we could detect satellites by projecting the pixels in a specific direction. The pixel collapse algorithm makes it possible to detect traces of satellites or space debris in astronomical images and make the link with the catalog of satellites. Detection depends mainly on the brightness of the track, its length and the signal to noise ratio SNR of the image. The results obtained are fair enough but can be improved.

The algorithm perfectly detects satellites bright enough to be seen with the naked eye. It detects up to 2% variations in brightness relative to the sky, but the number of false positives increases as one wishes to detect satellites with short and not very bright tracks. It is quite complicated to correctly estimate the rate of false positives detected but it must be less than 66%.

The algorithm is not very efficient if one wishes to measure the evolution of the variation in the brightness of the satellite in order to be able to access its speed of rotation, due to the significant noise in the astronomical images that we used. It seems to us that it is impossible to remedy this problem without using a *reference image* in order to eliminate any source of noise. Several other data concerning the orbit parameters are accessible in the case where we have a match with a satellite of the database that we used, the *SpaceTrack dataset*.

From a personal point of view, this project taught me a lot. It was the opportunity for me to work on real astronomical images, which was fun

and rewarding. I want to thank everyone who helped me and made this project possible. Thanks to prof. Jean-Paul Kneib for the supervision of the project and all his help. I would also like to thank prof. Frédéric Courbin, Cameron Lemon and Salzmann Mathieu for their help and ideas during our Zoom meetings. I would like to thank Yann Bouquet with whom I had the pleasure of working on this project.

Bibliography

[1] Olivier R. Hainaut and Andrew P. Williams. *Impact of satellite constellations on astronomical observations with ESO telescopes in the visible and infrared domains*. European Southern Observatory, 2020.

[2] Direkci Su. *Studies on satellite detection with difference imaging.* 2020. `drive.google.com/file/d/10a8wd31LoKL-9SeEkw9STMBkoMAp2Yg3/view`

[3] Schizas Alexis Georges. *Satellite tracks prediction using a TLE database.* 2020. `https://github.com/alexisschizas/predictsat`

[4] Information on Light Pollution, Radio Interference, and Space Debris. 2020. `https://aas.org/about/governance/light`

[5] IRAF help page. Zscale Algorithm. 2020. `astro.uni-bonn.de/sysstw/lfa_html/iraf/images.tv.display.html`