Delineating Satellite Tracks in Astronomical Images

Yann BOUQUET

273827

Submitted in partial satisfaction of the requirements for the degree of

MSc Data Science

School of Computer Science

EPFL

2020

Acknowledgements

I would like to express my sincere gratitude to Prof. Mathieu Salzmann who trusted me for this project and for his continuous support. I would also like to express my deepest appreciation to all the people who accompanied me throughout my semester project, namely Prof. Jean-Paul Kneib, Prof. Frédéric Courbin, Dr. Cameron Lemon and Dr.Kozinski Mateusz. Finally, I would like to thank Wajdi Imliki, with whom I had the pleasure of working on this project.

Abstract

We present two approaches to detect and extract information about satellite streaks in single epoch images produced by OMEGACAM on the VLT Survey Telescope. We distinguish long tracks from high velocity objects, usually crossing an important part of the image, and short tracks from low velocity objects. The first method is based on image processing technique. By applying filters and morphological transforms on the image we manage to detect the long satellite streaks with Hough transform technique. Facing the limitations of this approach for short tracks, we propose a neural network model inspired from the UNet model, for image segmentation, and train it on synthetic data to identify pixels belonging to satellites streaks with a f1 score of 95%. We show the feasibility of our methods by using example images the gravitationally lensed quasar fields SDSSJ0924+0219.

Table of Contents

1 Introduction **1**

2 Inputs **3**

3 High Velocity Objects **5**

3.1 Method . . . 5

3.1.1 Method Summary . . . 5

3.1.2 Morphological Tools... . . . 6

... For binary images . . . 7

...For gray-scale images . . . 8

3.1.3 Transient Effects Removal . . . 9

Elements . . . 11

Strategies for transient effects detection . . . 11

3.1.4 Light Blobs . . . 15

3.1.5 Hough Transform . . . 16

Canny filtering . . . 16

Thresholding the number of aligned points . . . 17

3.1.6 Distinguish the satellites . . . 18

3.2 Final Result . . . 21

3.2.1 Predictsat . . . 22

3.2.2 Further Corrections in the algorithm . . . 23

4 Low Velocity Objects **25**

4.1 Motivation . . . 25

4.2 Synthetic Image Generation . . . 26

4.3 UNet for Image Segmentation . . . 28

4.3.1 Model . . . 28

4.3.2 Training . . . 30

Dropout . . . 31

Batch Normalization 32
4.4 Final Results 32
4.4.1 Patch-wise Performance 32
4.4.2 Pixel-wise Performance 33
4.4.3 Future Work 34

5 Conclusion **37**

Appendix A Summary of the UNet Model adapted for this project **41**

Appendix B Other real satellite streak sample for testing the UNet model **43**

Chapter 1

Introduction

This report presents methods to detect object streaks in astronomical images. Due to the colonization of near Earth space by satellites, we observe an increasing need to know with high degree of accuracy the orbital parameters of satellites and debris. As those parameters constantly evolve with time because of some orbital disturbances, it is necessary to have tools to periodically observe these object and update their parameters. As a consequence, detection of straight lines in astronomical images is critical for detecting high velocity objects (relative to the exposure time), faint satellites streaks and slow motion object (also relative to exposure time).

An earlier method has been proposed for detecting such objects in images and is based on difference image analysis [1] which provides an optimal subtraction between an image and a reference image. Using difference image analysis for this problematic implies many exposures in order to observe the objects which we are looking for. Also by working with a combination of many images, the time frame was traded for decreasing the noise. Thus it prevents from gaining additional insight about the objects that are detected such as their speed, brightness's variations in their streaks, etc. Consequently, in order to create a pipeline that takes as input a single exposure of the telescope, the goal of this project is to propose new image processing methods to detect straight lines in astronomical images without difference imaging.

We will distinguish for this project two kinds of objects : high velocity objects (which creates long streaks across the image) and slow motion objects (which creates very

short streaks in the image). This motivates to study two distinct approaches. The first one implies image processing mainly using mathematical morphology tools and the Hough transform to detect long lines in the images. The second one involves deep neural networks for image segmentation on a synthetic set of images.

Chapter 2

Inputs

The satellite streaks studied in this project lie in the fields of a high-cadence lensed quasar monitoring project with OMEGA-CAM on the VLT Survey Telescope which takes approximately 5 minutes of exposure to generate one image. This image, or the input of our process, is a 16 000 x 16 000 pixels mosaic consisting of the concatenation of 32 4000x2000 pixels contiguous blocks separated by bands of NaN values. As a pixel is encoded as a 64-bit float, each mosaic represents approximately 2GB of data.

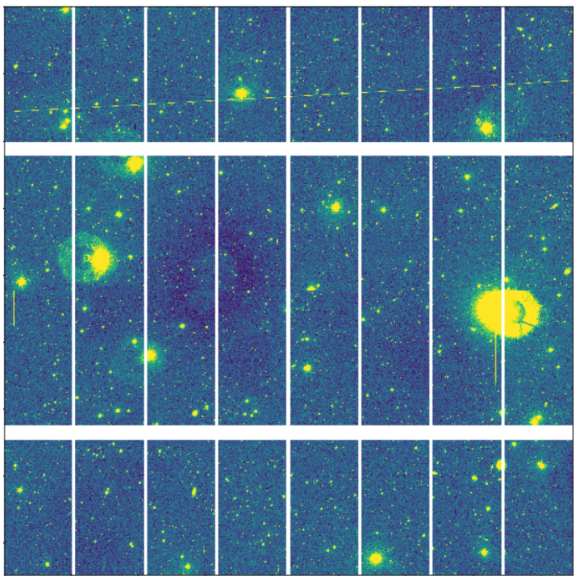


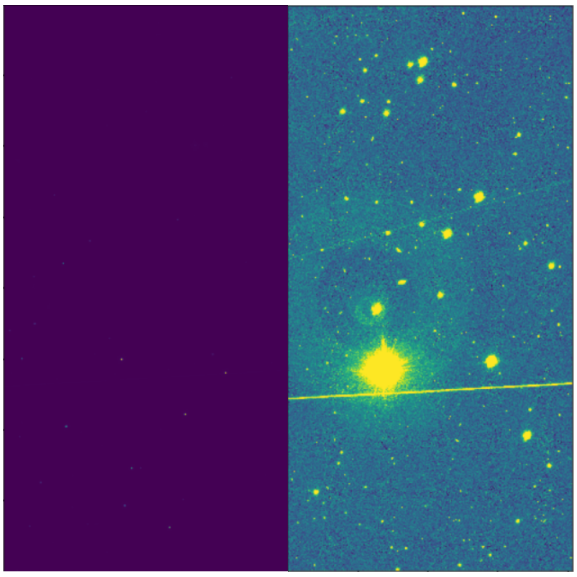
Figure 2.1: FITS Image : Mosaic of 32 blocks

The method proposed here is to analyze each block independently. This also allows us to parallelize the process to reduce the computation time. So we will detail the whole process by observing a single block of 4000 x 2000 pixels.

Min	Max	Mean	Standard Deviation
−37.5477	111075	697.561	1051.48

Table 2.1: Pixel Intensity Distribution in a Mosaic

In this block, the intensity of pixels sweep a wide range of values as referred in Table 2.1. In a gray-scale the image thus appears black, as in Figure 2.2a because of the huge intensity peaks. So in order to be able to process the image in the classical gray-scale, we perform a rescaling of the pixels values 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 in Figure 2.2b.



(a) Unscaled Block (b) Scaled Block

Figure 2.2: Iraf's ZScale algorithm applied on a block. (b) We can see two streaks in the block.

Chapter 3

High Velocity Objects

3.1 Method

3.1.1 Method Summary

Our image processing method for retrieving streaks such as the ones that can be observed in Figure 2.2b. This method presented in Figure 3.1 is based on the Hough transform method which analyses the alignment of points in a 2D space. As a consequence a primary process will be done to remove objects that we call bad columns and that can be interpreted as vertical lines in Hough space. Also filtering and morphological transforms on the image will be done in order to highlight linear objects in the image and remove as much light blobs as possible. We will apply a Canny filtering in order to remove the smaller light blobs before detecting the lines with the Hough transform technique. At the end, many lines will be detected. The assertion here is that every detected line is generated by a moving object's streak existing in the image. However the orientation of the line and its position can slightly differ from the associated streak. Furthermore many lines can be detected for the same streak. As a consequence, the last part of the method consists in regrouping the detected lines that have similar orientations and positions in the image and analysing the neighborhood of this groups to retrieve an accurate orientation and position of the streak.

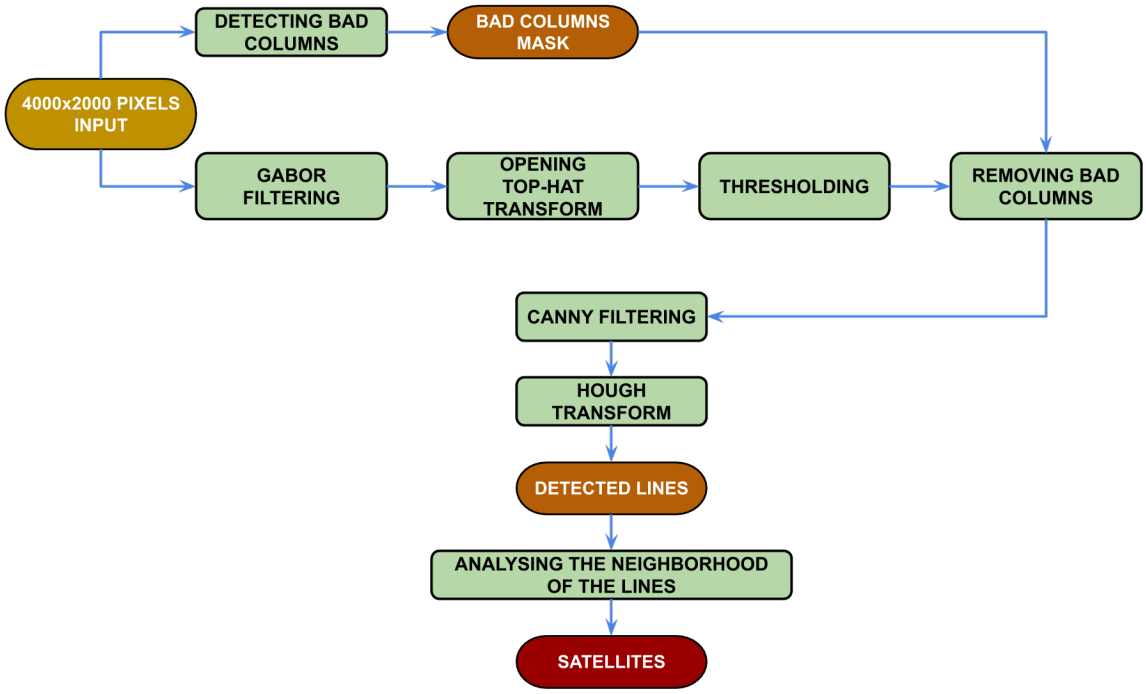


Figure 3.1: Flowchart representing the method for detecting streaks in an image.

3.1.2 Morphological Tools...

For the method, mathematical morphology [2] is rapidly used on the images in the process. Referring to a branch of nonlinear image processing, mathematical morphology focuses on geometrical structures within an image by analysing quantitatively how a structuring element fits or not in an image. Here is thus presented some theoretical background about it.

... For binary images

The **erosion**, or one the fundamental operations of mathematical morphology, of a set A (geometrical structure in the image) by a set B (structuring element) is defined as the following:

$$A \ominus B = \{x : B_x \subset A\}$$

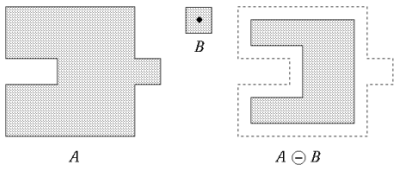


Figure 3.2: Erosion[1]

where $B_x = \{b + x : b \in B\}$ defines a translation of the set B by a point x.

The **dilation**, or the other fundamental operations of mathematical morphology, of a set A (geometrical structure in the image) by a set B (structuring element) is defined as the following:

$$A \oplus B = \left(A^c \ominus \breve{B}\right)^c$$

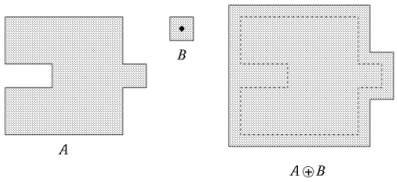


Figure 3.3: Dilation[1]

where $\breve{B}= \{-b : b \in B\}$ the 180-deg rotation of B about the origin and A^c is the set-theoretical complement of A.

To resume, considering both operations, it consists in bypassing a geometrical object of the image by a structuring element by making the edge of the element of the image correspond to the center of the structuring element. At this edge all the points of the image object that are included in the structuring element in the case of erosion are removed. In the case of dilation, all the points of the structuring element excluded from the object are added to the object. The complete operations are obtained by operating these processes at any point belonging to the edge of the image object.

[1]http://www.inf.u-szeged.hu/ssip/1996/morpho/morphology.html

The **opening** is a combination of the two fundamental operations. Considering two images A and B, it is defined by:

$$A \circ B = (A \ominus B) \oplus B$$

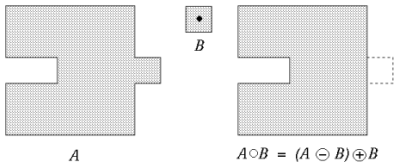


Figure 3.4: Opening[1]

which corresponds to the union of all translations of B that fit inside the input image.

The **opening top-hat transform** is an operation between two images A (image) and B (structuring element) defined by :

$$A \overset{\wedge}{\circ} B = A - (A \circ B)$$

...For gray-scale images

Despite the fact that the main concern in this project is image processing, the theory for signals will be developed here in order to keep the notations as simple as possible and to present simple illustrations. It appears that the theory is independent of the dimensionality of the application domain. Considering a signal f defined on a domain, $f(x)$ denotes the functional value at x.

The **erosion** of a signal f by a signal g (structuring element) is defined at a point x by:

$$(f \ominus g)(x) = max\{y : g_x + y \leq f\}$$

where g_x is the translation of g defined by $g_x(z) = g(z - x)$ at z. In other word we slide the structuring element g spatially so that its origin (which for signals is the Euclidean-plane origin relative to the structuring element) is located at x, and then we find the maximum offset y such that the element still lies beneath the signal f at any point.

The **dilation** of a signal f by a signal g is defined at a point x by :

$$(f \oplus g)(x) = min\{y : -\breve{g_x} + y \geq f\}$$

[1]http://www.inf.u-szeged.hu/ssip/1996/morpho/morphology.html

where $\breve{g}$ is the reflection of g defined as $\breve{g}\ (x) = g(-x)$ at x.

The **opening** of a signal f by a signal g is defined by :

$$f \circ g = (f \ominus g) \oplus g$$

The **top-hat transform** of a signal f by a signal g is defined by :

$$f \overset{\wedge}{\circ} g = f - (f \circ g)$$

It extracts elements of an image smaller than the considered structuring element and brighter than their surroundings. To do so this operation considers the difference between the input image and its opening by some structuring element in order to mark narrow peaks while not marking wide white disks in the image.

This theory is used in order to isolate the satellite streaks as much as possible from elements that can lead our method to give false positives in the detection of linear object.

3.1.3 Transient Effects Removal

In the images it is observed that vertical lines can be detected during the Hough transform, thus generating false positives in the detection of the streaks. This vertical lines comes from different elements caused by transient effects in the image. Many strategies are proposed to grasp all these elements that can be in the block and create a mask to ignore the pixels belonging to these elements during the Hough transform. This proposed method will also grasp some pixels that do not belong to these elements thus creating noise in the mask. So an iterative process will remove those pixels from the mask.

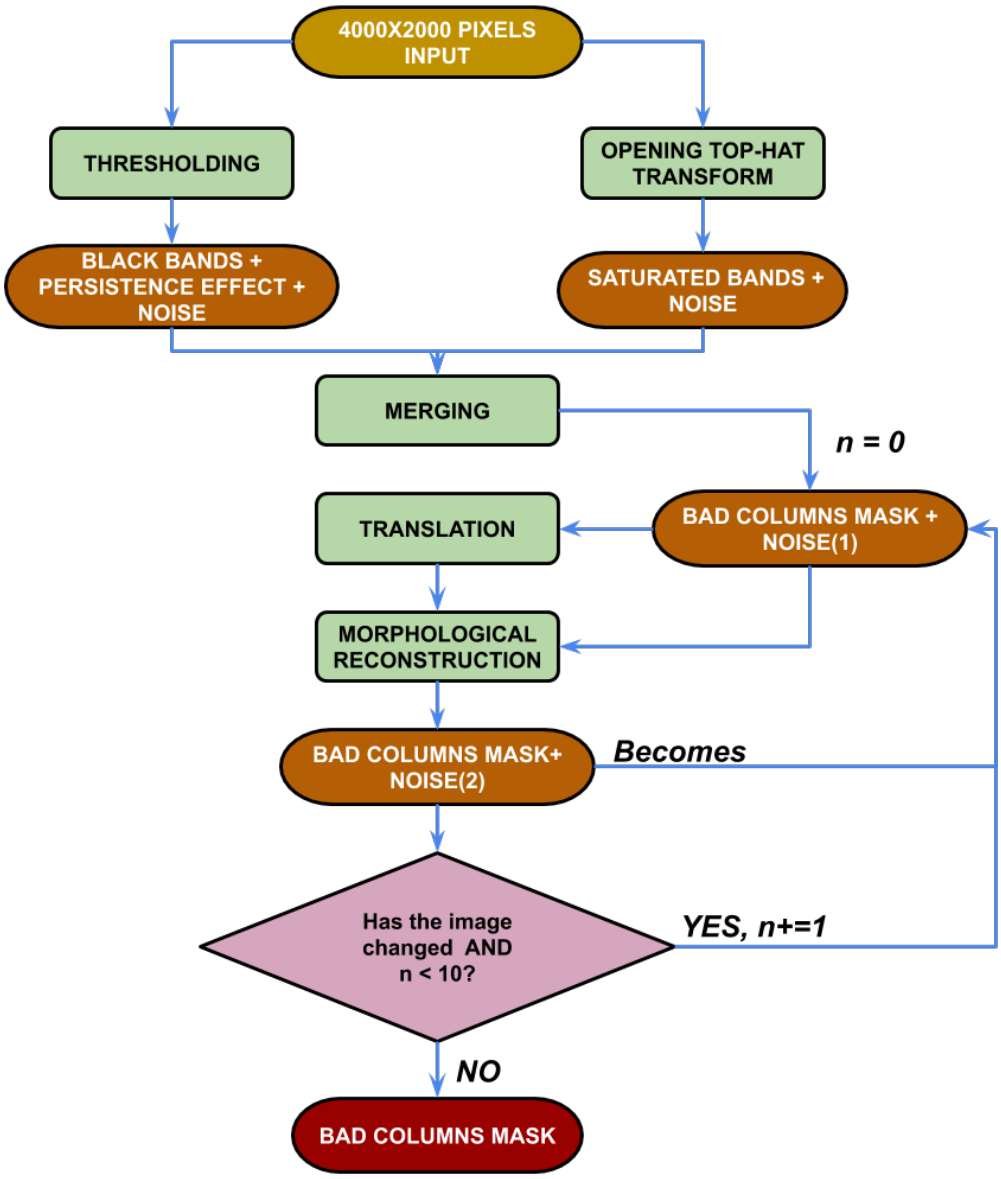


Figure 3.5: Flowchart representing the method for removing bad columns in an image

Elements

These elements which generate those vertical lines can be classified in different categories.

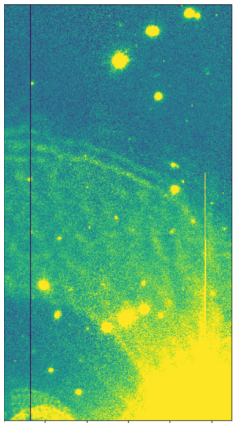


Figure 3.6: On the right: vertical white (yellow) line. On the left: vertical black line

These lines can come from dead pixels which form black columns (pixels with null intensity) in the image but also from saturated pixels which form white columns. The last cause of this vertical line is a persistence effect on the CCD (Charged-Couple-Device) sensors of the telescope. Bright sources can be overexposed implying a strongly localized increase in the dark rate, or after-image of the bright source during the next exposures. Thus they appear as very dark sources in the image.

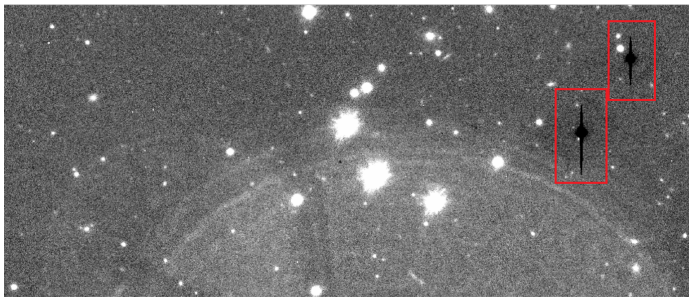


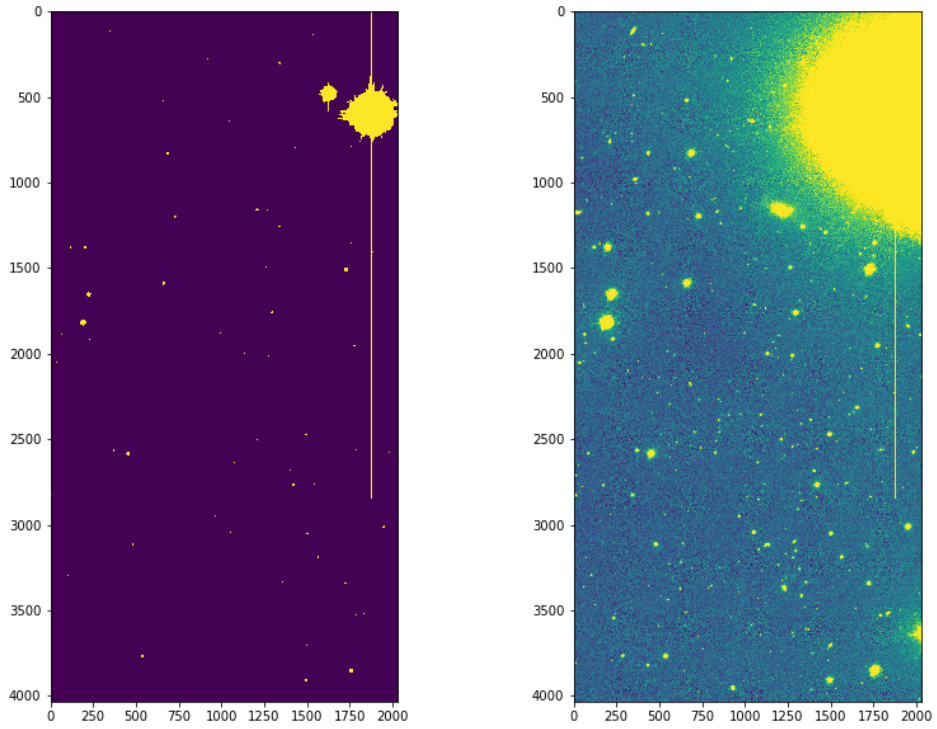
Figure 3.7: Persistence effects

Strategies for transient effects detection

Dark columns and persistence effects are easily detectable because they have very low intensity on a gray-scale. So they can be found by applying a fixed threshold on the image. By creating this mask we obtain these columns and some noise (Figure 3.9a).

Concerning the white columns which consists of saturated pixels, they can be caused

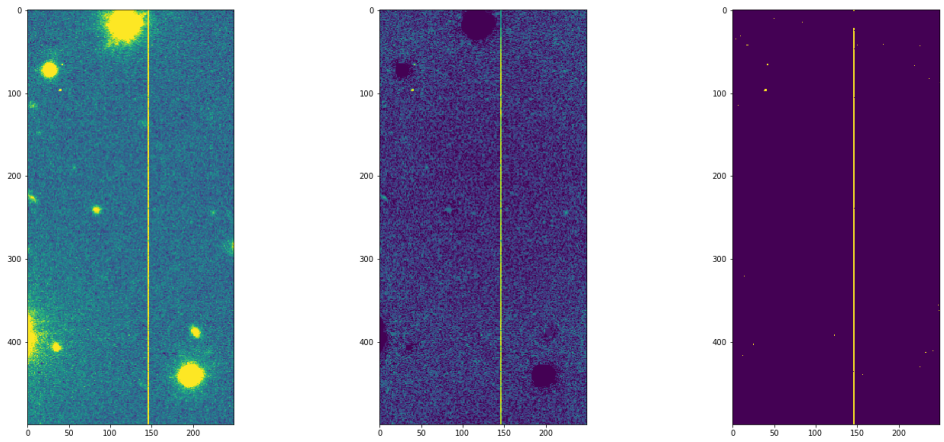
by very bright sources: the pixel intensity is abnormally high and thus these "outliers" pixels can be recovered using a threshold on the unscaled block.



(a) "Outliers" pixels in unscaled block (b) Scaled block

Figure 3.8: Example of saturated "outliers" pixels forming bright columns

The other saturated pixels that also form white columns in the image can have an intensity closed to the intensity values of the satellite tracks we are looking for. After the zscale rescaling, they forms vertical tracks of maximum intensity, as can be satellites streaks. Consequently, searching for these "inliers" pixels by thresholding alone is hardly feasible. On the other hand, we notice that these lines of dead pixels are very thin (1 to 3 pixels thick) and much thinner than the very high intensity satellite tracks. We will therefore be able to use the top-hat transform with a 8-pixel diameter disk as a structuring element to isolate the smaller elements. Among these elements we find the columns of white pixels as presented in Figure 3.9.

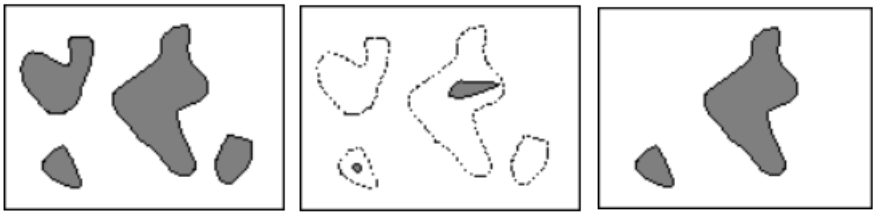


(a) Gray-Scale Image (b) Top-Hat Transform (c) Binary Mask

Figure 3.9: Top-Hat Transform for removing saturated "inliers" pixels

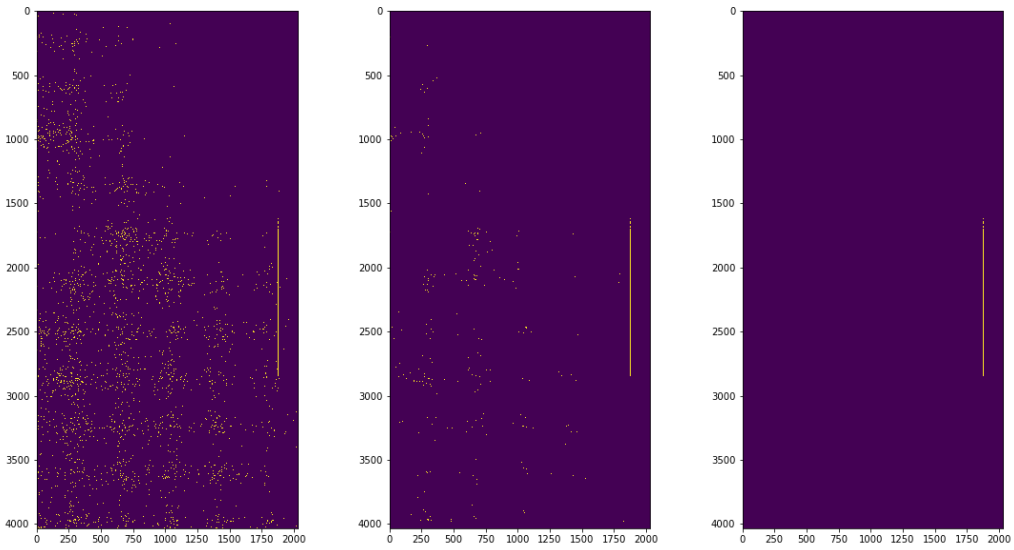
By coupling the different obtained masks we can make sure to recover all results of these transient effects. We will isolate them from the noise as presented in Figure 3.11 using their vertical property. Taking the previously obtained mask as input, we make a translation along the ordinate axis and recover the intersection between the input mask and this translation. In this way we remove a maximum of pixels that do not belong to the vertical line and recover a small part of the dead pixel trace. Thanks to a morphological reconstruction where we can see an example in Figure 3.10 taking as mask the result of the intersection and as reference image the original mask, we can recover the part of the vertical line that we have lost by the translation without reconstructing the totality of the pixels belonging to the noise. So the result of the morphological reconstruction contains a less dense noise and the same columns that we want to isolate. If this result becomes our new input mask, we can iterate the previously described process enough times to make sure to eliminate all noise from the mask. We notice that the translation must be done with a number of pixels small enough not to lose small tracks of dead pixels during the intersection operation and large enough to space the other pixels as much as possible to avoid reconstructing them during the morphological reconstruction. It

was decided to translate 50 pixels at each iteration, for 10 iterations to ensure that only traces are recovered each time.



(a) Reference Image (b) Mask (c) Reconstruction

Figure 3.10: Morphological Reconstruction for particles extraction[1]



(a) (b) (c)

Figure 3.11: Removing noise from the mask by an iterative method. (a) Original mask containing the columns and the noise. (b) Intermediate results of the iterative method : the noise is less dense. (c) Any noise seems to appear in the final result of the process.

The result is a binary mask representing these dead pixels that we will use later in the process.

[1] http://www.telecom.ulg.ac.be/teaching/notes/totali/elen016/node80$_m$n.html

3.1.4 Light Blobs

We will also use the top-hat transform to remove the light blobs. We know that for a 10-pixel diameter disk as the structuring element, we notice that the top-hat transform highlights a grid of points in the background of the image and this grid disturbs the line detection by Hough transform. This grid is already present in the background of the image and can be erased by a linear filter. It was decided to use Gabor filters to erase this grid. The goal is to take advantage of the properties of Gabor filters [3] and our knowledge of the precise structures of satellite tracks to both erase the grid and preserve the contours of the satellite tracks.

The Gabor filter (assumed to be centered at zero) is the product of a sinusoid and a Gaussian:

$$g(x', y', \lambda, \theta, \phi, \gamma) = \exp\left(- \frac{'x^2 + \gamma^2 y'^2}{2\sigma^2} \right) \cos\left(2\pi \frac{x'}{\lambda} + \phi \right)$$

where

$$x' = x\cos\theta + y\sin\theta, y' = -x\sin\theta + y\cos\theta$$

- λ is a wavelength (the number of cycles per pixel). It controls the width of the stripes in the filter;
- θ is an orientation (the angle of the normal to the sinusois). This corresponds to the orientation of the stripes in the filter;
- ϕ is a phase (the offset of the sinoid);
- $\gamma < 1$ is an aspect ratio producing the ellipticity. It controls the number of stripes in the filter.

By taking different θ we can highlight satellites that follow several possible directions. The goal is to take an average intensity value of all these results and take it as the input of a hough transform in order to remove the big luminous blobs without getting back the grid we had before. Thanks to the use of Gabor filters, the faint satellite

tracks are also highlighted by these filters and for the top-hat transform, the 10-pixel diameter disk is sufficient as structuring element.

We will ignore the background of the image thanks by thresholding the image. The threshold is a fixed value for any input that gave satisfying results so far.

3.1.5 Hough Transform

In a polar coordinate system with a defined origin, this technique defines any lines in this space by a vector with the following parameters in this system:
θ : the angle ;
ρ: the norm of the vector (the distance between the line and the origin of the system).
By considering a fixed point in this system, we can calculate all vector of parameters (ρ,θ) defining every line that goes through this point. We thus obtain one sinusoid per point called "Hough space". If the line associated with three points intersect, the place where they intersect in the Hough space corresponds to the parameters of a straight line connecting these two points.

Canny filtering

Knowing that we have already averaged the image using the Gabor filters, we will not perform a smoothing on the image as declared in the edge detection process. Moreover we apply the Canny filter on a thresholded image. So an additional motivation is that we don't want to spread out point sources which are sources of error for line detection by Hough transforms. Thanks to edge detection the remaining light spots will become circles with an insufficient number of aligned points to generate false detections while the edges of the traces of interest will remain present in the image generating lines easily detectable by a Hough transform.

Thresholding the number of aligned points

Because of the large image size and the important presence of point sources always present in the resulting image, we must impose a minimum number of points intersected by a line to consider that this this line belongs to a track.

Indeed the principal source of error can come from the diffraction spikes of very bright stars. These spikes have very similar properties with satellite streaks are not removed by the current method as we can see in the figure **??**. Their brightness and random orientation can be similar to some satellite streaks. Furthermore, a satellite streak can go through a bright star on the image, so this property is not enough to distinguish them from satellites. Although we could analyse the properties of many spikes that come out from the same light source, however nothing can exclude a satellite to pass by the center of a light source. However the spikes are still very short compared to a satellite streak in general.

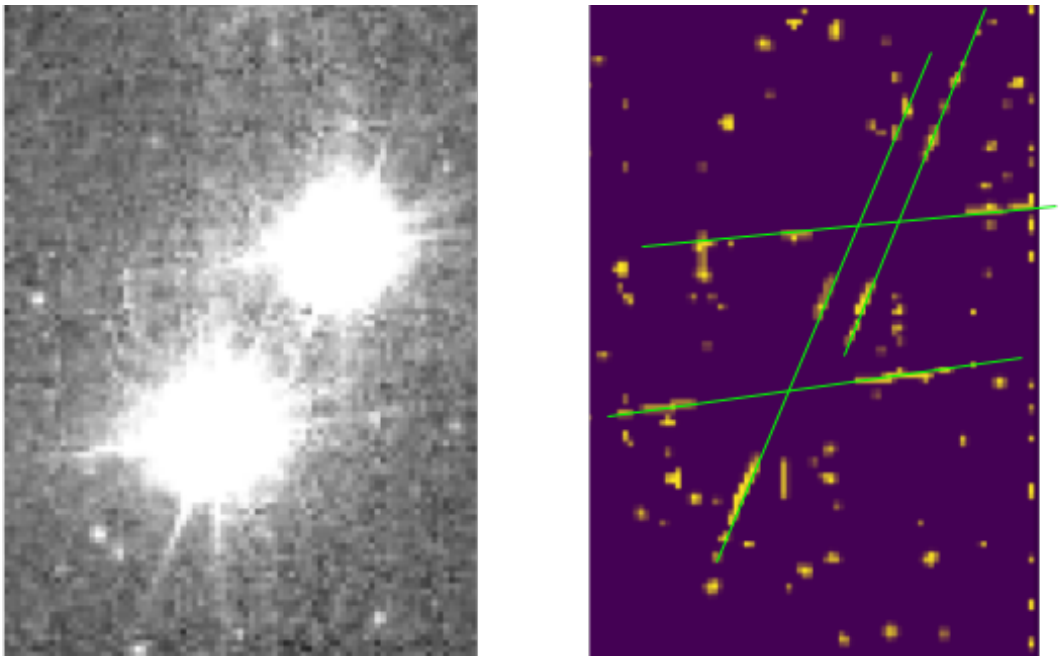


Figure 3.12: Pixels belonging to diffraction spikes that are given as input to the Hough technique. Green lines shows aligned pixels that can generate straight lines with Hough.

Thanks to the previous filters, we can set the limit to 200 pixels (empirical res-

ult) which is a low enough limit for high velocity traces. On the other hand, it is impossible to detect traces smaller than this threshold. However to overcome this problem we can consider smaller blocks of the image.

This threshold makes it possible to neglect the spikes coming from large light sources.

3.1.6 Distinguish the satellites

After obtaining the lines of interest using the Hough transform, we must correlate these lines with the existence of an object in the image. We must associate the lines of interest to each object in the image by comparing their slopes and their original ordinates.

Once this distinction is made, we can create one line per object by taking the median values of the slope and the intercept of each group and analyze the neighborhood of this line in the image to which a top-hat transform is applied to remove the largest luminous blobs. To do so, we will rotate this line around its original orientation and analyse the mean intensity of each group of pixels that has the same projection to any hyper-plan perpendicular to the rotated line as presented below.

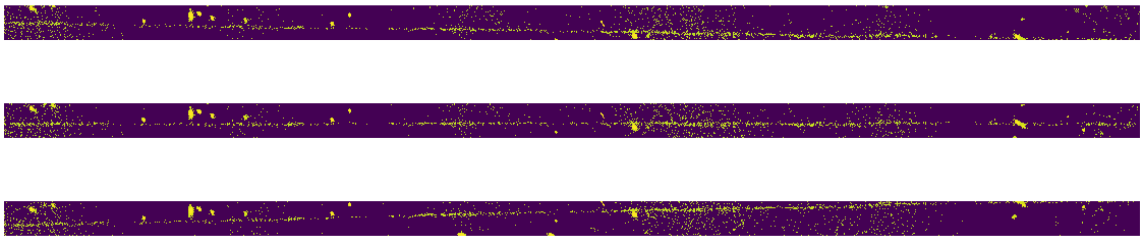


Figure 3.13: Neighborhood of three different orientations of the line detected by Hough because of the streak that can be seen in each image. We will look at the mean intensity of each row of the matrix represented by an image. We can see that the satellite streak is horizontal in the middle image

The idea is that the satellite that leads us to the detection of our previous line forms a peak of intensity in the neighborhood of this line for an orientation equivalent to the orientation of the satellite streak. So determining the right rotation of our line that leads to the greatest and thinnest peak of intensity will mean finding a satisfying

approximation of the orientation of the associated satellite streak. Considering this peak, we take the full width at half maximum in order to define the width of our streak.

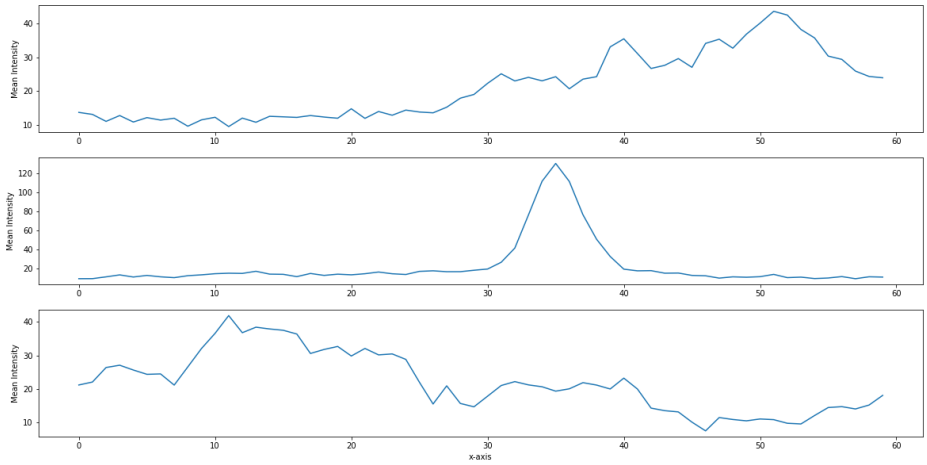


Figure 3.14: Mean intensity distribution along the rows of each image of the figure 3.13. We can observe the peak of intensity for the middle image.

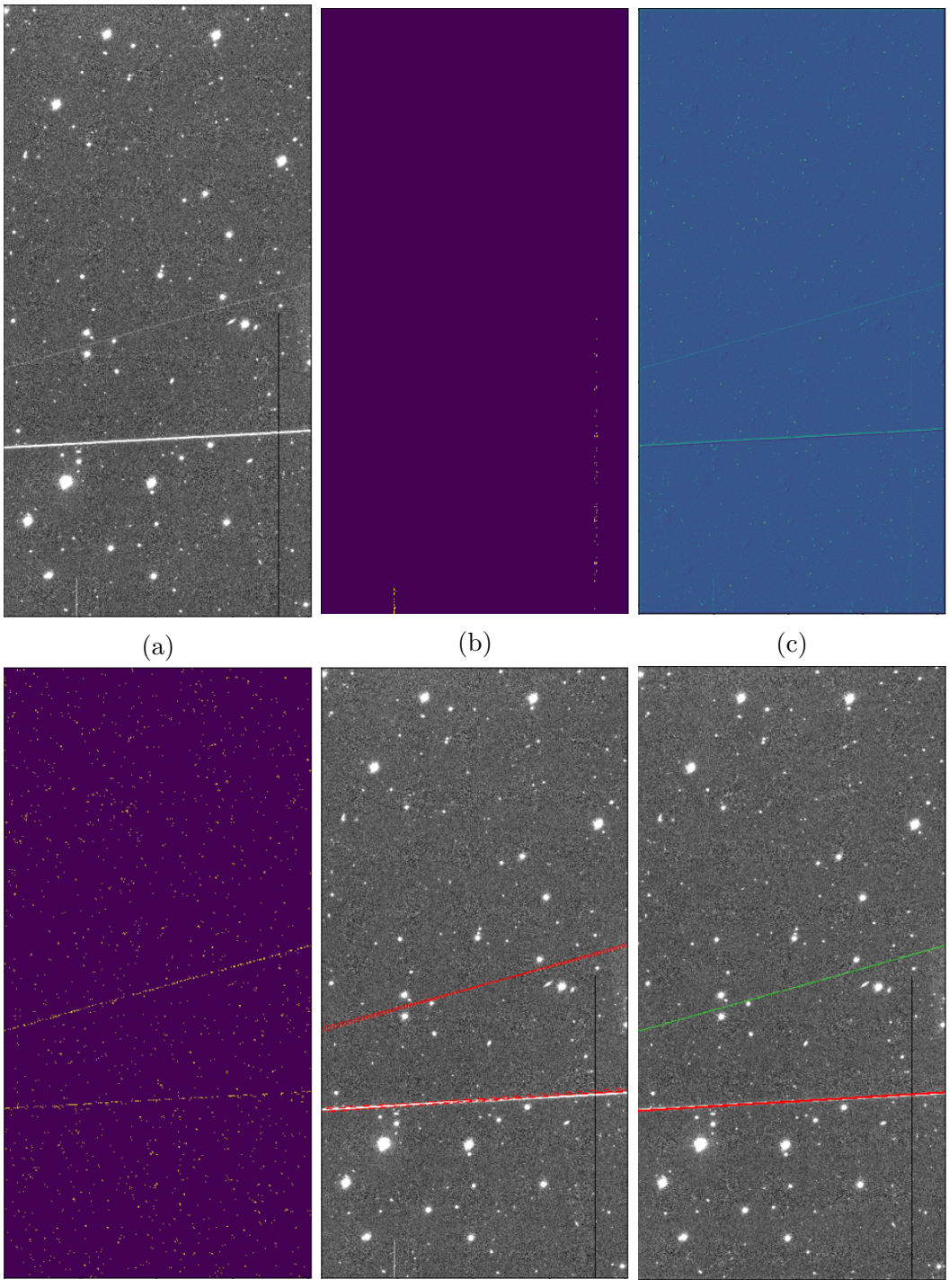


Figure 3.15: The entire process summary. (a) Input bloc. (b) Isolating pixels from transient effects. (c) top-hat transform on results of gabor filters. (d) Binary image after Canny filtering. (e) Lines detected in Hough transform. (f) Two satellites detected in the image

3.2 Final Result

Processing a full image takes approximately 9 minutes using 6 workers in a parallelized algorithm using a multiprocessor Intel Core i7 with 6 cores and 12 threads.

The results show that the process is able to detect long satellite tracks in most cases when these streaks are visible. The parameters allowing to ignore star spikes that can be interpreted as streaks are, in some cases, too strict and sometimes generate false negatives. On the other hand, no cases of false positives were observed among the 12 images studied.

Moreover, the cases of false negatives are isolated in blocks. In other words, the satellites that generated cases of false negatives in a block were detected in contiguous blocks of the same image as presented in figure 3.16. Thus no satellite present in these images and visible to the naked eye was forgotten by the process. On the other hand, the cases of false negatives allow us to conclude that particular cases of tracks passing in corners of the image or appearing only in one block in the whole image can lead the process to forget a satellite in the image. The first work to be done to reduce the number of false negatives without creating false positives is first to determine a method to exclude diffraction spikes.

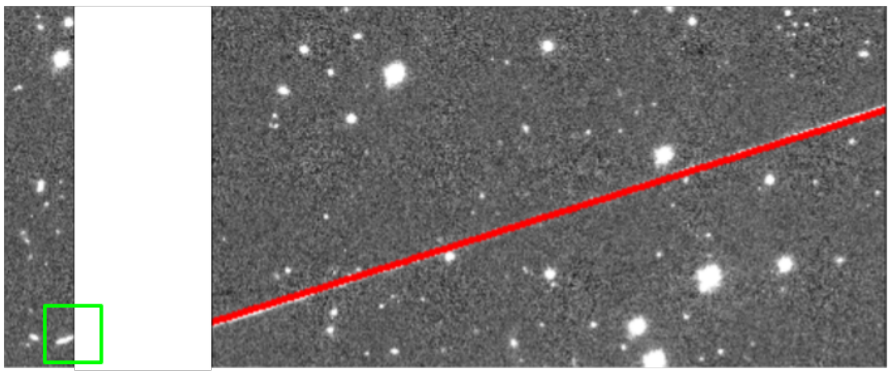


Figure 3.16: Example false negative inside the green box, next to the block in which the satellite is detected

3.2.1 Predictsat

The results of this process are compared with an algorithm, called Predictsat, predicting the trajectories of the satellites according to a predefined catalog. First the tracks detected by the method of this project are consistent with an official catalog referencing the known satellites. So it proves that we can recover satellite streaks with this method.

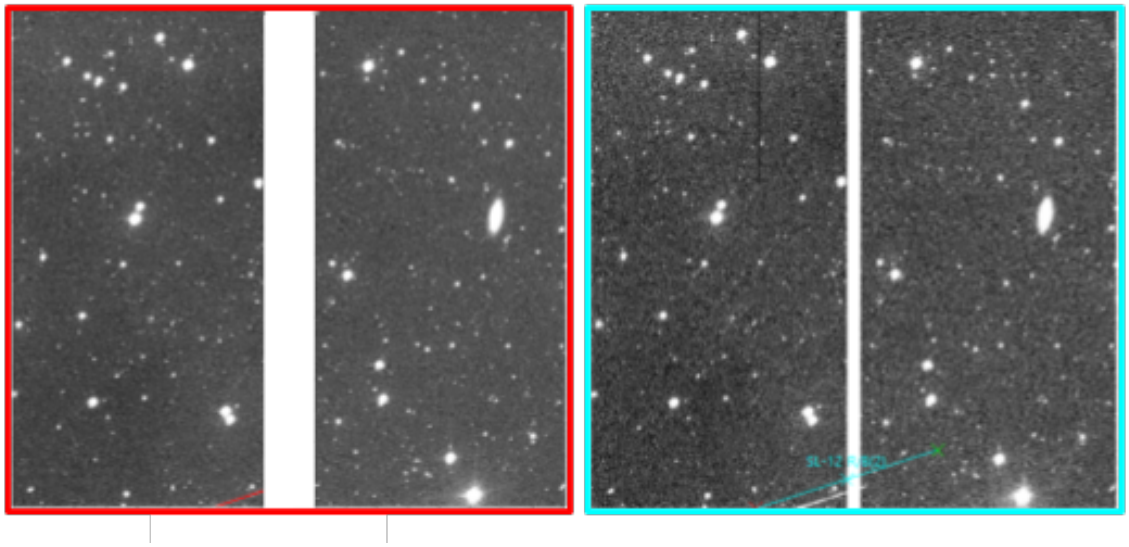


Figure 3.17: On the right, result of image processing method; on the left, prediction of Predictsat. The debris SL-12 R/B(2) is correctly detected.

However we can also conclude from some samples that non visible satellite won't be detected by this process.

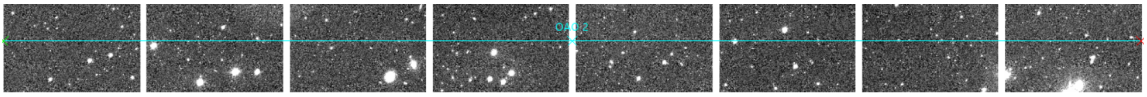


Figure 3.18: OAO-2 satellite streak predicted by Predictsat but undetected with image processing

It is also observed that a streak is detected by image processing but not predicted by Predictsat.

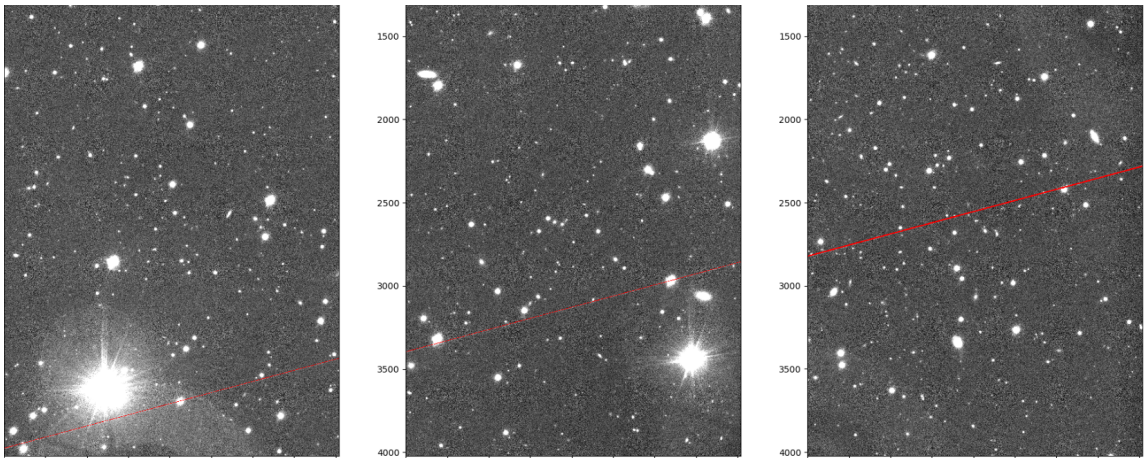


Figure 3.19: Example of unpredicted but detected streak

3.2.2 Further Corrections in the algorithm

The current algorithm has some shortcomings in determining the width of satellite streaks when they are too faint. Transforms used on the image before analyzing the intensity along axes may not be appropriate for these cases.

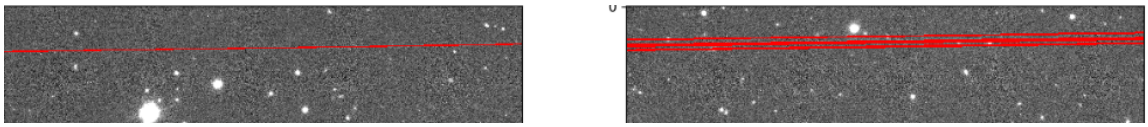


Figure 3.20: On the left, the computed width of the streak is consistent with the observations while the algorithm wrongly draws three thick lines on the right.

We can see three parallel bands on the satellite detected on the right in figure 3.20. It has to be precised that only one satellite is detected. However, by looking at the mean intensities along this streak, no clear peak of intensity is retrieved from the analysis. This by recovering the full width at half maximum (in figure 3.21) at the orientation where we found the maximal intensity, the algorithm will return three bands as the algorithm does not handle the case where many bands are retrieved.

Once these method is corrected, it could lead further works implying the analysis of the pixels inside the pixels band then retrieved. Thus it can give information about the satellite such as its speed, its angular velocity, etc.

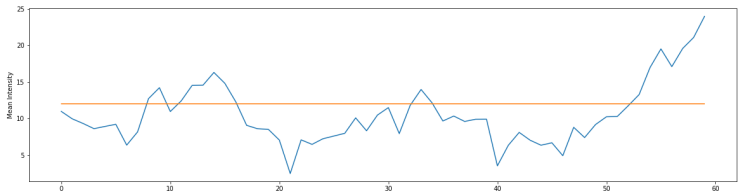


Figure 3.21: Mean intensity distribution where the peak of intensity is observed for the faint streak. We can observe that four parts of the curve is above the half maximum leading to the bands previously observed in the image

Chapter 4

Low Velocity Objects

4.1 Motivation

We have seen that Hough method has its limits because of the spikes of bright stars and the noise remaining in the image. Therefore, the low velocity satellite tracks are much smaller than the tracks we detect with the previous method and these streaks are undetectable with the previous method. The idea to overcome the limitations of the Hough method is to analyze the performance of a neural network to find these tracks. In previous works [1], the difference images and SExtractor have been studied in order to recover cutouts of objects that have a possibility of being a satellite. SExtractor is a tool that computes a catalogue of objects in astronomical images. These cutouts were used to train a model for the classification of cutouts. The goal was to detect if each cutout contained a satellite or not. This work showed very good performance (96.7 % accuracy) for classifying these cutouts. This method allowed the classification of long and short tracks. The study of a new method is therefore proposed within the framework of single epochs. The classifiers having already been studied, the idea is thus to propose a method of image segmentation. The number of elements in the single epoch image is much more important than in a difference image (starts, galaxies, and other static light sources). In order to isolate the small satellite tracks, it would therefore be interesting to isolate them from the rest of the image thanks to this segmentation method which will consist in classifying each pixel of the image as belonging (class 1) or not belonging (class 0) to a satellite

track. To do this, a dataset, which does not exist to the knowledge of this project, must be created.

The particularity of this exercise is the absence of a real sample of low velocity satellites presented in this project. Indeed the absence of a predefined dataset for this task, the lack of expertise and time have slowed down the creation of such a dataset for single epoch images. Moreover, this brought too little information about the actual samples: the general intensity distribution of the pixels belonging to a track, the dimensions and the size of such objects on the image. Therefore, the objective of this part is to see if the prior knowledge of the geometry of a single object allows to train a neural network on a synthetic dataset in order to be efficient on real samples or not.

The dataset then created for the Difference Image Classifier using SExtractor [1] was composed of about 50,000 32x32 cutouts where only 7.8 % contained satellites. The complexity of creating a dataset manually on single epoch images is much more complex since the streaks present on the image can be artificial as well as natural objects. The elements of this project do not offer sufficient expertise to be able to classify such objects in a reasonable amount of time. It was therefore decided to take advantage of a prior knowledge of the geometrical structure of these streaks to create a dataset of synthetic tracks, allowing both to propose a dataset of larger data and to control the ratio of satellites in the images to optimize the learning which aims to train a model that will be effective on non-synthetic samples.

4.2 Synthetic Image Generation

[H] The creation of the synthetic samples is based on the visual analysis of samples presented in a report that presented samples found in different images. It was therefore decided to draw lines whose size, thickness, intensity and orientation vary on real VST images.

This synthetic dataset is then based on an existing single epoch image taken by the

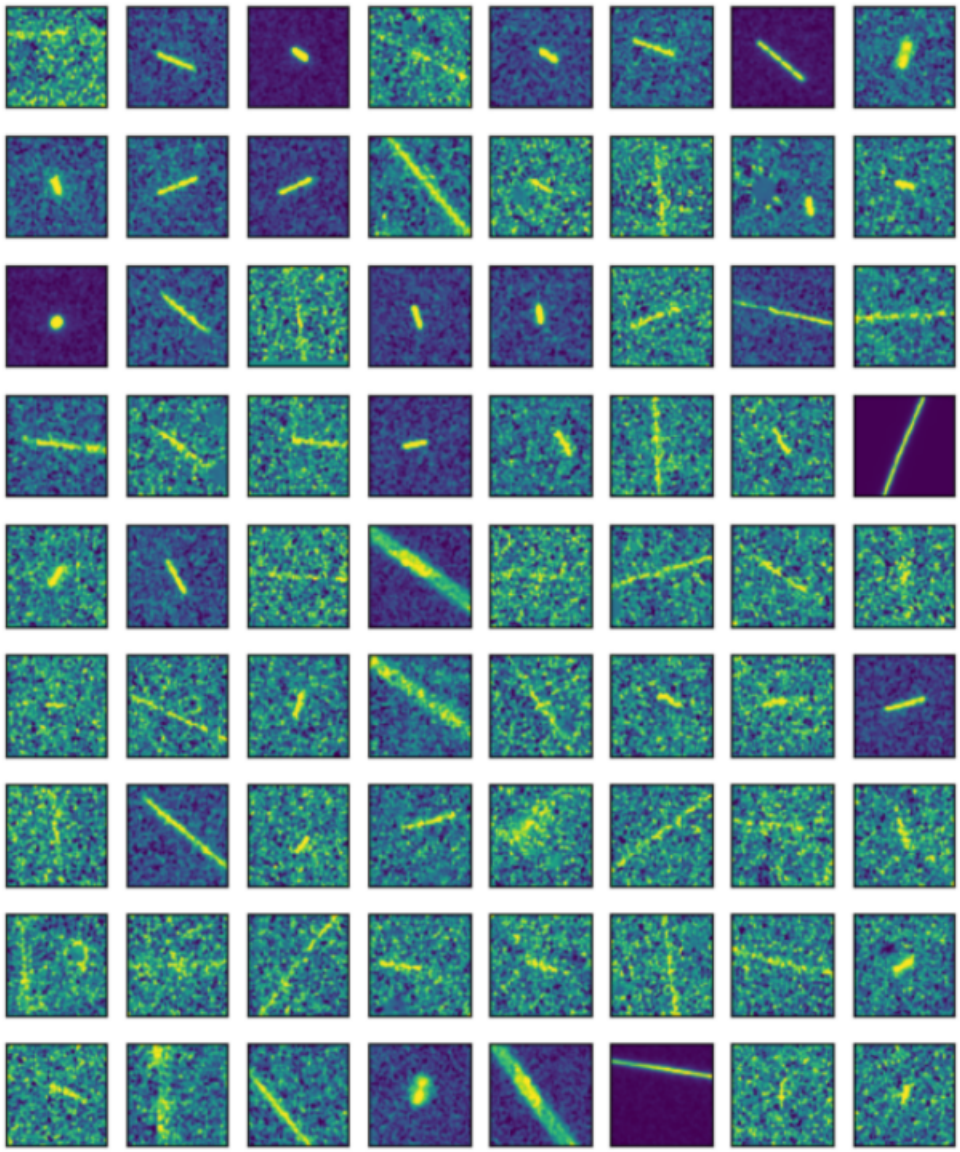


Figure 4.1: Real streaks retrieved in the project with difference imaging [1]

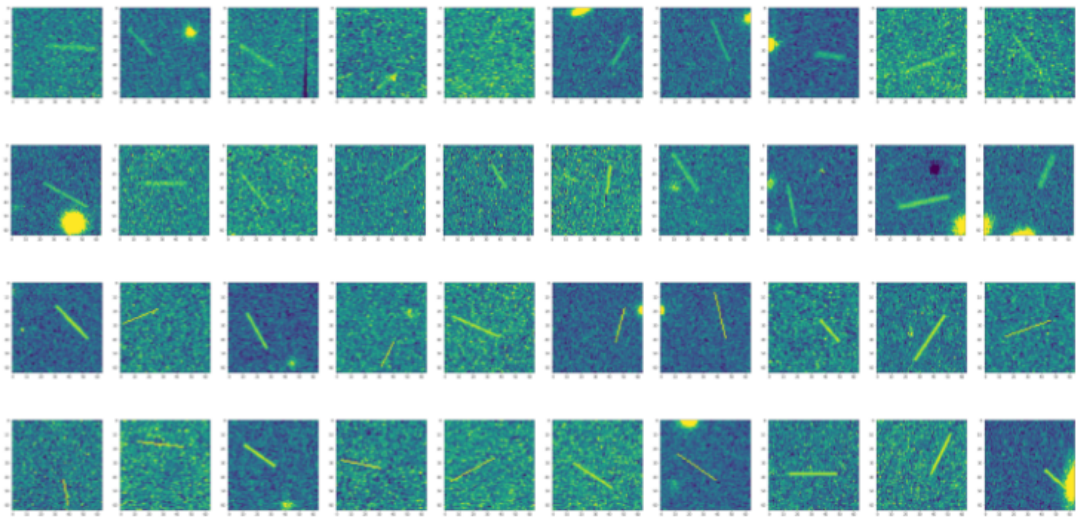


Figure 4.2: Samples of synthetic satellite streaks generated for training and testing the neural network

VST telescope, which does not a priori have any long visible satellite track. By taking such an image, it is necessary to cut it into several sub-blocks of 64x64 pixels. Random vertical and horizontal flips are applied to the 64x64 pixels block so that the same sections of the image can be reused without creating recognition patterns that the neural network can use to overfit a training set. In order to opt for a ratio of about 50 % of satellite images, a choice is made following a Bernouli's law, according to which a satellite is entered with a probability of $p = 0.5$ in the sub-block that has just been created. The length of the synthetic tracks varies between 20 and 40 pixels. The thickness is 1 to 6 pixels. The orientation varies between 0 and 179 degrees with respect to the horizontal axis. A gaussian blur is applied on the generated track then its transparency is defined between 0.4 and 1. before overlaying it on the original block.

4.3 UNet for Image Segmentation

4.3.1 Model

The UNet architecture is presented as a model for image segmentation in the biomedical field. It is also used for road segmentation in satellite images. This convolutional

neural network model is design to localise elements in the image and label each pixel of the images according to predefined classes. The model consists of a contracting path and a symmetric expanding path, both separated by a bottleneck layer. The contracting path consists of a sequence of convolutions and max-pooling to produce high-resolution feature maps. Then, the expanding path provides a series of convolutions and transposed convolutions. The interesting results this architecture offered for the road segmentation problem [4] inspired this project thanks to some geometrical similarities one can observe between a road and a satellite streak.

Some modification has been made to the original model for the convenience of this project. First the complexity of the architecture has been decreased by reducing the number of channels at each convolutional layer. Also the original model propose 2 successive convolution layers before each max-pooling (or transposed convolution). For this project we propose to remove one of this two convolution layers every time. The model became then less time-consuming to be trained and also would prevent from overfitting a training dataset. Then, the number of channels of the output of the model is reduced to a single channel, as we only need one probability per pixel in order to apply the classification.

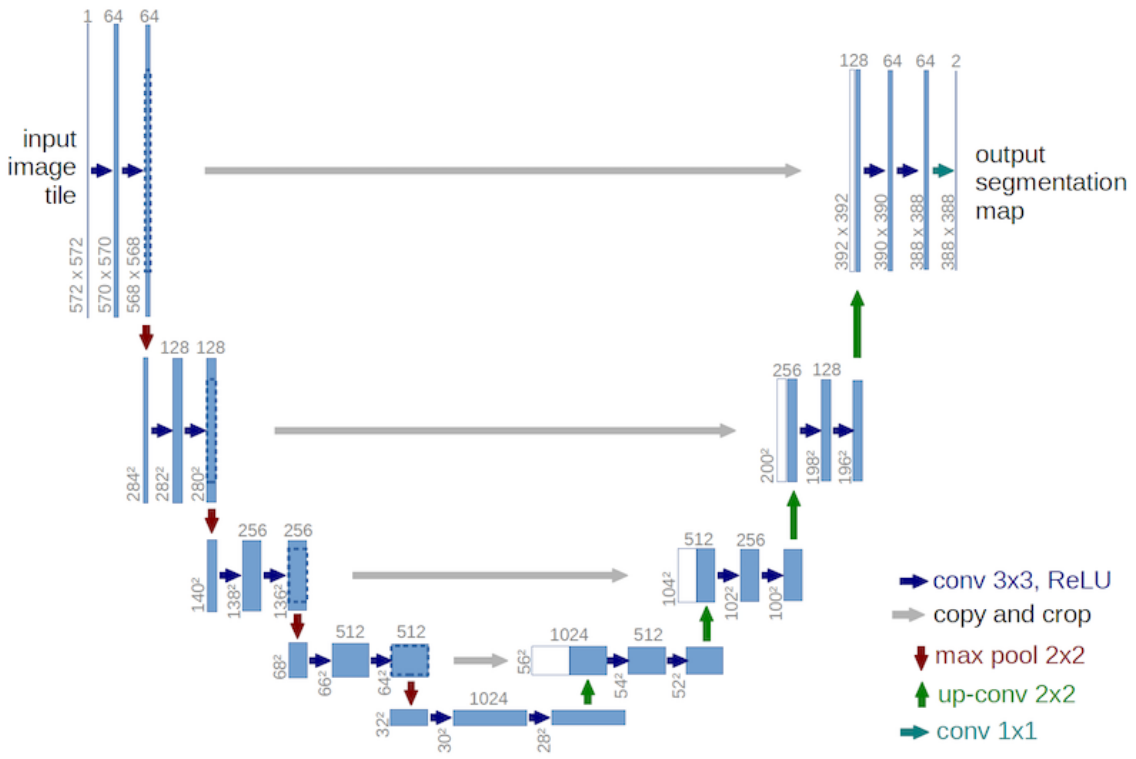


Figure 4.3: Original UNet Architecture [5]

4.3.2 Training

Time and memory limitations motivated the use of a dataset composed of 38397 samples including 19250 synthetic satellites (3.7 GB of data). 20 % of the samples (7680) were used to validate the model, and thus 30717 samples were used to train the model.

The aim of our model is to predict if a pixel belongs to a satellite streak (1) or not (0). As a consequence the binary cross entropy loss has been chosen as the loss function for the training of the UNet architecture. Considering $x \in \mathbb{R}^n$ as the input of our model and $y \in \{0,1\}^n$ the classes of the input, then the binary loss is defined as :

$$l(x,y) = mean(L)$$

$$L = [l_1, ..., l_N]^T$$

where $l_n(x,y) = -[y_n.log(x_n) + (1-y_n).log(1-x_n)]$ As a consequence the model will

apply a sigmoïd function to each element of the output before returning it. Therefore each of these element can be considered as the probability that a pixel belongs to a streak or not.

The optimizer is the adaptive moment estimation (Adam) [6] optimizer which is a variation of the stochastic gradient descent that computes adaptive learning rates for different parameters based on the estimations of first and second moments of the gradient in order to accelerate the convergence.

The metric that has been used as a reference to the performance of the model on training and validation sets is the Jaccard distance. This metric quantifies the dissimilarities between two sample sets. If we define the Jaccard index of two sample sets A and B as :

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

then the Jaccard distance is defined as :

$$d_J(A, B) = 1 - J(A, B)$$

Dropout

Many over-fitting phenomena were observed as the mean final loss on the train set was significantly smaller than the mean final loss over the validation set. To reduce such a behavior, dropout [7] was inserted into the different layers of the network. We tried to add dropout layers with $p = 0.1$ the probability of keeping the weight. Those dropout layers are append in this way for the contracting path :

$$Convolution\ Block \longrightarrow Max\ Pooling \longrightarrow Dropout$$

and in this way for the expanding path :

$Transposed\ Convolution \longrightarrow Concatenation \longrightarrow Dropout \longrightarrow Convolution\ Block$

Batch Normalization

In order to accelerate the convergence and increase the robustness of the model, it was decided to analyze the performance of a model with batch normalization [8] which consists in shifting and rescaling the inputs to internal layers according to the mean and variance estimated on the batch.

4.4 Final Results

After 16 epochs, the training reached an asymptotic loss, leading to an early stop of the training process. One epoch takes approximately 20 minutes on a CPU.

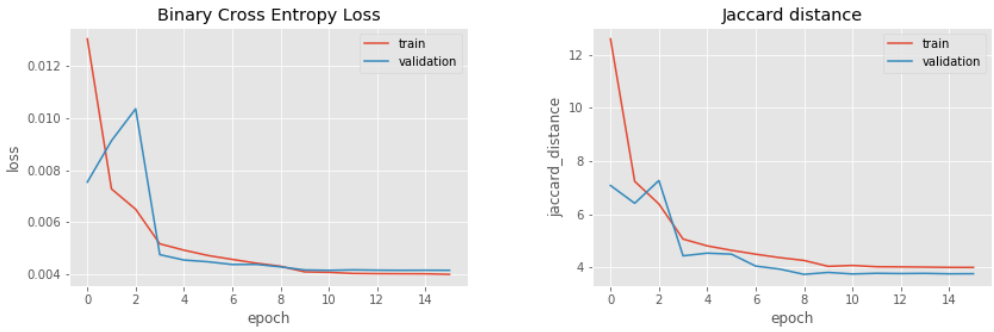


Figure 4.4: Training History

4.4.1 Patch-wise Performance

After training, the model detects a satellite on a 64x64 patch from the test set with an accuracy of 98.8 % and a f1-score of 98.7 %.

true positive	true negative	false positive	false negative
3757	3830	20	73

Table 4.1: Patch-wise classification : whether or not our segmentation method correctly detect if a satellite is in a 64x64 patch or not.

4.4.2 Pixel-wise Performance

After training, the model correctly classifies each pixel of the test set with a f1 score of 95.1 %.

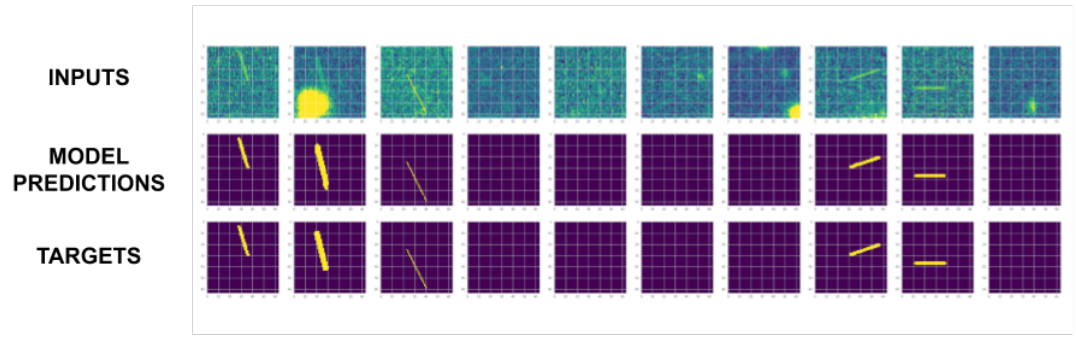
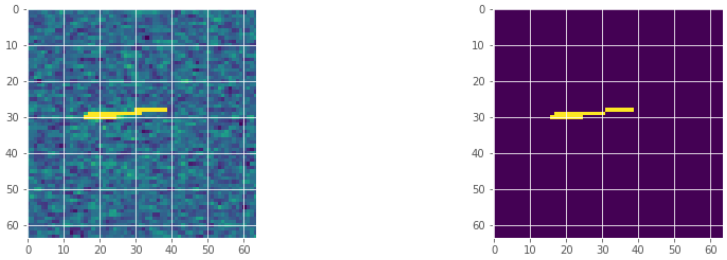


Figure 4.5: Sampled results of the UNet over a synthetic test set

Considering a real satellite streak sample retrieved from an image taken by the VST, the aim is to appreciate the potential of our model on real data with a training on synthetic data.

First, we cut out a 64x64 patch centered on the satellite so that the model returns a qualitatively satisfactory result as presented in Figure 4.6 which shows that our model is sensitive to real streaks.

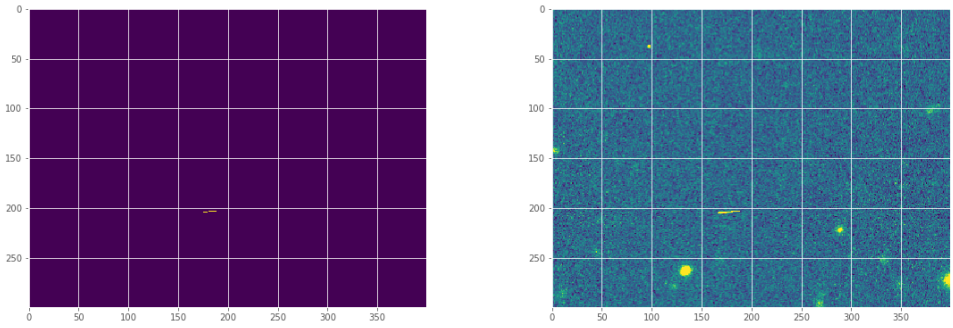


(a) (b)

Figure 4.6: (b) Prediction of the model on a (a) real sample.

If we apply the model on a full block we observe after 20 seconds of computation that the returned segmented image does not contain any false positive. Also the

prediction is less precise than on the centered patch since the satellite streak can be shared by two patches.



(a) (b)

Figure 4.7: (b) Prediction of the model on a (a) real sample by sequential process on a full block.

Furthermore it is interesting to observe the performance of the model on a block containing a long track in Figure 4.8 which is satisfying enough to think that the model can be applied on any kind of streaks.

However it has to be recalled that in order to propose this result over a long track, the model has to be trained over the same portion of the sky during almost 6 hours while the Hough method does not need to know a priori what portion of the sky the telescope would observe.

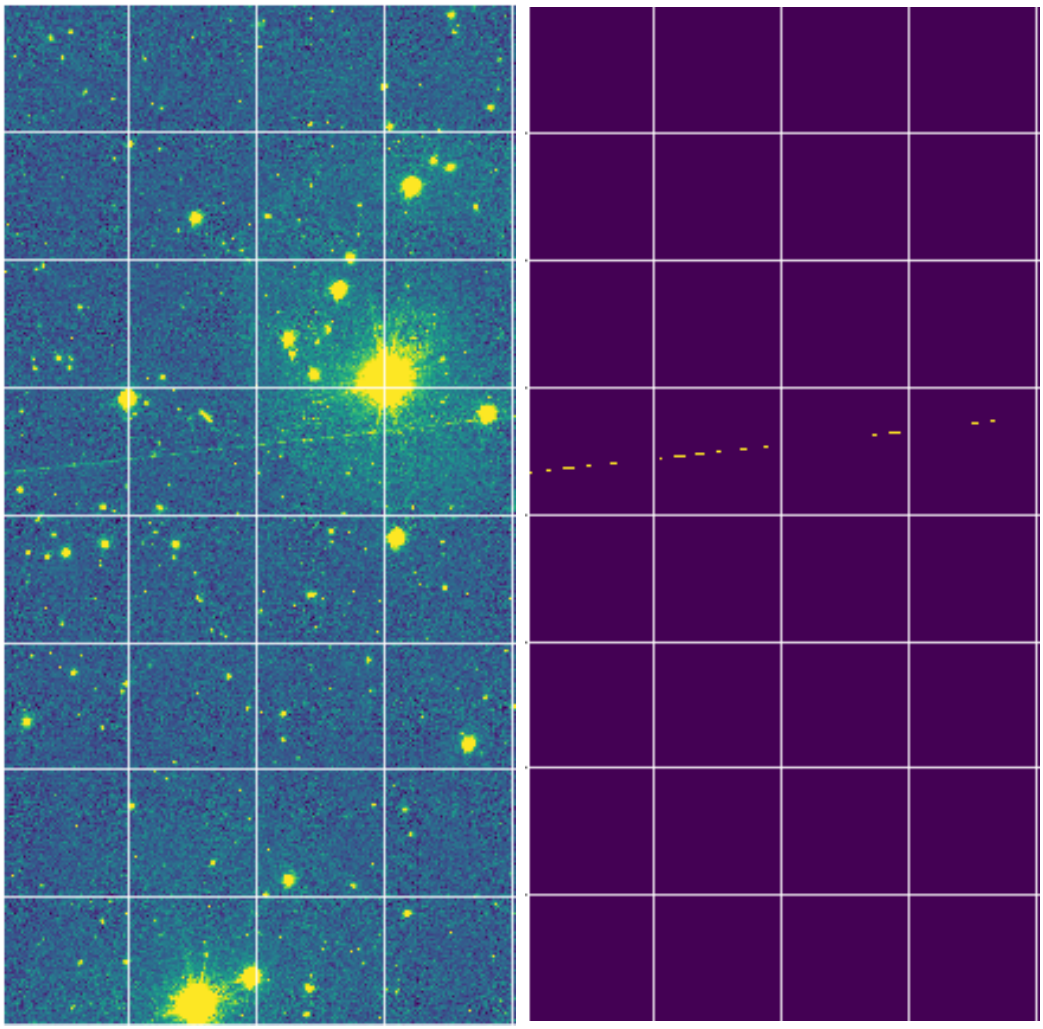
4.4.3 Future Work

This helps to see that the UNet is capable of segmenting an image with real satellite streaks after a training on a synthetic dataset. Thus knowing only the approximate geometry of the object that we want to detect in the image leads to encouraging results. This motivates to further work on this model such as:

- Increasing the synthetic dataset;

- Deporting the model and its training on a GPU. This would give the possibility to complexify the model if requested in the future;
- Improving the creation method of a synthetic dataset. Creating a primary dataset with real samples would lead to study of a Generative Adversarial Networks which could potentially generate all kind of satellite streaks and backgrounds. Thus a model trained over a sufficient generated training set would be robust enough to perform over any portion of the sky.

Going in this direction will therefore make it possible to create datasets that are sufficiently structured to also allow the performance of new models to be quantified.



(a) (b)

Figure 4.8: (b) Prediction of the model on a (a) real long track by predicting sequentially each 64x64 patch of a 4000x2000 block.

Chapter 5

Conclusion

In response to previous work using the difference imaging technique to detect satellites in astronomical images, this project aims to prove that satellite streaks can be detected in single epoch images in a systematic way. For this purpose we have distinguished two types of moving objects leaving streaks in the astronomical image. The high velocity objects leave long tracks of variable intensity on the image and the low velocity objects leave streaks of a few tens of pixels long on the image. Our first method focuses on the detection of high velocity objects. To do so, the method proposes different processes. A first process removes the transient effects of the image that does not concern the objects of interest by applying morphological transforms. Then it is a question of highlighting linear objects thanks to filters and morphological transforms to then detect straight lines in Hough space. This allows to detect satellite tracks independently of their intensity as long as they are visible to the naked eye. The results were compared with software that predicts the trajectories of debris and satellites in an official catalog to prove the consistency of our results with our initial problem. Despite some additional corrections to be made in the current algorithm, the results allow to consider new research perspectives such as the development of an algorithmic method to compare the detections of our method with the predictions of Predictsat to provide information on the detected streaks (name of the debris or satellite) and statistics on the performance of detections and predictions. It will also be a question of developing algorithms for the analysis of the detected tracks to recover additional insights about the satellites such as their

speed and angular velocity. Finally, a comparison with other existing methods [9, 10, 11] to detect satellite tracks would be interesting to quantify the performances of our method compared to those already existing.

The observation of the limitations of this image processing method regarding the detection of tracks coming from low velocity objects leads us to propose a machine learning method to overcome these limitations. The training of a simplified version of the UNet for image segmentation has shown that prior knowledge of the geometry of the object to be detected in the image can be sufficient to detect with a certain accuracy real satellite tracks. This encourages the further study of this model by building a dataset of real samples of satellite tracks as well as the development of a GAN whose aim will be to diversify the backgrounds and the potential sources of false positives (high intensity light sources, diffraction egrets) in order to produce a more robust model that can be adapted to several fields of the sky. Also deporting the model and its training in a GPU will accelerate the learning process and also provide a more complex model closer to the original UNet if needed.

Bibliography

[1] Su Direkci. 'Studies on Satellite Detection with Difference Imaging'. en. In: (2020).

[2] Edward R. Dougherty and Roberto A. Lotufo. *Hands-on Morphological Image Processing*. en. SPIE, July 2003. ISBN: 978-0-8194-7866-5. DOI: 10.1117/3.501104. URL: https://spiedigitallibrary.org/ebooks/TT/Hands-on-Morphological-Image-Processing/eISBN-9780819478665/10.1117/3.501104 (visited on 09/01/2021).

[3] Yitian Zhao et al. 'Automatic 2-D/3-D Vessel Enhancement in Multiple Modality Images Using a Weighted Symmetry Filter'. en. In: *IEEE Transactions on Medical Imaging* 37.2 (Feb. 2018), pp. 438–450. ISSN: 0278-0062, 1558-254X. DOI: 10.1109/TMI.2017.2756073. URL: http://ieeexplore.ieee.org/document/8049478/ (visited on 09/01/2021).

[4] Arkadiusz Nowaczynski. 'Deep learning for satellite imagery via image segmentation'. In: (2017). URL: https://blog.deepsense.ai/deep-learning-for-satellite-imagery-via-image-segmentation/.

[5] Philipp Fischer Olaf Ronneberger and Thomas Brox. 'U-Net: Convolutional Networks for Biomedical Image Segmentation'. In: (2015).

[6] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[7] Nitish Srivastava et al. 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting'. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[8] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[9] M P Lévesque and S Buteau. 'Image Processing Technique for Automatic Detection of Satellite Streaks'. en. In: ().

[10] Guy Nir, Barak Zackay and Eran O. Ofek. 'Optimal and Efficient Streak Detection in Astronomical Images'. en. In: *The Astronomical Journal* 156.5 (Oct. 2018). arXiv: 1806.04204, p. 229. ISSN: 1538-3881. DOI: 10.3847/1538-3881/aaddff. URL: http://arxiv.org/abs/1806.04204 (visited on 15/01/2021).

[11] Dino Bektešević et al. 'Linear feature detection algorithm for astronomical surveys – II. Defocusing effects on meteor tracks'. In: *Monthly Notices of the Royal Astronomical Society* 474.4 (Dec. 2017), pp. 4837–4854. ISSN: 1365-2966. DOI: 10.1093/mnras/stx3085. URL: http://dx.doi.org/10.1093/mnras/stx3085.

Appendix A

Summary of the UNet Model adapted for this project

Layer (type)	Channels #	Kernel size	Number of Weights
Conv 2D	32	3x3	320
Batch Norm 2D			128
ReLu			
MaxPooling 2D		2x2	
Dropout			
Conv 2D	64	3x3	18496
Batch Norm 2D			256
ReLu			
MaxPooling 2D		2x2	
Dropout			
Conv 2D	128	3x3	73856
Batch Norm 2D			512
ReLu			
MaxPooling 2D		2x2	
Dropout			
Conv 2D	256	3x3	295168
Batch Norm 2D			1024
ReLu			
MaxPooling 2D		2x2	
Dropout			
Conv 2D	512	3x3	1180160
Batch Norm 2D			2048
ReLu			

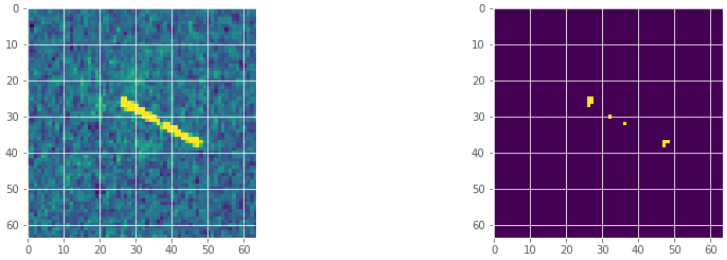
Figure A.1: Modified version of the UNet Architecture for the project with a total of 4,705,377 trainable parameters (Contracting Path and Bottleneck

Layer (type)	Channels #	Kernel size	Number of Weights
TransConv 2D	256	3x3	1179904
Dropout			
Conv 2D	256	3x3	1179904
Batch Norm 2D			1024
ReLu			
TransConv 2D	128	3x3	295040
Dropout			
Conv 2D	128	3x3	295040
Batch Norm 2D			512
ReLu			
TransConv 2D	64	3x3	73792
Dropout			
Conv 2D	64	3x3	73792
Batch Norm 2D			256
ReLu			
TransConv 2D	32	3x3	18464
Dropout			
Conv 2D	32	3x3	18464
Batch Norm 2D			128
ReLu			
Conv 2D	1	3x3	33

Figure A.2: Modified version of the UNet Architecture for the project with a total of 4,705,377 trainable parameters (Expanding Path and Output

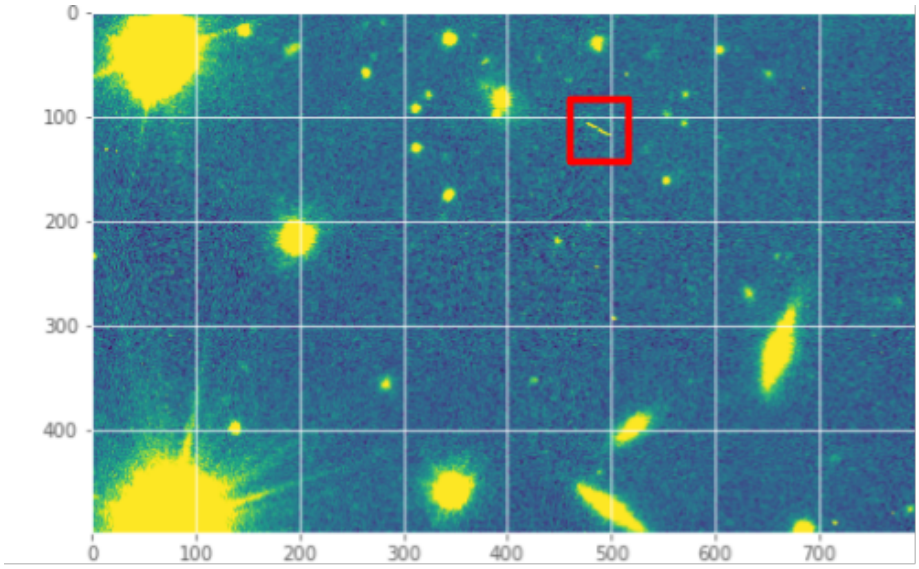
Appendix B

Other real satellite streak sample for testing the UNet model

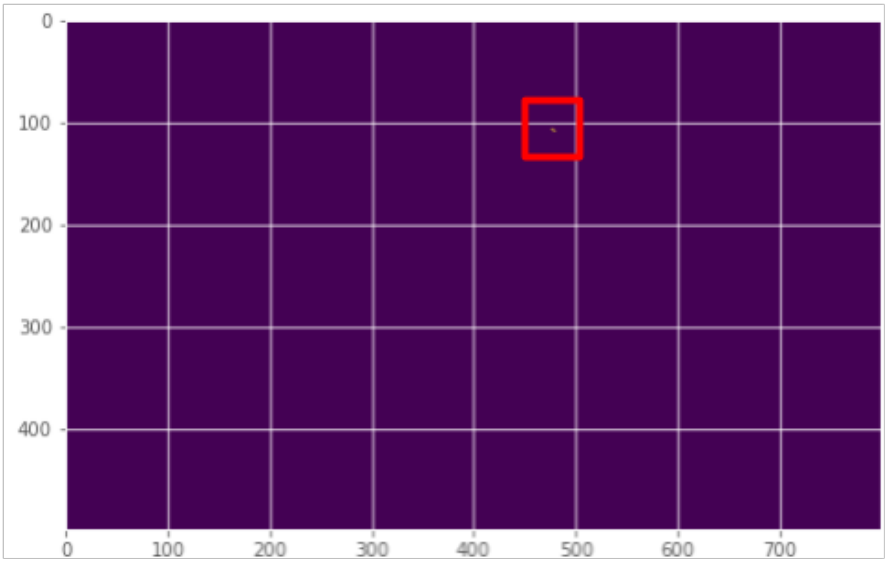


(a) 64x64 input patch (b) Model prediction

Figure B.1: (b) Prediction of the model on a (a) real sample.



(a)



(b)

Figure B.2: (b) Prediction of the model on a (a) real sample by sequential process on a full block.