EPFL

SSA ASSOCIATION - COMPUTER VISION LABORATORY

Master of Science in Computational Science and Engineering

Detection of satellite tracks in astronomical images

Supervisor:

Prof: Mathieu Salzmann

Student:

Theo Patron

ACADEMIC YEAR 2022

Abstract

The increasing number of satellites orbiting the earth strengthen the need of a catalog containing relevant information about each object. In order to create such a catalogue, satellite streaks have to be detected in astronomical images.
The goal of this project is to develop a reliable method that allows to detect the streaks on the images taken by the SSA's telescope. This has to be done under the constraint of a low data regime.
Two approaches will be tested. The first one is based on the previous work from [1] and [2] and uses a U-Net based network to detect the streaks. Synthetic data and an iterative training will be used in order to train the network in an unsupervised way. The second approach is based on the work of [5] and [4] and uses a novel architecture that is called HT-LCNN. Transfer learning will be used here since we will start from the model pretrained on the ShanghaiTech dataset [3].
Even if some good results (0.7 mean IoU) are achieved with the first method, a pixel-wise data annotation is needed to assess the model performance. Furthermore the robustness of the network to unseen data cannot be assured since only a dozen real streaks were available at this time. The second method yields very good results even with the model trained on the ShanghaiTech dataset (0.844 sAP). Fine-tuning on our dataset allowed to improve the performance (0.988 sAP). This method is prefered since the annotation is easier to do and the training is easily extendable to new data.

Acknowledgments

I would like to adress a special thanks to Prof. Mathieu Salzmann for his guidance throughout the semester. I also thank Yann Bouquet and Alexandre Di Piazza for their previous work on this subject, Luca Hartmann for his availability during the semester, and Elisabeth Rachith for the advices and the help.

Contents

Chapter 1

Introduction

The rapid growth of the number of satellites orbiting the earth make it useful to have a catalog of objects orbiting the Earth in order to prevent collisions with existing satellites and ease space development. This is precisely the goal of the Space Situational Awareness (SSA) association. To do so, it is necessary to detect, classify and identify orbiting objects. This report focus on the detection part. To this end, two different methods are used to detect the satellite streaks. The first method is based on the work of Yann Bouquet and Alexandre Di Piazza and uses the network U-Net, a well-known network in image-segmentation. The lack of data will make us try a second method, based on the network HT-LCNN [4], which uses prior knowledge to improve data efficiency.

Chapter 2

Input data

Unlike the previous semester projects done for the SSA association, the data used is not taken from the OMEGA-CAM on the VLT Survey Telescope. The SSA association now owns its own telescope and the data collected by their observations will be used.
The telescope provides 6248×4176 grey-scale images and the exposure time varies between 0.5s and 30s. An example is shown on Figure 2.1.

Figure 2.1: Example image containing a streak taken by SSA's telescope

The iraf's ZScale algorithm is used to rescale the pixel intensities and is described in [1].

Chapter 3

Method 1 - U-Net approach

3.1 Generation of synthetic data

At the moment this approach was tested, only a dozen of streaks were available. As it is way too few to train a neural network, some synthetic data had to be created. The creation of the synthetic samples is based on the visual analysis of the available streaks. Lines whose size, thickness, intensity and orientation vary are drawn on real images taken by the telescope that did not contain any streak.
The length varies between 1000 and 2000 pixels, the thickness between 10 and 50 pixels and the intensity is set to 255.
An erosion is then performed on some random parts of the line in order to make it less uniform. The intensity is also decreased on some part of the line with the same goal. A gaussian blur is applied on the generated track and the line is then overlayed on the original image. An example is shown in Figure 3.1.

Figure 3.1: Example of a synthetic streak.

As the GPU memory cannot handle images as large as 6248×4176, each image is then split into several patches of size 768×688.

3.2 U-Net model

The U-Net is widely used in many image segmentation domains (road segmentation, biomedical field, ...). The original architecture is shown in Figure 3.2.
This convolutional neural network model is designed to localise elements in the image and label each pixel of the images according to predefined classes. Some modifications have been made to the original model for the convenience of this project. The model became then less time-consuming to be trained and also would prevent from overfitting the training dataset. The modification are as explained in [1].

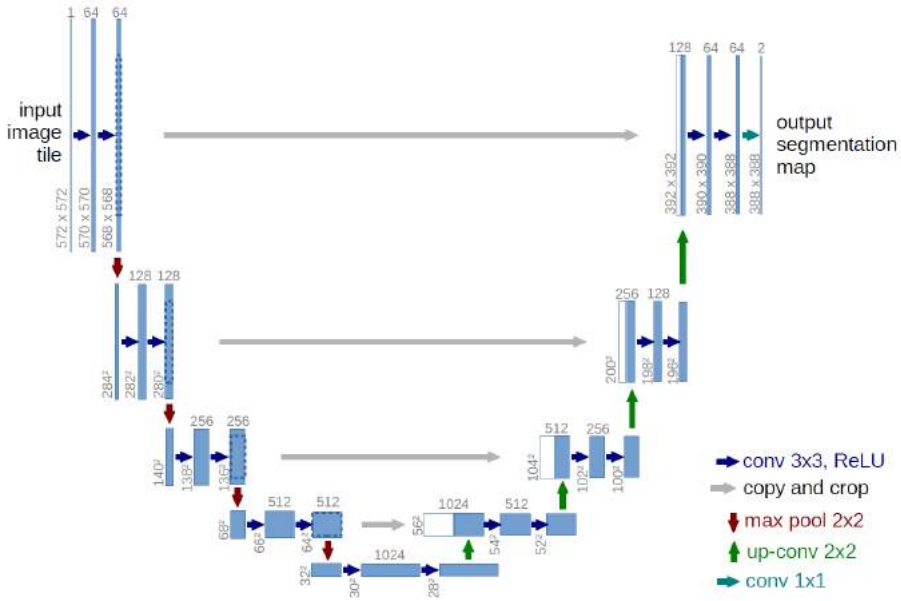
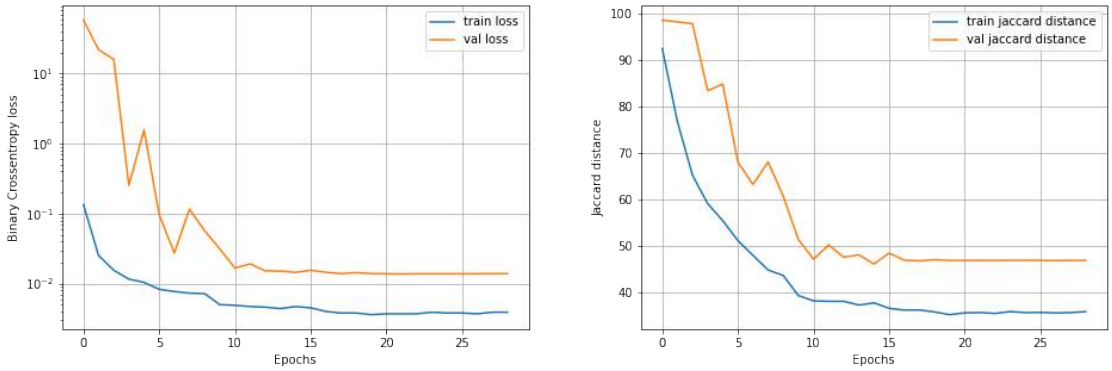


Figure 3.2: Original U-Net architecture.

As in [1], the binary cross entropy loss and the Adam optimizer are used. The metric used to measure the performance of the model is the Jaccard distance.

3.3 Training

Using all available images available at this time, the dataset is composed of 613 training patches and 327 validation patches (each patches being 768×688).
The model is trained for 65 epochs with a batchsize of 20 on the EPFL cluster.
After 16 epochs, the training reached an asymptotic loss, leading to an early stop of the training process.



(a) Loss (b) Jaccard distance

Figure 3.3: Training history.

3.4 Results

In order to evaluate the performance of our model, the mean IoU (intersection of union) is computed for two datasets. The IoU is computed for each patch and then averaged across all the patches.
First, using the validation set of the dataset containing the synthetic streaks (the one used to train the model), the mean IoU is 0.71.
Second, using the set of the real streaks, the mean IoU is 0.31. As can be seen on Figure 3.4, the real streaks are often not detected by the model at all.
Then, the model does not generalize well to real data. We decided to try to fine-tune the model using some real streaks.

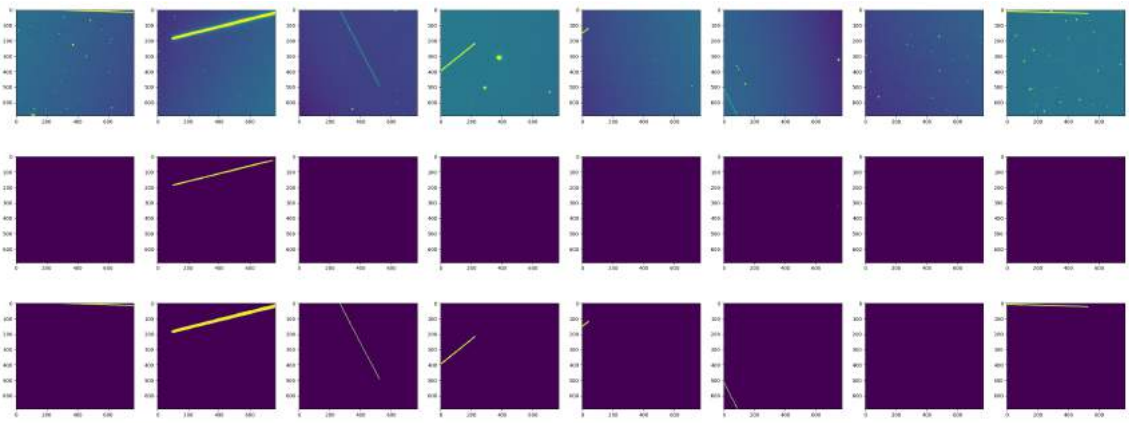


Figure 3.4: Sample results of the trained U-Net on some real streaks. The original patches are on top, the ground truths are on the bottom, and the predictions are in between.

3.5 Fine-tuning

In order to fine-tune the model on some real examples, the available streaks were annotated. As some streaks were present on multiple images (same streak but not at the same place on the image), it was decided to use only one image per streak, to avoid any overfitting that would invalidate the results. The streaks are then separated into a training and a validation set. The network is then trained with this new dataset, starting from the weights obtained with the training on the synthetic examples. The mean IoU on the validation set is 0.64. This shows that the model is now more accurate on real data. However the extremely low number of available streaks (22 patches in the validation set and 30 in the training set) make the results uncertain, and probably not robust to new data. Furthermore, the pixel-wise annotation is long and tedious to do, so it does not seem like a good solution, even with if more examples of real streaks are available in the future. We decided to try an unsupervised approach.

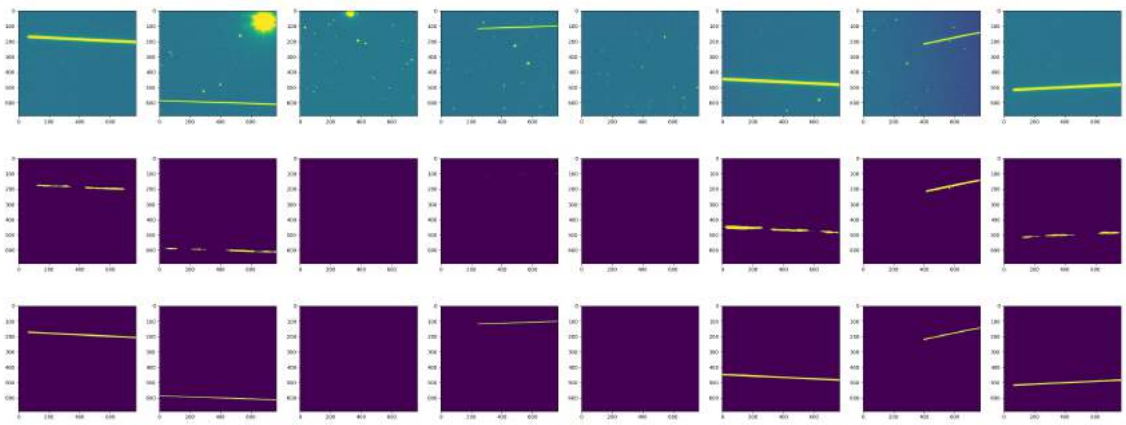


Figure 3.5: Sample results of the fine tuned U-Net on some real streaks. The original patches are on top, the ground truths are on the bottom, and the predictions are in between.

3.6 Unsupervised training

3.6.1 Iterative training

As can be seen on Figure 3.4, the model makes no (or very few) false-positive (this was confirmed by looking at all the examples extensively). This gave the idea of an iterative training as fine tuning.
The idea is to use the prediction as the ground truth and to retrain the model for a few epochs with this ground truth. Then, the new prediction is taken as the new ground truth and the model is trained again for a few epochs. This is done iteratively. To do such a training, no annotations are needed, but a custom loss has to be defined. The idea is to use the binary cross entropy loss with a slight change:

around a streak, we define a zone of uncertainty, where the loss is set to 0.
The definition is the following:

- If the ground truth pixel (obtained from last prediction) is in a streak, we expect a streak at this place.
- If the ground truth pixel is not in a streak, but is close to be in a streak, we set the loss to 0.
- If the ground truth pixel is not in a streak and is far from being in a streak, we expect no streak at this place.

We defined the loss this way since the model make very few false positives, which means that when a streak is detected, it is likely to really be a streak. We added a zone of uncertainty since sometimes just parts of the streak were detected, then the zone close to the streak might be part of the streak.
In order to define the uncertainty zone, a dilation was performed on the streak, then the dilated part is taken as uncertain.
Between each iteration, an opening is performed on the prediction in order to remove small artifacts that could have been detected as streaks (likely stars). An opening with a disc kernel would remove those artifacts without removing the lines. The result of the opening is taken as the new ground truth after thresholding. Several new hyper-parameters come into play: the kernel size of the dilation, the disc size for the opening, the threshold chosen to discretize the prediction, and the number of epochs of training at each iteration.

3.6.2 Grid search

In order to choose the different hyper-parameters, a grid search using 4-fold cross validation is used.
The optimal parameters are the following: 5 epochs of training at each iteration, a disc of 5 pixels for the opening, a disc of 41 pixels for the dilation, and a threshold varying between 0.1 and 0.8 for 17 iterations and then fixed at 0.8.

As can be seen on Figure 3.6, most of the real streaks are correctly detected. The mean IoU of 0.70 confirm this visual feeling. This method shows good results but those are still uncertain due to the lack of real examples.
We were advised to try the network HT-LCNN to work on this problem so we discuss this method in the next Chapter.

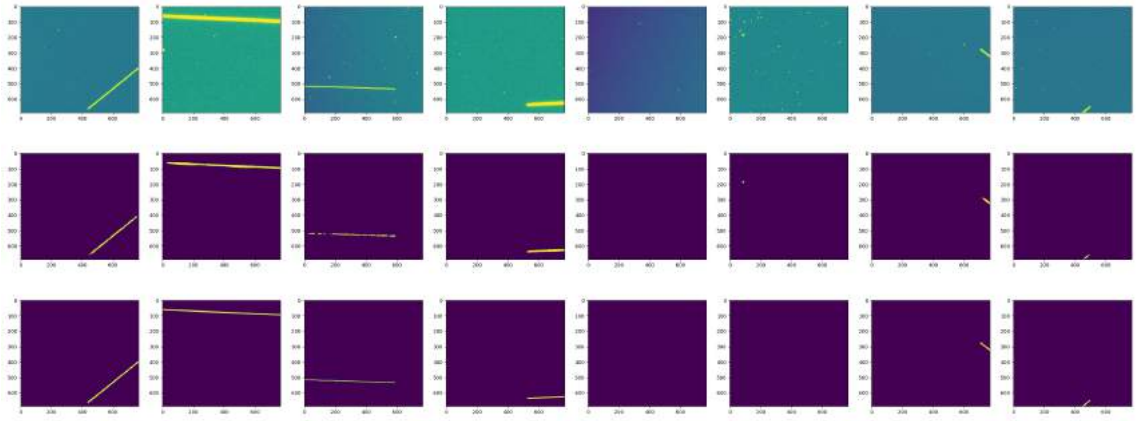


Figure 3.6: Sample results of the iteratively trained U-Net on some real streaks. The original patches are on top, the ground truths are on the bottom, and the predictions are in between.

Chapter 4

Method 2 - Deep Hough Transform Line Priors

4.1 Introduction

Classical work on line segment detection is knowledge-based; it uses carefully designed geometric priors using either image gradients, pixel groupings, or Hough transform variants. Instead, current deep learning methods do away with all prior knowledge and replace priors by training deep networks on large manually annotated datasets. Here, the dependency on labeled data is reduced by building on the classic knowledge-based priors while using deep networks to learn features. Line priors are added through a trainable Hough transform block into a deep network. Hough transform provides the prior knowledge about global line parameterizations, while the convolutional layers can learn the local gradient-like line features [4].

The Hough transform is widely known in image processing to extract features such as lines or other arbitrary shapes. It parameterizes lines in terms of two geometric terms: an offset and an angle, describing the line equation in polar coordinates.

The network HT-LCNN [4] is largely based on the previously developed L-CNN [5]. We will explain first explain how L-CNN works.

4.2 L-CNN

Existing researches address the wireframe parsing problem with two stages (wireframe parsing is detecting lines and their junctions). First, an input image is passed through a deep convolutional neural network to generate pixel-wise junction and line heat maps. After that, a heuristic algorithm is used to search through the generated heat map to find junction positions, vectorized line segments, and their connectivity. The heuristic algorithms can be sub-optimal, so they decided to build an end-to-end trainable network with the goal of pushing the state-of-the-art performance for wireframe parsing.

In order to train their network, the ShanghaiTech dataset is used. An sample image is shown in Figure 4.1.

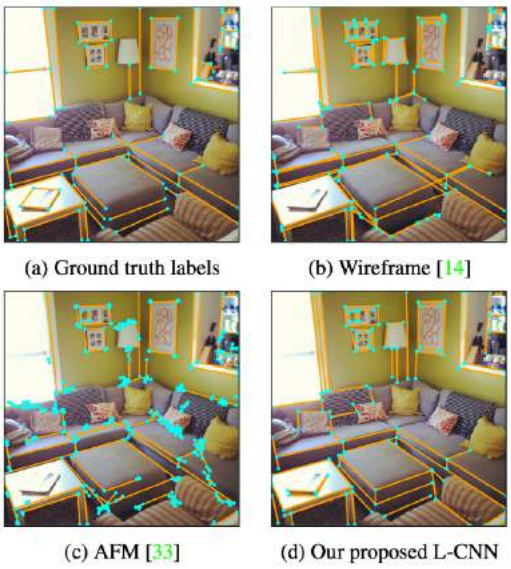


Figure 4.1: Sample image from the ShanghaiTech dataset. Wireframe and AFM are previous algorithm developed to parse wireframe.

4.2.1 L-CNN architecture

Taken an RGB image as the input, the neural network directly generates a vectorized representation without using heuristics. The L-CNN architecture is illustrated in Figure 4.2. It contains four modules:

- a feature extraction backbone that takes a single image as the input and provides shared intermediate feature maps for the successive modules;
- a junction proposal module which outputs the candidate junctions;
- a line sampling module that outputs line proposals based on the output junctions from the junction proposal module;
- a line verification module which classifies the proposed lines.

The output of L-CNN are the positions of junctions and the connectivity matrix among those junctions. The mathematical details of each module are explained in [5].

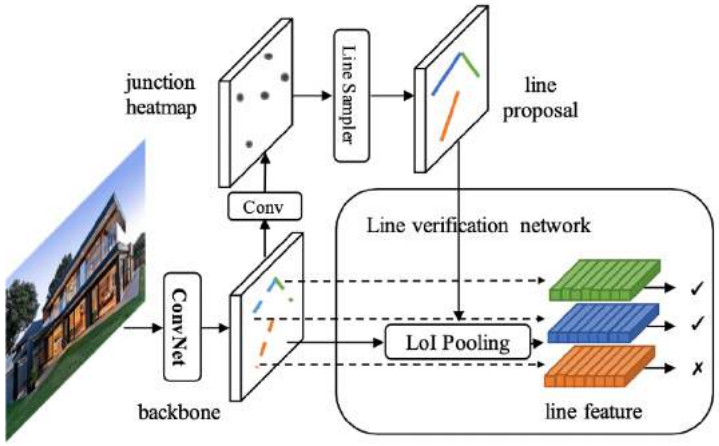


Figure 4.2: L-CNN architecture illustration

4.2.2 Evaluation metrics

Previously, researchers use two metrics to evaluate the quality of detected wireframes: the heat map-based average precision (AP^H) to evaluate lines and junction average precision (AP) to evaluate junctions. These metrics are problematic in wireframe detection since they are based on the heatmap: they do not penalize for overlapped lines (Figure 4.3a), and they do not properly evaluate the connectivity of the wireframe (Figure 4.3b).

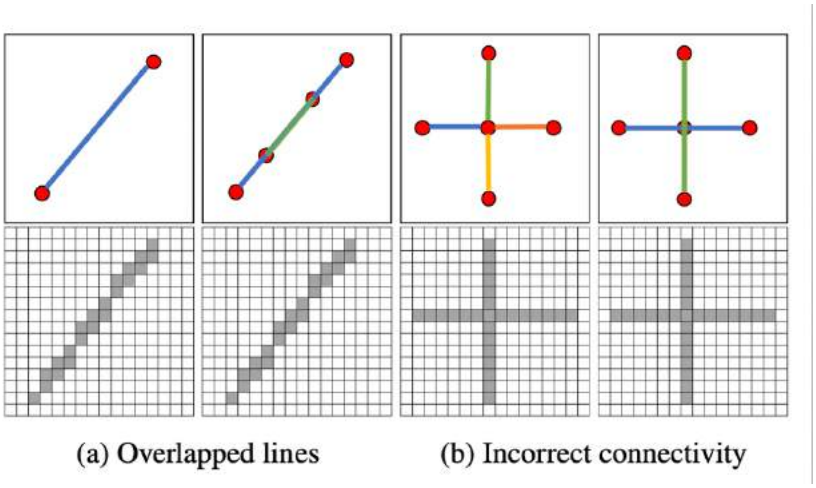


Figure 4.3: Illustration of the issues the old metrics have.

A new evaluation metric defined on vectorized wireframes rather than on heat maps is proposed [5]. The metric is called structural average precision (sAP). Structural AP is defined to be the area under the precision recall curve computed from a scored list of the detected line segments on all test images.
The junction mean AP (mAP^J) they use evaluates the quality of vectorized junctions

of a wireframe detection algorithm without relying on heat maps.

They show that their network have state-of-the-art performance on the ShanghaiTech dataset [5].

4.3 HT-LCNN

4.3.1 The HT-IHT block

In order to add line priors to the network L-CNN, [4] has developed a trainable Hough transform block called HT-IHT. This block is put in the network backbone and replace the stack hourglass blocks. An illustration of the added line priors is shown in Figure 4.4. The goal is to reduce the dependency on labeled data by building on the classic knowledge-based priors while using deep networks to learn features.

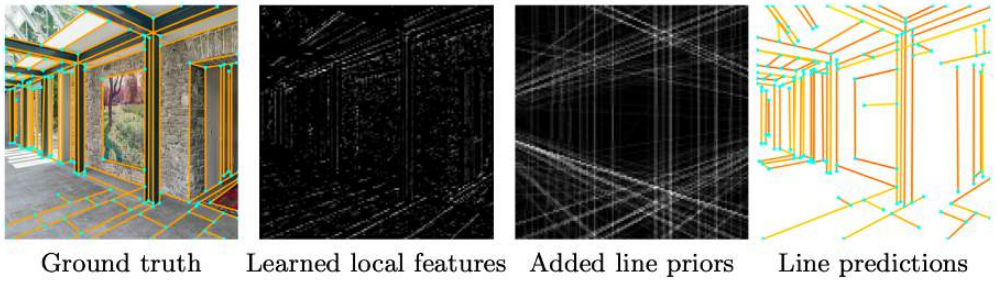


Figure 4.4: Line priors illustration

The HT-IHT block is end-to-end trainable and is composed of a Hough transform, several convolutions in the Hough domain, and the inverse Hough transform. This is shown in Figure 4.5.

Several experiments are done in [4] to show that their network provides state-of-the-art performances, especially when working with less data.

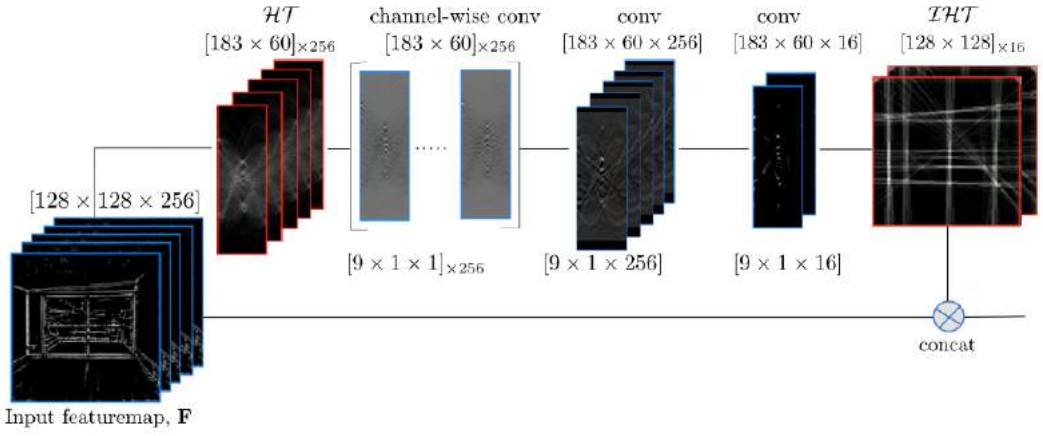


Figure 4.5: HT-IHT block illustration

4.3.2 Pretrained model performance

As [4] gives the weights of their network trained on the ShanghaiTech dataset, we decided to give a shot to their pretrained model without any further training. Some predictions results are shown in Figure 4.6 and 4.7. As can be seen, the results are very impressive: all the streaks shown here are perfectly predicted.
At this time of the semester, some new real streaks were available (93 real streaks in total). This allowed us to test this network on more data. Moreover, a visual inspection of the newly available streaks led us to change a bit the way the synthetic streaks are created to make them more realistic.
With the pretrained weights, the sAP was evaluated to 0.812. The sAP is evaluated here on all the 93 real streaks. To make a comparison with L-CNN, the sAP of L-CNN with their pretrained weights is evaluated to 0.800. A slight improvement is observed with HT-LCNN.

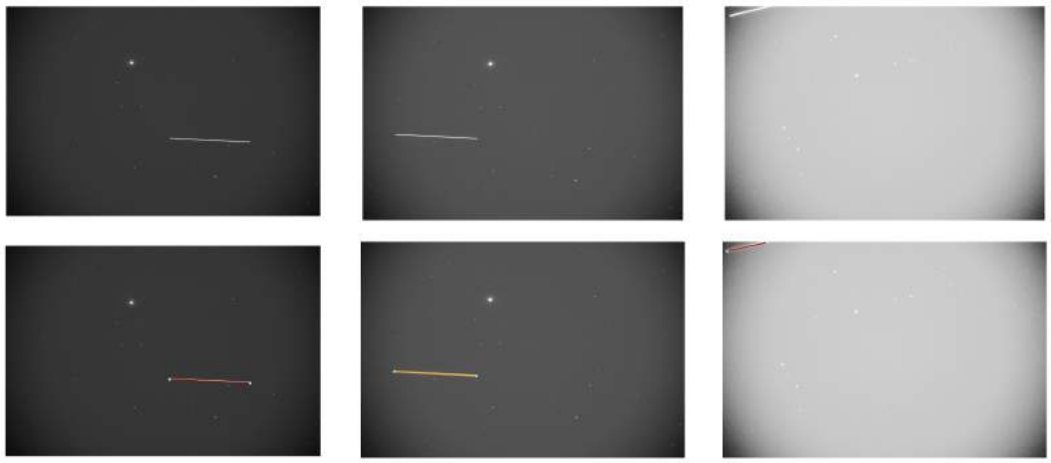


Figure 4.6: Sample images and the line predictions using the pretrained model

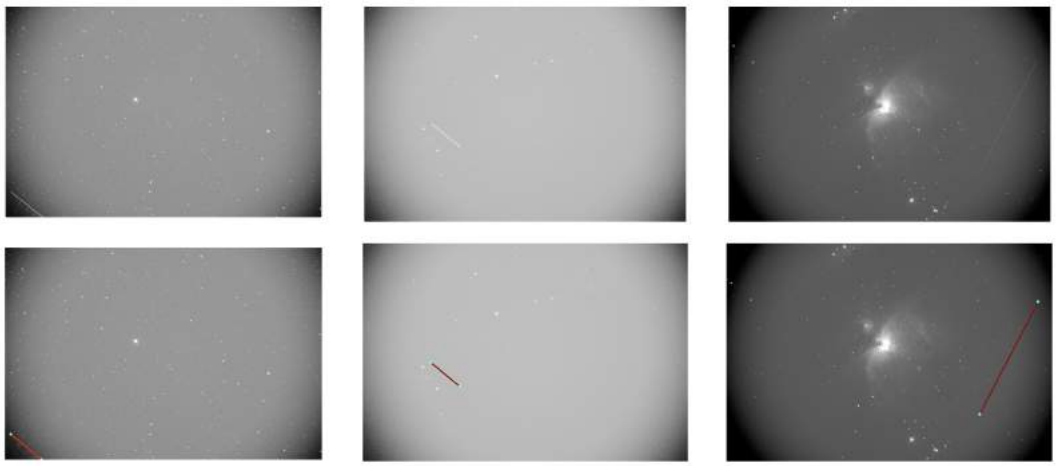


Figure 4.7: Sample images and the line predictions using the pretrained model

4.3.3 Sensitivity analysis

In order to evaluate the robustness of the model with respect to noise and to the intensity of the streak, 2 sensitivity analyses are performed.

Sensitivity to salt and pepper noise

Salt and pepper random noise is added to sample images with different probabilities. As can be seen on Figure 4.8 and 4.9, the algorithm is capable to find the streaks even with some noise. Here until a probability of 0.4 but it varies with the different streaks. The other streaks not shown here have comparable results.

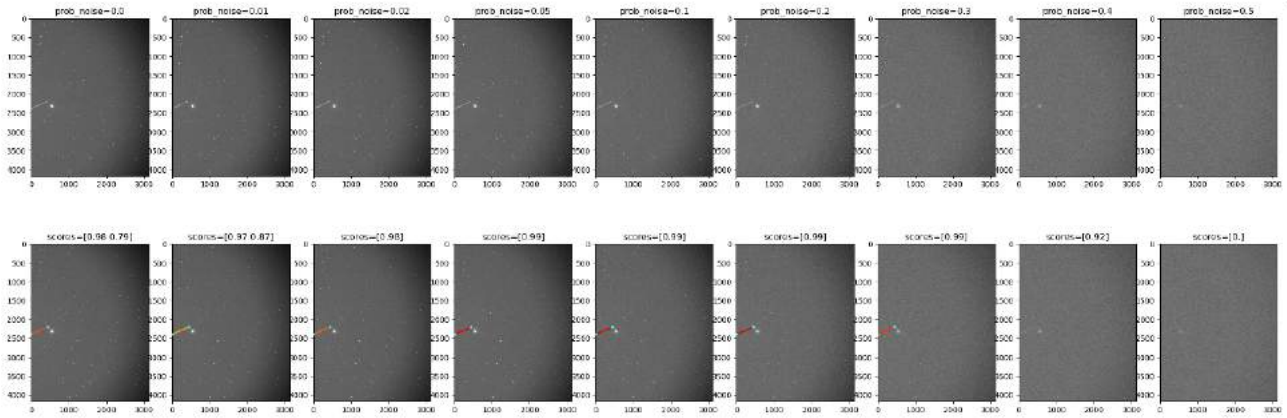


Figure 4.8: Sample image with more and more salt and pepper noise added

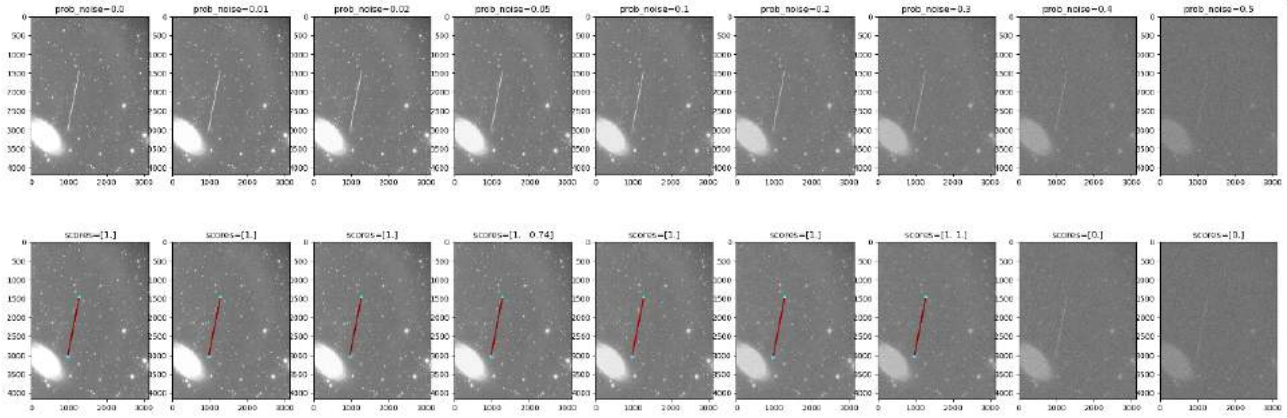


Figure 4.9: Sample image with more and more salt and pepper noise added

Sensitivity to streak intensity

Streaks with varying intensities are generated. As can be seen on Figure 4.10 and 4.11, the algorithm is capable to find the streaks even when the streak has a low intensity. Here until an intensity of 120 and 160 but it varies with the different streaks. The other streaks not shown here have comparable results.

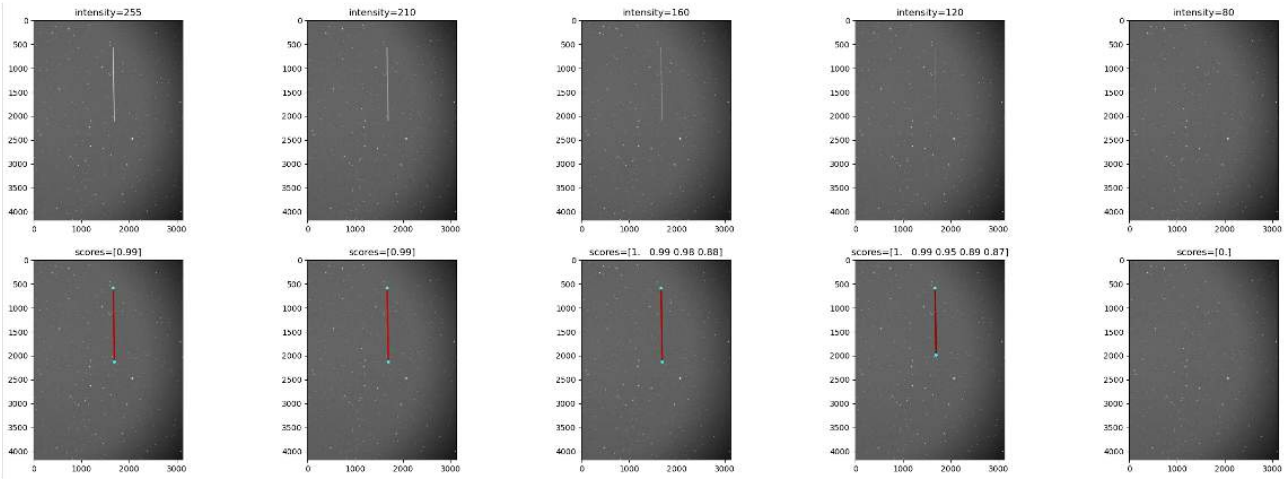


Figure 4.10: Sample image with a streak with less and less intensity

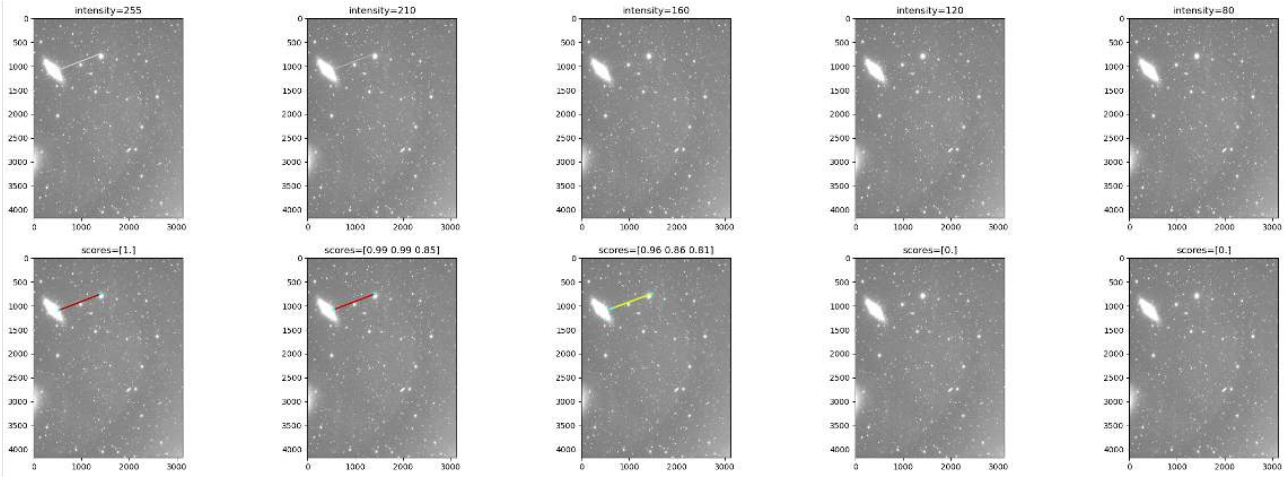


Figure 4.11: Sample image with a streak with less and less intensity

This shows that even with weights pretrained on another dataset, this network is quite robust at detecting the lines in our dataset.

4.3.4 Fine tuning

In order to increase the performance of the model, we fine-tuned the model on the real streaks that were available. This required us to annotate the 93 streaks but this network take as input the two endpoints of the line, thus the annotation is much easier to do. As the network needed to have at least two streaks on an image for the training to work [1], we added one synthetic streak on each image. Each image is then composed of a real and a synthetic streak.

[1]More precisely, it needed to have at least 2 junctions that were not connected by a line. Adding a second streak solves this issue.

The sAP is then evaluated only on the validation set of the real streaks (not containing the added synthetic streak). As a reference, the pretrained HT-LCNN has a sAP of 0.844 on this validation set.

The fine-tuning is ended when the validation loss start to increase again, and the epoch with the best validation loss is kept. The sAP obtained after fine-tuning is 0.988. This shows that the training was improved the model for our dataset. In order to have reliable results, all images of the same streak (the streak appearing at different places on the image) have been put in the same set (training or validation).

The annotation being quite easy to do and not so time consuming, the SSA association will be able to retrain the network in the future when they will have more real streaks available. This would make the network more robust to unseen data.

This method directly outputs the two endpoints of the streaks and is easily re-trainable with more data. Moreover, this method is less dependant on hyper-parameters that may need to be adjusted if the dataset changes. This method is then more reliable and easier to work with.

4.4 Further work

In order to compare the two methods described in this report, both should be tested in the same conditions. However, to have reliable results, it would require annotating all the 93 streaks pixel-wise to evaluate the performance of the first method. As is it quite time-consuming, it was not possible to do it before the end of the semester. This being said, the second method looks more promising since it is easily scalable to a larger dataset in the future. It also has excellent results on the dataset we were provided.

Chapter 5

Conclusion

Two methods have been tested to detect the satellite streaks in astronomical images. We have seen that with the U-Net based approach, we were able to reach good results using synthetic streaks for the training, then an iterative training allowed us to fine-tune the model in an unsupervised way. However the lack of real samples at this time and the difficulty to create annotations to check the model performance make this method less attractive. Despite the results being good on our dataset, we can't say for sure that the model is robust to unseen data. On the other hand, the second approach based on the network HT-LCNN showed very promising results with low effort. Even the pretrained model on the ShanghaiTech dataset performs well. The fine-tuning of the model on our dataset is straighforward to do and yields even better results.

Bibliography

[1] Yann Bouquet. Delineating satellite tracks in astronomical images. 2020.

[2] Alexandre Di Piazza. Study on the detection of slow satellite tracks. 2021.

[3] Kun Huang, Yifan Wang, Zihan Zhou, Tianjiao Ding, Shenghua Gao, and Yi Ma. Learning to parse wireframes in images of man-made environments. 2018.

[4] Yancong Lin, Silvia L Pintea, and Jan C van Gemert. Deep hough-transform line priors. 2020.

[5] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *ICCV 2019*, 2019.