A CONVOLUTIONAL NEURAL NETWORK APPROACH TO STAR SENSORS IMAGE PROCESSING ALGORITHMS

Marco Mastrofini
Sapienza University of Rome
10 PUBLICATIONS **18** CITATIONS
SEE PROFILE

Francesco Latorre
Sapienza University of Rome
6 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Ivan Agostinelli
Sapienza University of Rome
12 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Fabio Curti
Sapienza University of Rome
103 PUBLICATIONS **615** CITATIONS
SEE PROFILE

AAS 21-589 (PREPRINT)

A CONVOLUTIONAL NEURAL NETWORK APPROACH TO STAR SENSORS IMAGE PROCESSING ALGORITHMS

Marco Mastrofini[*], Francesco Latorre[†], Ivan Agostinelli[‡], Fabio Curti[§]

The present work will investigate an Artificial Intelligence (AI) approach in the framework of image segmentation and clustering algorithms for star sensors applications. It focuses on the architecture development and test of a Convolutional Neural Network (CNN) based algorithm and results comparison with the state of the art. The problem of image segmentation will be faced using the U-Net to detect the brightest objects in the sensor's Field Of View (FOV) for attitude determination purposes. The dataset creation for the network training, algorithm design process and definition of performance indices are provided together with comparison test results.

INTRODUCTION

The problem of star segmentation and accurate centroids estimates from images is a crucial point for the on board autonomous star sensors algorithms.[1] Indeed both star identification algorithms and star tracking ones require precise estimates for attitude determination purposes.[2] To improve the performances of a star identification algorithm, an initial processing of images data is mandatory because the reduction of false stars, SEUs and straylight noise would increase the star identification rate.[3,4]

The scope of this work is the development of a star detection algorithm capable of filtering the faintest objects from background in several signal-to-noise levels scenarios. In particular this algorithm aims to provide information about the brightest objects, in order to facilitate star identification and reduce the memory storage burden. The algorithm design aims to have few modules to reduce the algorithm size, complexity and hardware implementation difficulties in order to provide a reliable star detection product.

In this work, the road of AI algorithms has been investigated to take advantage of their capability of learning specific tasks and reducing algorithm complexity. Moreover, their good performances against unpredictable situations make them suitable for this application; this could avoid the problem of algorithm re-calibration to process successfully images in different noise conditions.

[*]PhD student, School of Aerospace Engineering, Sapienza University of Rome, Via Salaria 851, 00138 Rome, IT, email: marco.mastrofini@uniroma1.it

[†]Graduate student, School of Aerospace Engineering, Sapienza University of Rome, Via Salaria 851, 00138 Rome, IT, email: latorre.1597476@studenti.uniroma1.it

[‡]Graduate student, School of Aerospace Engineering, Sapienza University of Rome, Via Salaria 851, 00138 Rome, IT, email: ivan.agostinelli@uniroma1.it

[§]Associated Professor, School of Aerospace Engineering, Sapienza University of Rome, Via Salaria 851, 00138 Rome, IT, email: fabio.curti@uniroma1.it

Contributes of this work are:

- Creation of a balanced dataset of real and simulated star images for training, validation and test of the CNN block.
- Design of a AI based star detection algorithm capable of providing the most significant information for star identification and tracking algorithms.
- Definition of performance indices to assess the accuracy of a segmentation algorithm prediction against the target image mask.
- Provide results of CNN training and test over simulated test images and compare performances against a classical algorithm.

The work is organized as follows: a brief overview of CNN and U-Net model is described in the next section together with the proposed dataset and our proposed segmentation-clustering algorithm scheme. Moreover the performance indices mentioned before are introduced and a description of a classical reference algorithm is provided. As a last section, the results of our algorithm are presented: these include the U-Net accuracies and optimal performance with respect to our chosen thresholding technique. This is followed by comparison with the classical algorithm and conclusion with future developments.

METHODOLOGY

The approach used here to face the segmentation of star trackers images is based on Machine Learning (ML). This AI area is being extensively explored and applied for image processing due to great improvements achieved in several tasks. In particular, CNN have been developed for computer vision algorithms because of their capability of image data processing and information extraction. Actually, lots of them are available to be used as Vgg-16 and Vgg-19,[5] Inception-v3[6] from Google , the U-Net,[7] AlexNet[8] and many others. CNN have shown great performances in achieving many tasks as objects detection and classification, semantic segmentation and pose problem involving both cooperative and non cooperative targets.

The problem faced in this work is an instance of semantic segmentation. Indeed here the scope is dividing the original image pixels in two groups: over threshold pixels and under threshold pixels belonging respectively to the white area and black area of the segmented output. This result will be achieved through the application of the U-Net to get a final output which contains as white areas just the brightest objects in the sky, avoiding straylight noise, faintest stars and objects on the background and minimizing the overall noise signal on the image. The reason why this is done through a ML based algorithm is due to the CNN capability to cope well with objectives and targets easily understandable for a human being but very difficult sometimes to formulate in a rigorous fashion as the one of segmenting just stars which stand out the most in a night sky image. This is the target purchased in this work.

Convolutional Neural Networks

CNNs are heavily applied whenever complex information extraction from images is required. In a rude way they are stacks of convolution and maxpooling layers. They are capable of learning high and low level features of an image regardless of their position and orientation. Another great

advantage of a CNN is that the weights to be learned are shared between the layers, minimizing the needed memory storage.
The convolutional layer works in the following way: a convolution operation is done by sliding a small window (typically 3×3 pixels) over height and width of the image and over its color channels. The window creates a *patch feature map*, which is multiplied with a learned weights matrix called *convolution kernel*. This operation could cause image size reduction, which is handled by applying *padding*, which means to add border pixels in order to make the output size the same as the input. The neurons contained in the network layers are activated through an *activation function*: ReLU (Rectified Linear Unit), *softmax* and *sigmoid* are typically used.
The pooling layer has the aim to downsample the image in order to optimize the learning of features. In particular, the max pooling layer takes a portion of the image (typically a 2×2 window) and substitutes it with the maximum pixel value.
The performance of a CNN is represented by values of *loss* and *accuracy*, which are computed during training, validation and test processes. It is desirable to obtain similar accuracies in all these phases, to avoid *overfitting* and generalize network performances against never seen data. There are different ways to overcome this issue: these include L2 weight regularization,[9] batch normalization,[10] data augmentation[11] and dropout.[12]

U-Net

The U-Net is a Fully-Convolutional Neural Network (FCN) which was first used for segmentation of biomedical images, and was later adapted to space-based applications such as crater detection.[13] The network architecture in this paper, is chosen to be a slightly modified version of the original one, is shown in Figure 1 and it has a symmetric encoder-decoder structure: the encoding, downsampling path is a stacked sequence of two ReLU 3×3 convolutional layers followed by a 2×2 max pooling layer. At each level, the number of filters doubles, reaching its maximum value at the bottom.
The decoding, upsampling path contains 2×2 up-convolutional layers concatenated with layers coming from the corresponding downsampling level, in order to preserve already learned features. The concatenated layers are followed by *dropout* and a sequence of two 3×3 convolutional layers up to the the output layer, in which a final *sigmoid*-activated 1×1 convolution produces data ready to be binarized. The present U-Net will be fed with 512×512 images and their corresponding masks.

Configuration parameters used for this Network to prevent overfitting are:

- *Learning Rate*. It is an optimizer's parameter that fixes the step size at each iteration during minimization of the loss function.
- *Regularization Factor*. It is a parameter needed for the *l2 regularization* based on penalization of the cost function.
- *Dropout Rate*. It regulates the percentage of inactive network elements in the dropout layers during training and validation process.
- *Kernel initializer*. It sets the weight initialization method: in this case it is set on *he_normal*.[15]

In this work, *Adam* was chosen as optimizer and *Binary Crossentropy* was chosen as *loss function*.

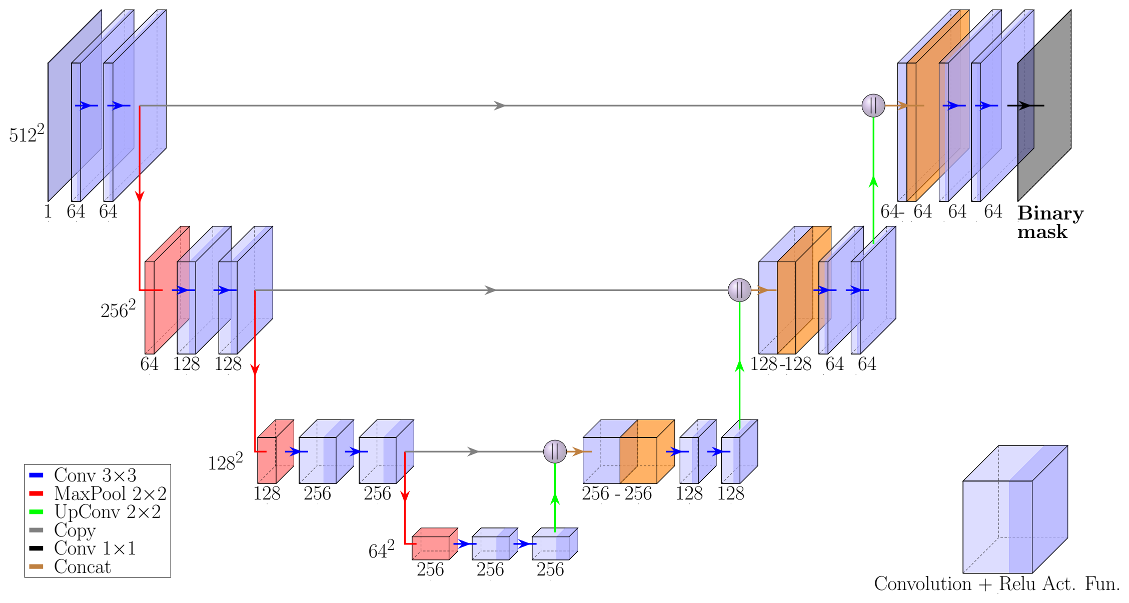

Figure 1. U-Net model with features dimensions and number of features.[14]

Dataset Creation and Input Data Preprocessing

Here a description of the dataset created and used for the U-Net training phase is provided. It is composed of 5600 squared monochromatic jpeg images and 5600 associated jpeg masks. All of them have a size of 512 × 512 px and a bit depth of 8. Masks' pixels can have one of two possible values: 0 for under threshold and 255 for over threshold pixels.

The samples for training and validation phases are the first 5300 ones. They are generated from a batch of 600 images to which 180°, 90° clockwise rotation, added noise, increased straylight, blur effects, increased luminosity, contrast and further modifications have been applied. This original batch is made of 300 samples coming from several night sky acquisition campaigns and 300 samples randomly acquired using Stellarium[16] software in night sky conditions considering different FOV and attitudes (Figure 2). Acquisition campaigns were performed using several kind of cameras and objectives: ZWO ASI 120MM-S camera with default optics, reflex Nikon D3100 equipped with a Nikkor 18-105 mm and a ProLine PL16803 camera coupled with an Officina Stellare RiFast 400 telescope.

Last 300 dataset's images have been obtained with a High Fidelity Star Tracker Image Simulator. It is used to simulate realistic night sky images, simulating all the physical, functional and geometrical characteristics of the star sensor, along with all the instrumental and environmental noises.[17–20] Simulator images have been used only in test phase to provide independent samples the network has never been trained on, to monitor the generalization capability of the trained network.

The dataset contains images of night sky where stars are the main actors but where planets, light pollution caused straylight noise, Single Event Upsets (SEUs), clouds, airplanes' streaks, satellites, comets, nebulae and galaxies appear too. Masks have been obtained using specific thresholds for different groups of images from different campaigns, sensors and software. In particular thresholds selection has been carried out using Adobe Photoshop CC 2019 with the scope of getting just the

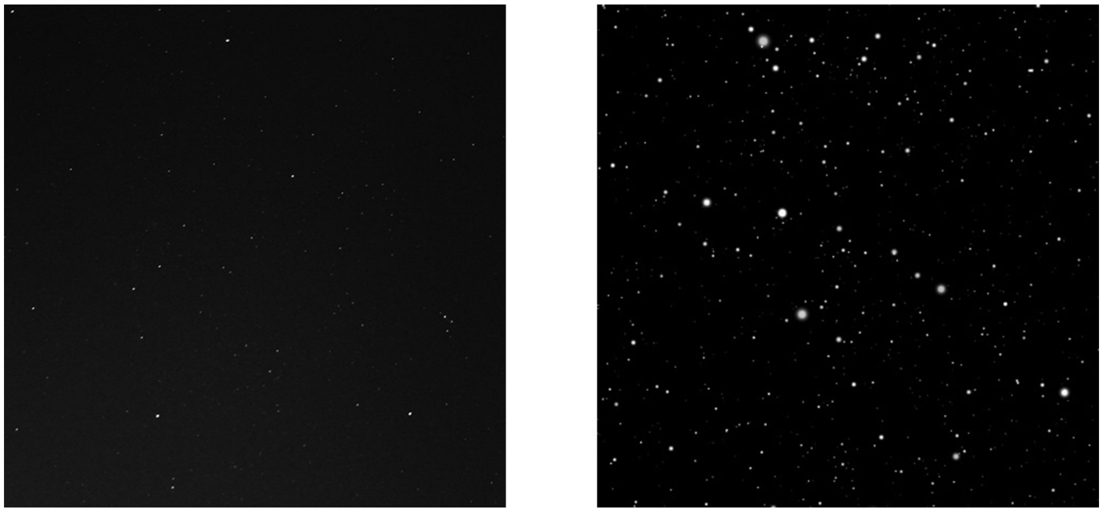

Figure 2. Dataset's samples: real night sky image (on the left) and Stellarium generated night sky (on the right)

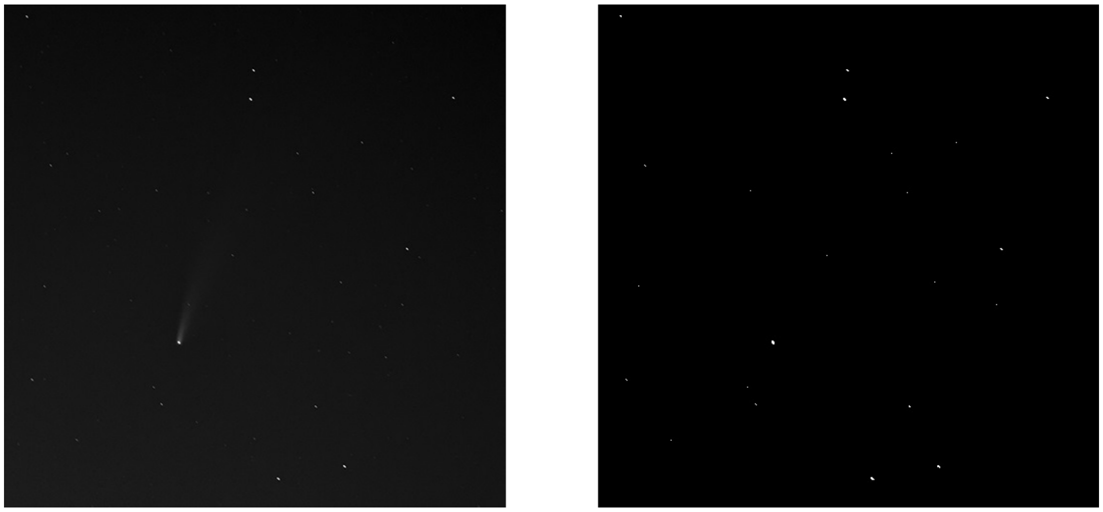

Figure 3. Dataset's samples: real night sky image (on the left) and relative mask (on the right)

brightest objects in the FOV and filtering all the noise and undesired objects cited above. An example of dataset image and mask is provided in Figure 3, where it is clearly visible that just the brightest points appear inside the mask. Input images and masks preprocessing is performed in order to organize these data in two float 4D tensors for neural network training and test. Every image is converted into floating point arrays and normalized using its maximum value to carry on an image adaptive normalization process: every most significant image pixel will have an associated value close to 1. This will help the network to detect the most significant pixels and make the learning process easier for the net by constraining the interval of signal values between 0 and 1. Moreover, this normalization process improves the algorithm capability of working with images that have different range of energy levels. The same process applies for masks data vectorization and normalization with the only difference that the normalizing factor is constant and equal to 255 for every mask.

Algorithm Design and Configuration

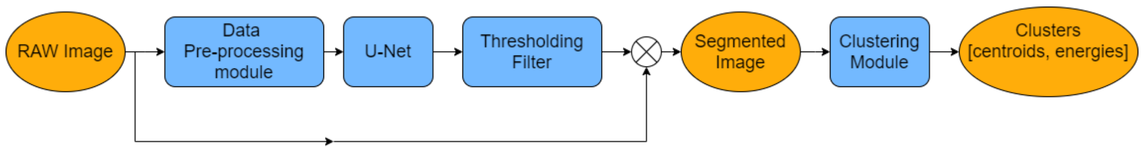

Figure 4. Image segmentation algorithm scheme: data are in orange and modules are in light blue.

The segmentation algorithm scheme is shown in Figure 4. The monochromatic raw image from the camera arrives to pre-processing module where it will be resized, normalized and vectorized into a floating point array. This module will be tailored according to specific sensors' bit depth while the size must be the one the NN is trained on. Then, the array is processed by the trained U-Net which gives its output prediction. This is a 2D array where every pixel has an associated value of probability p of being over threshold. If $p \geq p_{min}$ then this value is rounded up to 1. After the thresholding filter, a binary image is obtained and its element product with the original image will provide an array with just active pixels' energy values. This output will be used by the clustering algorithm to detect clusters and computing their centroids and total energies. The clustering module developed within this work takes all the active pixels in the segmented image and organizes them in a list. Then it associates the first pixel in the list to the first cluster and starts to verify if the next pixels are part of the same cluster or not through a distance based criterion. Whenever the next active pixel does not belong to the already identified clusters, then a new cluster is identified.

The distance criterion uses the euclidean norm and the condition to assess the membership of two active pixels ($\mathbf{p}_i$ and $\mathbf{p}_j$) to the same cluster is expressed by Equation 1.

$$\|\mathbf{p}_i - \mathbf{p}_j\| \leq \sqrt{2} \tag{1}$$

The clustering algorithm then first performs a filtering action with a minimum and maximum dimension and then a sorting of all the clusters. Minimum dimension filter is needed to avoid possible noise signal in the clustering outputs (ex. hot pixels) while the maximum dimension filter is needed to avoid great detected clusters due to non stars objects. The sorting operation is then performed in descending order and considers the first N clusters in output from the previous filtering actions. It is based on the dimension to consider the greatest stars in the FOV while N is computed by applying a 1.25 factor to the average number of stars which depends on the FOV and cut-off magnitude of the selected sensor.[2] The maximum number of clusters in output is fixed a priori to process just the most significant ones and the 25 % margin factor is selected to take into account possible uncertainties of the average number of stars formula.

The U-Net is capable of resolving background and foreground pixels after training, validation and test phases but it still gives continuous values that have to be discretized. This is the reason why of Thresholding Filter and the selection of a reasonable value for p_{min} will be done in the next chapters in order to get the best correspondence between algorithm prediction and targeted masks. Once the U-Net is trained and p_{min}, FOV and cut-off magnitudes are chosen, the algorithm will be compared with a classical image segmentation one described in next section both in terms of segmentation performances and clustering ones.

Classical Image Processing Algorithm Description

In this framework, a state of the art image segmentation algorithm for star sensors is chosen for a reasonable comparison with the AI based and proposed algorithm. This segmentation algorithm can be configured using a static or a dynamic approach[21] for the background noise estimation.[22] In this work, a dynamic approach based on a zigzag local thresholding with moving average is used.[23] In particular, a pixel is saved if its energy value is greater than the background noise, evaluated through a moving average line-by-line, plus a threshold τ_{pre}. Data storage is performed using Run-Length Encoding (RLE)[23] algorithm. RLE is a form of lossless data compression in which runs of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and length. For example, considering an image containing black text on a white background, there will be many long runs of white pixels in the blank space and many short runs within the text. A hypothetical scan line, with B representing a black pixel and W representing a white pixel, might be read as:

WWWWWWWWWWWWBWWWWBBBWWWWWWBWWWWWWWWWWWWWW

With a RLE data compression algorithm applied to the above hypothetical scan line, it would be rendered as: 12W1B4W3B6W1B14W.

This can be interpreted as a sequence of twelve Ws, one B, four Ws, three Bs, etc. The RLE allows to represent the original 41 characters in only 16. While the actual format used for the storage of images is generally binary rather than ASCII characters like the ones in the example above, the principle remains the same. The RLE algorithm stores only the coordinates of the first pixel of the detected segment and its length rather than the coordinates of all the pixels belonging to the detected segment, reducing the storage burden. Furthermore, the RLE algorithm is performed, as mentioned above, changing the direction of encoding line by line (which explains the name *zigzag*) to avoid reading biases. In particular, if the i^{th} line is read from left to right, the $i^{th}+1$ line will be read from right to left. The outputs of the preprocessing are the coordinates $\mathbf{p} = [p_y, p_z]^T$ of the first pixel (from left to right) of each selected segment along with their length and energy $\mathbf{E}(\mathbf{p})$, obtained subtracting the background noise to the signal intensity of the pixel.

The information relative to the segmented image is used for the clustering process, i.e. the combination of segments belonging to the same star streak.[17–20] The most important step required by the clustering algorithm is the evaluation of the *primitive clusters*, containing pixels which share at least one corner. A suitable technique has been developed in order to relate streaks in the image to the same star if the corresponding streak is broken in multiple streaks due, for example, to the high rate of the spacecraft. Three conditions shall be satisfied, related to the minimum distance, the direction and the density of the considered primitive clusters. Centroids' coordinates of each cluster are evaluated taking into account the energies of segments merged to build the cluster itself, according to Equation 2:

$$\mathbf{p}_{cluster,i} = \sum_j \left(\frac{E_{seg,j} \cdot \mathbf{p}_{seg,j}}{E_{cluster,i}} \right) \tag{2}$$

where, $\mathbf{p}_{seg,j}$ are the baricenter coordinates of the j^{th} segment in the image plane, while $E_{cluster,i}$ is the sum of the energies $E_{seg,j}$ which forms the cluster i^{th}:

$$E_{cluster,i} = \sum_j E_{seg,j} \tag{3}$$

Definition of Comparison Indices

The comparison between performances obtained with the U-Net based segmentation algorithm and with the classical one will be made in the Results section. Both the algorithms will be tested with a set of gray scale images and relative masks; in order to assess the accuracy of the algorithm's predictions against test masks, the definition of suitable indices is necessary. The indices which are going to be described require as input data two masks related to the same gray scale image: prediction mask coming as U-Net output and ground truth from the dataset. These indices can be also used to compare predictions of any couple of segmentation algorithms. Before their introduction a few quantities must be characterized:

- $cnt0_{prd}$: number of under threshold pixels in the predicted mask.
- $cnt1_{prd}$: number of over threshold pixels in the predicted mask.
- $cnt0_{trg}$: number of under threshold pixels in the target mask.
- $cnt1_{trg}$: number of over threshold pixels in the target mask.

Moreover, the sum of predicted and target array of binary numbers is needed and this sum will show values that can be 0,1 and 2. Following quantities about masks' sum need to be defined:

- $cnt0$: number of 0 pixels.
- $cnt1$: number of 1 pixels.
- $cnt2$: number of 2 pixels.

It must be said that a good prediction has under threshold pixels and over threshold ones almost in the same position inside an array as the ground truth masks. An ideal one should give a value of $cnt1$ equal to 0 because this means perfect superposition between algorithm's prediction and target masks. Anyway, a slight difference always occurs but to have a good prediction it is important to have $cnt1$ the lowest possible. An index to define is the following one:

$$G = \frac{cnt0 + cnt2}{N_p} \times 100 \tag{4}$$

It represents the percentage of the actual matched pixels by prediction with respect to the whole number of pixels in an image N_p. The lower is $cnt1$ and the higher is G (with maximum value equal to 100% under ideal conditions). This index, in the framework of night sky camera images shall have always a high value because of the high percentage of under threshold pixels and few over threshold ones. Another index to be defined is the following one:

$$G_0 = \frac{cnt2}{cnt1 + cnt2} \times 100 \tag{5}$$

It is the G index without the $cnt0$ information. The purpose of it is to have a measure of the matched useful signal (over threshold pixels, $cnt2$) with respect to their sum with the uncertainties ($cnt1$). This index shall be higher than 50% for a good prediction because a lower one could mean having more uncertainties than the detected useful signal. Sometimes if the algorithm predicts few more stars than the target mask one, G_0 could have a value lower than 50% but the prediction would still be a good one. In order to recognize if a situation with $G_0 < 50\%$ still is a good one, a third index is defined:

$$M = \frac{cnt1_{prd}}{cnt1_{trg}} \tag{6}$$

This index directly compares the over threshold pixels number between the prediction and target masks. It is the only index that directly uses information of the single masks and not info about their intersection. In this way these two possible scenarios can be distinguished:

- $G_0 \simeq 50\%$ and $M \geq 1$: More star detected than reference mask (acceptable prediction).
- $G_0 < 50\%$ and $M < 1$: Less star detected than reference mask (not acceptable prediction).

It must be also pointed out that the previous first scenario could mean a relevant amount of undesired detected noise by the segmentation algorithm too. To distinguish this undesired condition from a good one, the first index G is needed:

- $G_0 \simeq 50\%$, $M \geq 1$ and $G \simeq 100\%$: It means that more stars have been detected with respect to target mask (acceptable prediction).
- $G_0 < 50\%$, $M \geq 1$ and $G \ll 100\%$: It means that a lot of undesired noise has been detected with respect to target mask (not acceptable prediction).

In conclusion, all of these three indices are needed and will be used later to make algorithm's design choices and comparisons.

RESULTS

In this section results of U-Net's training, validation and test will be shown together with a reasonable choice for p_{min} and number of initial filters. Then a comparison test of the algorithm with a classical one is made against the same set of images both at segmentation and clustering level.

U-Net Training and Test

The U-Net tested here is configured with the values shown in Table 1.

Table 1. U-Net Configuration Parameters

Parameter	Value
Image size	512
Learning Rate	10^{-4}
Regularization Factor	10^{-5}
Dropout Rate	0.25
Kernel Size	3
Kernel initializer	$'he\_normal'$

Training and validation were performed in Tensorflow Keras[24] considering 3 *epochs* and a dimension of the training batch equal to 3. This has been done for three different values of initial filters: 16, 32 and 64. The reason why this choice was made is to see how the performances in terms of accuracy and loss vary with the increasing network complexity in order to select the minimum required number of filters while achieving satisfactory performances. Every trained network shows reached accuracies of 99.97% after the first epoch and remain constant for training, validation and test. The same behavior applies for the final losses which are not higher than 0.01. Moreover, no overfitting phenomenon shows up.

The training, validation and test were performed on the following workstation:

- CPU: AMD Ryzen 5 3600X-6 Core 3.79 GHz
- RAM: 32 GB
- GPU: AMD Radeon 5600 XT.

Tuning of Thresholding Filter

From the previous phase it seems that every model has learnt its task but their output is not a binary mask yet. To achieve this, a thresholding operation has to be done to select a suitable value of p_{min}. Its tuning is done in a range of values from 0.05 to 0.95 and different models over the 300 images test set.

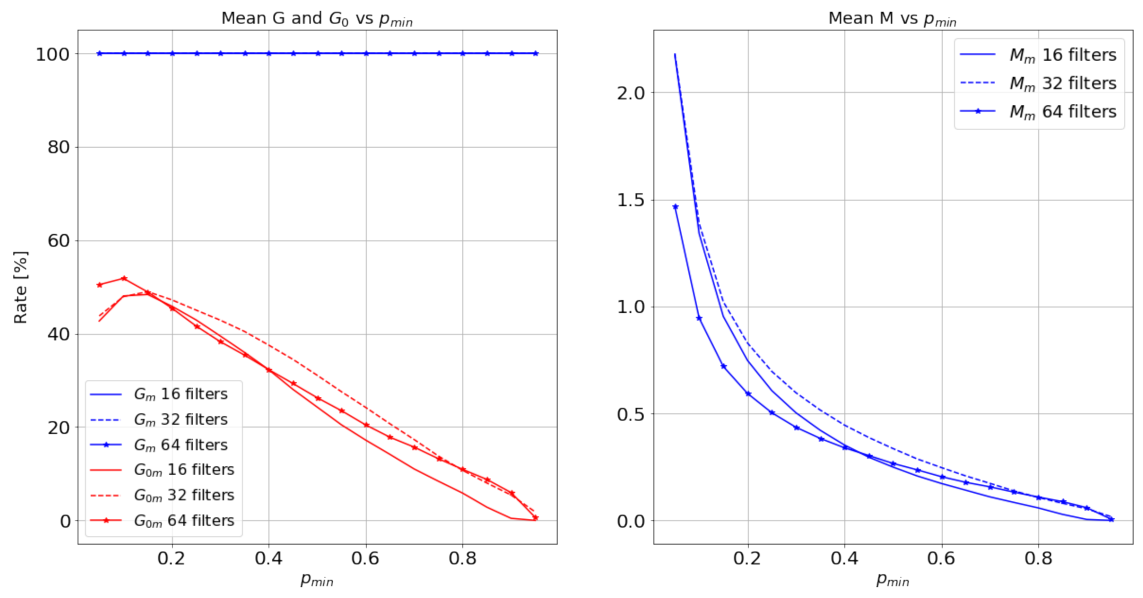

Figure 5. Performance indices curves vs p_{min} and models

For every p_{min} and model, the mean values of G, G_0 and M are computed and reported in Figure 5. It can be seen that the network with 64 filters performs better in terms of output mask. Values of G_0 greater than 50 % and M index near 1 mean that the result is closer to the reference mask. According to this result a model with 64 filters and $p_{min} = 0.1$ was chosen. It could seem that with a low value of threshold the output mask may present a significant amount of noise but this hypothesis is excluded by the behaviour of G close to 100% .

G_0 behaviour for low p_{min} values is due to a variation of number of useful matched signal ($cnt2$) and uncertain signal ($cnt1$):

- The greater is p_{min} and the less will be the over threshold pixels (low value of $cnt2$).
- The lower is p_{min} and the greater the over threshold pixels will be and thus the uncertainties (higher value of $cnt1$ against a saturation of useful matched signal pixels $cnt2$).

Tests with 128 and 256 initial filters models have been done but performances do not show relevant improvements with respect to the 64 initial filters model while the required storage memory and computational time increment is not negligible (and also undesired). This is the reason why they are not reported in Figure 5.

Masks Comparison with a Classical Image Segmentation Algorithm

Here a comparison between our proposed algorithm and a classical one is done using the same test set of 300 images. This will be done first using the $1024 \times 1024\ px$ images and then using the $512 \times 512\ px$ equivalent ones, because the classical algorithm is optimized to work with the 1024 px resolution while our algorithm has been designed to work with a smaller resolution due to workstation memory constraints. Thus for comparison on bigger image size a new processing strategy is used here: the $1024 \times 1024\ px$ is divided in four patches of $512 \times 512\ px$ and then the output masks are assembled together. This multi-patching strategy was analyzed for 64 filters model, tuning the p_{min} between 0.01 and 0.95 range to assess the behaviour of our algorithm before comparison with the classical algorithm.

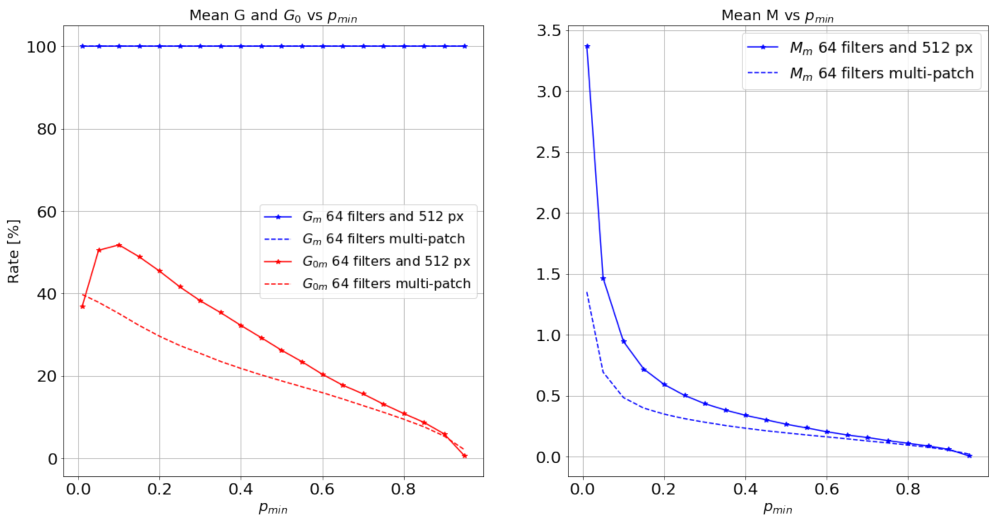

Figure 6. Performance indices curves vs p_{min} and processing strategy

Performances behaviour on p_{min} variation for the previous 64 filters , $512 \times 512\ px$ image size model and the new multi-patching strategy is shown in Figure 6. It can be seen that the performances of the model with the new strategy are worse with respect to working directly with 512 px input images. This happens because there has been an increasing of over threshold pixels in the reference 1024 px mask and the $p_{min} = 0.01$ value is still not sufficient to match all the target over threshold pixels. This is confirmed by the decreasing monotonic trend of the dashed red curve in Figure 6. For the classical algorithm comparison test over 1024×1024 px images, the $p_{min} = 0.01$ is selected while for the 512×512 px images test, $p_{min} = 0.1$ is chosen to achieve the best performances with our algorithms.

Table 2. Pre-processing Configuration Parameters

Parameter	Value (DN)
τ_{pre}	27
BKG_0	2000

The nominal values of the threshold τ_{pre} and initial background noise for the classical algorithm are set according to the Table 2. Thus, no tuning is required for the classical algorithm because it was set with the nominal threshold values from an extensive test campaign to maximize its performances. With this choice, mean values for G, G_0 and M are reported, together with our algorithm ones, both in Table 3 for the 1024 test set and in Table 4 for the 512 test test.

Table 3. Performance Indices Comparison over the 1024 px images Test set

Mean Index Value	U-Net algorithm	Classical Algorithm
G_m (%)	99.97	99.98
G_{0m} (%)	39.76	35.46
M_m	1.35	0.22

From Table 3 it can be seen that the global performances in terms of G_m are similar thus there is not a significant amount of noise in both the algorithms' outputs. Moreover the G_{0m} obtained using the CNN is slightly greater than the value reached using the classical algorithm. This would not mean a sensible improvement if M_m was not considered. Indeed, this index is greater than the classical algorithm and this means that the CNN predicts more pixels than its competitor while better detecting the useful objects' signal using this multi-patching strategy (Figure 7).

Table 4. Performance Indices Comparison over the 512 px images Test set

Mean Index Value	U-Net algorithm	Classical Algorithm
G_m (%)	99.98	99.99
G_{0m} (%)	51.81	28.81
M_m	0.95	0.06

Table 4 instead shows that the CNN performs better image segmentation with respect to the classical algorithm. G_{0m} around 50 % and M_m close to one are index of a good global behavior and this had to be expected because of the previous design choices. The classical algorithm shows a degradation of performances due to the reduced image size and consequently reduction of over threshold pixels to be detected. Indeed the classical algorithm is not capable of well segmenting the image for different noise levels scenarios while the CNN training dataset has been designed to achieve good segmented images in different noise conditions and the previous tables confirm this expectation. Moreover, the p_{min} threshold is something different from the τ_{pre} and BKG_0 because fixing it means just to decide which over threshold and under threshold pixels are within the U-Net prediction which does not present a great variety of noise levels. This explains why the the U-Net has been included in the design approach, to cope with the different noise conditions and give as output a baseline image on which differentiate pixels.

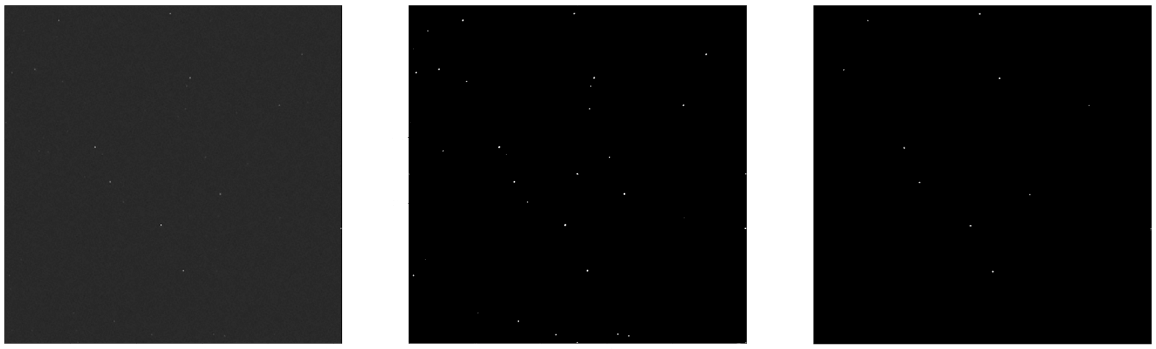

Figure 7. From left to right: Simulated image, CNN prediction and classical algorithm one.

Clustering Modules Comparison

A comparison of Clustering modules outputs is provided in this section. Ten 512 px images are provided to our algorithm and to the reference one, they are both artificially generated and acquired with a star tracker facility. Then differences in terms of clusters' centroids are evaluated and shown. Only the geometric information comparison is considered because of the future use of geometry based Lost In Space and Tracking algorithms. Tests results show that each algorithm has successfully identified the same 70 clusters on which the following distribution is built (Figure 8). The coordinates differences are obtained by subtraction of classical algorithm predictions to the U-Net based one.

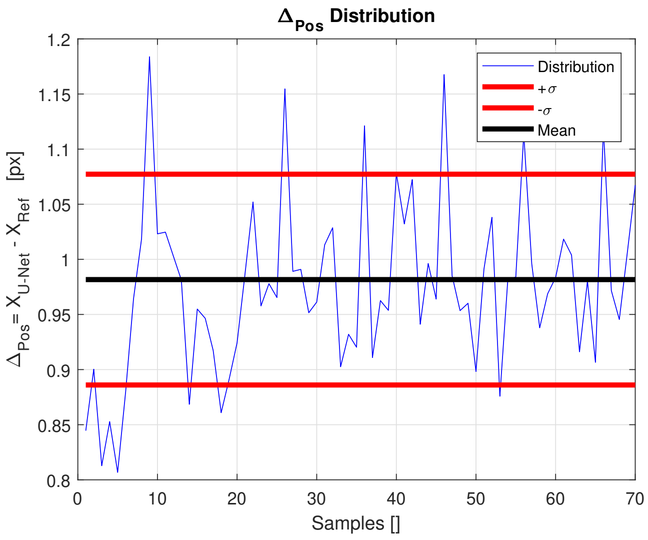

Figure 8. Distribution of clusters' centroids coordinates differences.

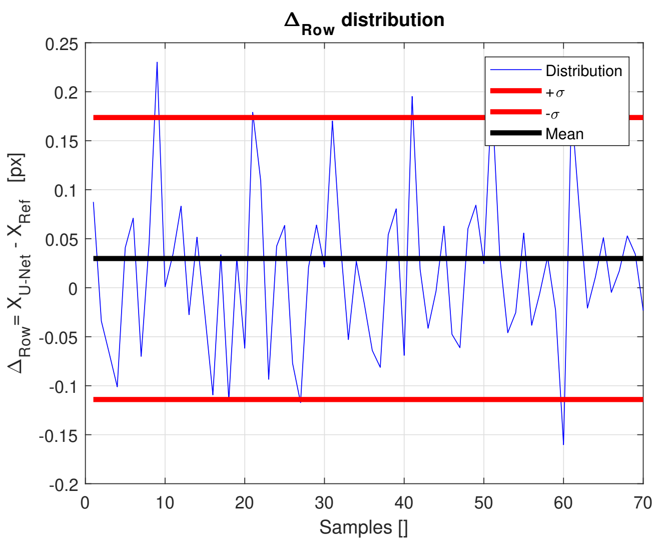

Figure 9. Distribution of clusters' centroids rows differences.

In Figure 8 all the differences are computed combining centroids' rows and columns differences with the Pythagorean Theorem. The distribution shows that the average values of this predictions' difference is around one pixel. This difference is evaluated just in terms of pixels and not in arcseconds because an angular value is strictly related to camera's features such as the Instantaneous Field Of View (IFOV). The distribution does not show relevant displacements from the average value due to its low Relative Standard Deviation (RSD) reported in Table 5.

Table 5. Distribution values for centroids' coordinates differences

Difference	Mean Value (px)	Standard Deviation (px)	RSD (%)
Rows	0.030	0.144	480
Columns	0.972	0.077	7.9
Position	0.981	0.096	9.7

Considering centroids' coordinates differences in terms of rows and columns (Figures 9 and 10), it can be noticed that rows differences distribution shows a not negligible standard deviation if compared to distribution's average value. On the contrary, columns' differences distribution shows a more stable behaviour with a low RSD and an average value around the unit. These distribution are reported to have an idea of the differences in terms of clusters' centroids estimates coming from the two algorithms. It must be pointed out that this estimates are obtained via the same weighted average process for both the algorithms. Thus differences in centroids coordinates are due to the different segmentation processes:

- In the AI based algorithm, segmentation process works in the same way both along row direction and column one. This is due to the capability of the U-Net in recognizing objects that stand out from the background without the implicit selection of a direction between row and column to achieve this target.
- In the reference algorithm the segmentation works along rows through the zig-zag process. Here, attention is paid to recognize over and under threshold pixels just looking at the energy

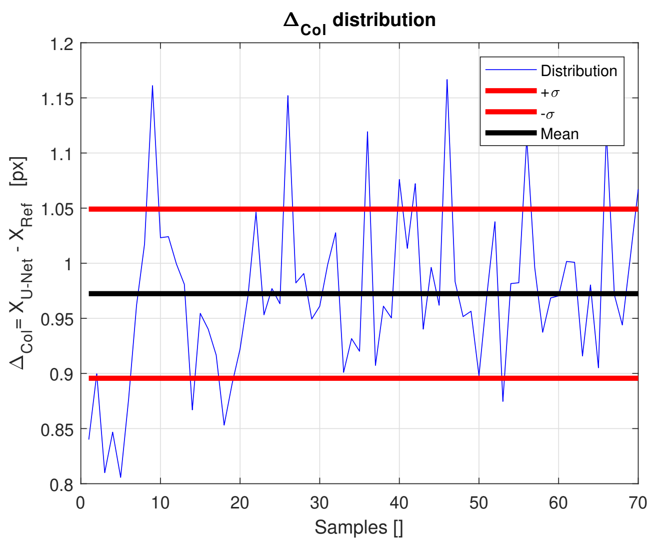

Figure 10. Distribution of clusters' centroids columns differences.

distribution along every row. This does not apply for columns.

The result of this different philosophy is that the generic cluster associated to the same object shows remarkable differences in terms of its horizontal segments that lead to the different distribution behaviours reported in the previous table and figures.

CONCLUSIONS

This work is still ongoing because other clustering solutions (based on AI) and further models for segmentation purposes have to be investigated. Moreover, a deeper analysis in multi-patch strategy with possible overlapping for larger images will be carried on. Provided results show several aspects:

- The increment of initial filters number increases the accuracy of the model predictions in terms of image segmentation quality and brightest objects detection.
- Three performance indices have been provided, discussed and used for design and comparison choices.
- Our proposed algorithm is capable of achieving satisfying segmentation performances against different signal-to-noise scenarios.
- The comparison shows that the classical algorithm performs generally worse than ours because it is not capable of detecting a significant amount of useful signal due to noise conditions variation.
- Multi-patch strategy seems not to perform as good as the global reduced size image processing strategy.
- Clustering modules comparison shows a difference of 1 pixel in terms of centroids estimate that is mainly due to centroid's column coordinate.

- Classical Segmentation algorithm is intrinsically biased due to its row oriented segmentation process while the U-Net does not favour any direction and this make it more reliable for estimating clusters' centroids.

In the end, a CNN dataset for night sky images segmentation has been realized and provided.

REFERENCES

[1] J. Jiang, L. Lei, and Z. Guangjun, "Robust and accurate star segmentation algorithm based on morphology," *Optical Engineering*, Vol. 55, No. 6, 2016, p. 063101.

[2] C. C. Liebe, "Accuracy performance of star trackers - a tutorial," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 38, No. 2, 2002, pp. 587–599, 10.1109/TAES.2002.1008988.

[3] D. Rijlaarsdam, H. Yous, J. Byrne, D. Oddenino, G. Furano, and D. Moloney, "Efficient Star Identification Using a Neural Network," *Sensors*, Vol. 20, No. 13, 2020, p. 3684.

[4] L. Xu, J. Jiang, and L. Liu, "RPNet: A Representation Learning-Based Star Identification Algorithm," *IEEE Access*, Vol. 7, 2019, pp. 92193–92202, 10.1109/ACCESS.2019.2927684.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, Vol. 25, 2012, pp. 1097–1105.

[9] C. Cortes, M. Mohri, and A. Rostamizadeh, "L2 Regularization for Learning Kernels," *CoRR*, Vol. abs/1205.2653, 2012.

[10] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," *arXiv preprint arXiv:1805.11604*, 2018.

[11] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," *2018 international interdisciplinary PhD workshop (IIPhDW)*, IEEE, 2018, pp. 117–122.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, Vol. 15, No. 1, 2014, pp. 1929–1958.

[13] A. Silburt, M. Ali-Dib, C. Zhu, A. Jackson, D. Valencia, Y. Kissin, D. Tamayo, and K. Menou, "Lunar crater identification via deep learning," *Icarus*, Vol. 317, 2019, pp. 27–38.

[14] H. Iqbal, "HarisIqbal88/PlotNeuralNet v1.0.0," Dec. 2018, 10.5281/zenodo.2526396.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[16] G. Zotti and A. Wolf, "Stellarium 0.19. 0 User Guide," tech. rep., Technical report. Available online at github. com/Stellarium/stellarium . . . , 2019.

[17] F. Curti, D. Spiller, L. Ansalone, S. Becucci, D. Procopio, F. Boldrini, P. Fidanzati, and G. Sechi, "High angular rate determination algorithm based on star sensing," *Advances in the Astronautical Sciences Guidance, Navigation and Control 2015, Vol. 154*, p. 12, 2015.

[18] V. Schiattarella, D. Spiller, and F. Curti, "Star identification robust to angular rates and false objects with rolling shutter compensation," *Acta Astronautica*, Vol. 166, 2020, pp. 243–259.

[19] V. Schiattarella, D. Spiller, and F. Curti, "Efficient star identification algorithm for nanosatellites in harsh environment," *Advances in the Astronautical Sciences, Vol. 163*, pp. 287–306, 2018.

[20] V. Schiattarella, D. Spiller, and F. Curti, "A novel star identification technique robust to high presence of false objects: The Multi-Poles Algorithm," *Advances in Space Research*, Vol. 59, No. 8, 2017, pp. 2133–2147.

[21] L. Kazemi, J. Enright, and T. Dzamba, "Improving star tracker centroiding performance in dynamic imaging conditions," *2015 IEEE Aerospace Conference*, IEEE, 2015, pp. 1–8.

[22] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, Vol. 9, No. 1, 1979, pp. 62–66.

[23] R. C. Gonzalez, R. E. Woods, *et al.*, "Digital image processing," 2002.

[24] F. Chollet, *Deep Learning with Python*. Manning, Nov. 2017.