Instance Segmentation for Feature Recognition on non-cooperative Resident Space Objects

Niccolò Faraco [*], Michele Maestrini [†], and Pierluigi Di Lizia [‡]
Politecnico di Milano, Department of Aerospace Science and Technology, Via La Masa 34, 20156 Milan, Italy

Active debris removal and unmanned on-orbit servicing missions have gained interest in the last few years, along with the possibility to perform them through the use of an autonomous chasing spacecraft. In this work, new resources are proposed to aid the implementation of guidance, navigation and control algorithms for satellites devoted to the inspection of non-cooperative targets before any proximity operation is initiated. In particular, the use of Convolutional Neural Networks (CNNs) performing object detection and instance segmentation is proposed and its effectiveness in recognizing components and parts of the target satellite is evaluated. Yet no reliable training images dataset of this kind exists to date. A tailored and publicly available software has been developed to overcome this limitation by generating synthetic images. Computer Aided Design models of existing satellites are loaded on a 3-D animation software and used to programmatically render images of the objects from different point of views and in different lighting conditions, together with the necessary ground truth labels and masks for each image. The results show how a relatively low number of iterations is sufficient for a CNN trained on such datasets to reach a mean average precision value in line with state-of-the-art-performances achieved by CNNs in common datasets. An assessment of the performance of the neural network when trained on different conditions is provided. To conclude, the method is tested on real images from the MEV-1 on-orbit servicing mission, showing that using only artificially generated images to train the model does not compromise the learning process.

I. Introduction

The incipient overcrowding of the most exploited orbits due to the presence of man-made Resident Space Objects (RSO) is a non-trivial challenge for the design of new space missions. Among the various practices to mitigate the problem, the adoption of end-of-life measures, such as the injection into graveyard orbits or a controlled atmospheric reentry, are recommended internationally when the RSO preserves some control capabilities [1]. For non-cooperative

[*]Ph.D. student, corresponding author, niccolo.faraco@polimi.it
[†]Research fellow, michele.maestrini@polimi.it
[‡]Assistant Professor, AIAA Member, pierluigi.dilizia@polimi.it

objects, besides their continuous and accurate tracking with advanced ground-based sensors [2], the implementation of active removal missions appears to be a crucial service to be developed in the upcoming future [3].
Active debris removal (ADR) has increasingly gained the attention of the community and has been drawing big efforts since it is the only long-term solution to the problem when atmospheric reentry disposal is not feasible. However, a very precise knowledge of the conditions of the satellite to be removed is needed to enable ADR. To this aim, missions specifically targeted to the inspection of the objective spacecraft come into play, such as the e.Inspector mission currently developed by ESA [4] or NASA's OSAM-1 mission [5]. The inspection is carried out through a secondary satellite, the *chaser*, whose orbit is defined based on the *target* RSO. Within a standard approach to operations, the satellite would rely on a constant flow of information, back and forth from the spacecraft to the ground, in order to determine the current dynamics of the target body. Usually, the data transmitted from the space segment is elaborated on ground, and relevant commands are transmitted back on orbit to be executed. This approach is well suited for typical monitoring trajectories around known objects, e.g. the ones exploiting the concept of safety ellipses [6], which can be designed to ensure the possibility to realize a wide range of different observation conditions, especially if the target is tumbling. However, it has been demonstrated that such investigation techniques could suffer from considerable shortcomings when dealing with unknown objects [7], whose tumbling attitude can't be assessed in advance. Indeed, should the target be controlled (i.e., not tumbling) this would impair the visibility of certain regions of its surface if the relative trajectory is kept constant. In this context, classical mission control routines may suffer from the need to continuously adapt the relative orbits to ensure a complete inspection of the RSO. This increases, in turn, the effort spent by mission analysts and limits the applicability of standard approaches.

Promoting autonomy in satellite operations may be a solution to the above issues. Since the main objective of inspection missions is the visual probing of the target satellite, the chaser is generally equipped with one or more cameras that are used to shoot pictures that are downlinked for subsequent analysis. These images could directly be exploited to untie the Guidance, Navigation, and Control (GNC) tasks from the ground segment: in fact, they could be fed to state-of-the-art image processing algorithms running on the on-board computer to detect the prominent components of the target satellite. Among the many possible techniques, the recent advances in machine learning approaches have been shown to grant adequate accuracy and applicability on low-resource systems, such as the ones typically adopted for space applications [8]. This kind of computer vision processes provides also a measure of the reliability of the results, which can be used to suggest the portions of the target to be further examined. This information can serve another piece of the GNC pipeline, namely an autonomous guidance algorithm (e.g., see [9]), to define a new flight profile for the chaser around the target to get more beneficial views on the most uncertain components of the inspected RSO. By performing the suggested course adjustments iteratively, full coverage of the target spacecraft could be obtained with a prescribed accuracy. Moreover, if properly tailored, also autonomous navigation could be achieved with a similar approach, as demonstrated in [8].

At this point, it is clear that the implementation of recognition algorithms and the assessment of their performances is of uttermost importance to enable the use of autonomous guidance and navigation algorithms for future inspection and proximity missions [10] and the thorough assessment of the target spacecraft configuration and degradation conditions. While many different solutions have already been proposed to perform pose estimation of the RSO, such as [11–13], little effort has been done, to date, to validate new techniques for the visual inspection of unknown target objects. In this work, a specific declination of Region-based Convolutional Neural Networks (R-CNN) has been chosen as the tool to tackle the problem [14]. These algorithms, just like any other image recognition algorithm, have to be trained on sets of pictures for which the category of the objects depicted and their position in the image are provided. Among its key innovations, this work introduces an approach to obtain training data for such neural networks. Up to date, in fact, something similar has only been done by training neural networks for the specific task of recognizing solar panels. The dataset for this application consisted of manually annotated images taken from the internet [15]. This approach has several disadvantages. The first one deals with the scalability of the method, which is a relevant drawback since it is well known that Neural Networks performances dramatically improve when trained on larger datasets. In fact, the images have to be manually labeled and there is no way to automate the process, which makes it labor-intensive for a human operator to provide a large enough dataset. Secondly, it is not possible to tailor the recognition for a specific target, should a mission aim to inspect a specific and *a priori* known RSO. Finally, this approach is strongly limiting in the kind of identified components (i.e. only solar panels), which makes the range of applicability of the approach narrower and less interesting. To overcome these limitations, the use of 3-D Computer Aided Design (CAD) models of various satellites is explored in this work to programmatically generate images with different points of view and lighting conditions. This has been achieved through the open-source computer graphic software Blender®[*], accurately scripted in order to generate, together with the images, the necessary information on the objects depicted and their position in the picture, which hereafter will be referred to as *ground truth*. To the best of our knowledge, no other annotated dataset providing such variability of models and components of RSOs has been made publicly available to date to aid research efforts on the tasks of object detection and instance segmentation in the aerospace field (while well-known alternatives exist for pose estimation [11, 16]). This approach proved to be an effective method for the training of the neural networks, also solving the problems outlined for the approaches previously used in the literature. Moreover, it granted adequate results not only on simulated images, but also on images from a real mission, namely the Intelsat 901 operative life extension mission[†][‡].

The next section illustrates the reasoning behind the choice of the kind of NN to be employed as well as the specific implementation used. The working principles of the training and testing datasets generation tool are then highlighted in

[*]`https://www.blender.org/`

[†]Company webpage: `https://www.northropgrumman.com/space/space-logistics-services/`

[‡]Press release: `https://news.northropgrumman.com/news/releases/intelsat-901-satellite-returns-to-service-using-northrop-grummans-mission-extension-vehicle`

Section III and the results obtained on an extensive testing campaign are discussed in Section IV.

II. Convolutional Neural Networks for image recognition

In the last few decades, Machine Learning has gained increasing popularity due to the exponential improvement of the computing capabilities of microprocessors and, above-all, to the exploitation of graphic cards architecture [17]. Among the various approaches used in Machine Learning (ML), Artificial Neural Networks (ANNs) and, in particular, Convolutional Neural Networks (CNN) have proved to be very effective in the field of computer vision, which is the ability of the machine to gain high-level understanding of their surrounding environment from digital images. Among the various kinds of tasks that such algorithms can perform, the most interesting ones for the problem at hand are object detection and instance segmentation. Object detection refers to the capacity to identify various objects in each image, drawing for each of them a bounding box. Instance segmentation instead, rather than finding the envelope of each object as in the previous case, focuses on telling exactly which pixels belong to the object.

The approaches applied in the dedicated literature belong to two main categories: dual stage approaches like Faster R-CNN [14, 18–20] and single stage approaches [21, 22]. During the first step of a typical dual stage algorithm the image undergoes a first level of processing where an *object proposal algorithm* extracts patches from the original image which may contain an object. These object proposal techniques can be standard image processing algorithms [14], as well as other layers of a CNN [18]. In the second step, the proposed patches are fed to a classical CNN which assigns them a category. On the other hand, single stage approaches only apply a single CNN to the full image directly. The CNN divides the input into regions and predicts bounding boxes and probabilities for each region. It is clear that the underlying architectures cause the dual stage approaches to be slower then single stage ones, while retaining higher accuracy. In this work, the problem specific situation of satellite inspection is considered, where natural observation trajectories in LEO would last several tens of minutes [9]. Therefore, we select Mask RCNN [20], which is the state-of-the-art algorithm for semantic segmentation using a dual stage approach. This approach adds a mask prediction branch to Faster RCNN [18] with almost no additional computational time. While the use of segmentation maps for navigation about known small irregular bodies has been recently investigated [23], this technology has never been applied before for the inspection of unknown RSO, but could prove useful in different ways, since it provides useful information not only on the generic bounding box containing a certain part of the satellite in the image, but also on where the material is located inside that box. This information may turn out to be paramount for the determination of attitude, geometrical properties, and mass distribution of the observed RSO, should it be in an off-nominal configuration. Think, for example, of promising inspection techniques as the one explained in [7] or other based on the concept of safety ellipses. Since the center of mass of the object is assumed to coincide with the center of the ellipsoid, these techniques would fail in the case of highly asymmetric or elongated objects. Knowing the typical materials which constitute the elements of a spacecraft, the present work could provide a more reliable estimation of the mass distribution of the object and the related properties. Due to these reasons, the present work is thought as an instrument to provide additional information to make other strategies more robust, rather than offering a self-contained alternative solution by itself.

III. JINS: a synthetic images generator for the training of machine learning algorithms

The idea to overcome the issue of providing accurate automated labeling to images relies on the use of Python™ programming language and the scripting capabilities of the 3-D graphic software Blender®. The software that has been implemented to this purpose is called JINS, which stands for JINS Is Not a Simulator. JINS's pipeline relies on three main steps: a manual pre-processing phase, the generation of the images, and the generation of the annotation file containing the ground truth.

A. Model pre-processing

First, the provided spacecraft CAD model is adjusted to comply with the code requisites. This step is a one-time-only procedure, which must be applied for every new model that the user desires to add to the dataset. Blender gives the user the ability to load several of the most used output formats from CAD programs, which is an advantage since this kind of files are always produced during the design process of the satellite and could therefore be used for the study of a tailored on-orbit servicing or disposal mission.

The necessary preprocessing of the model starts with its subdivision into parts corresponding to the components of interest. Each occurrence of these components included in the list of classes to be identified must be interpreted by the software as a distinct standalone object. Then, a custom property has to be specified for each of the parts of interest: this has to be identified by the 'label' keyword and its value has to be set to a string corresponding to the class.

Secondly, an *empty entity*[§] for the camera to point to must be provided. In this case, a simple set of Cartesian axes has been used. The origin of these axes is positioned in a random point close to the satellite but not necessarily coincident with any notable physical or geometrical point (e.g. the center of mass). The aim of this entity is to supply an off-nominal target for the camera, with the aim of introducing a small offset from the center of the image (e.g. to simulate a non-ideal pointing of the RSO). Indeed, this is a desirable feature since it adds diversity to the dataset and forces the algorithm to learn to recognize the objects independently of their position in the image.

B. Image generation

Once the model has been adapted, it is fed to the part of the JINS software that is responsible for the images generation. In short, it generates a user-defined number of light objects in the scene and activates them one at a time. For each of these light sources, it generates a number of cameras and renders the image from each of these points of view. At this step, also the ground truth pixel masks are generated for each object of each category. A conceptual

[§]In Blender, an Empty is an object which is not rendered, i.e. an object that can be used for modeling purposes but that does not appear in the image when it is generated.

scheme of the procedure is provided in Fig. 1 and a sample of the results is provided in Fig. 2. As already said, variety is a key feature for the dataset to have good performances, therefore some measures have been taken to ensure that the obtained dataset is sufficiently representative of real conditions.

1. Light conditions

A user specified number of light sources are added to the scene. These mimics the lighting from the Sun, which illuminates the scene with rays that are all parallel to a specified direction. The script generates a random unit 1×3 vector, spawns a Sun lighting object in the scene and then applies a quaternion rotation so that its direction matches the one of the vector just created.

2. Camera generation

For each of the generated light sources, a number of cameras is randomly generated in a solid angle with maximum aperture of $35°$, centered around the lighting direction and pointed towards the offset target empty object. Thanks to this random sampling, each light source will have its own random and unique set of cameras. This approach ensures that the lighting conditions in the image are always reasonable. In fact, in a real application, there would be no reason to take pictures of the target satellite when it is not properly lit. Since the used 3D spacecraft models are not characterized by a uniform scale, the camera focal length is kept constant and the range of distances from the target in which the camera can be generated is tailored to each model in order to guarantee that the satellite is clearly visible in the image, without being too small for its parts to be distinguishable or so large that one of its components completely fills the camera field of view.

3. Mask generation

Exploiting Blender® *compositor nodes*, binary masks like the ones shown in Fig. 2 can be obtained at almost no additional computational effort at the same time that the full image is rendered. These masks specify the area in the picture covered by each of the parts of the satellite and are used to encode the ground truth for the dataset.

Earth background The JINS software is also capable of including the Earth in the background of the image (if needed), in order to increase the complexity, variety and verisimilitude of the training and testing datasets. Due to limitations of the available computational resources, the planet model could not be directly added to the scene. Therefore a workaround inspired by [8, 11] was implemented. First, a series of pictures of the Earth in different light conditions and from different points of view are rendered. Secondly, JINS randomly selects one of these pictures and uses it as a background when rendering the model of the spacecraft.

This is obviously a non-optimal solution as it could lead to images that are not physically accurate, but solving this issue is considered to be beyond the purpose of this study.

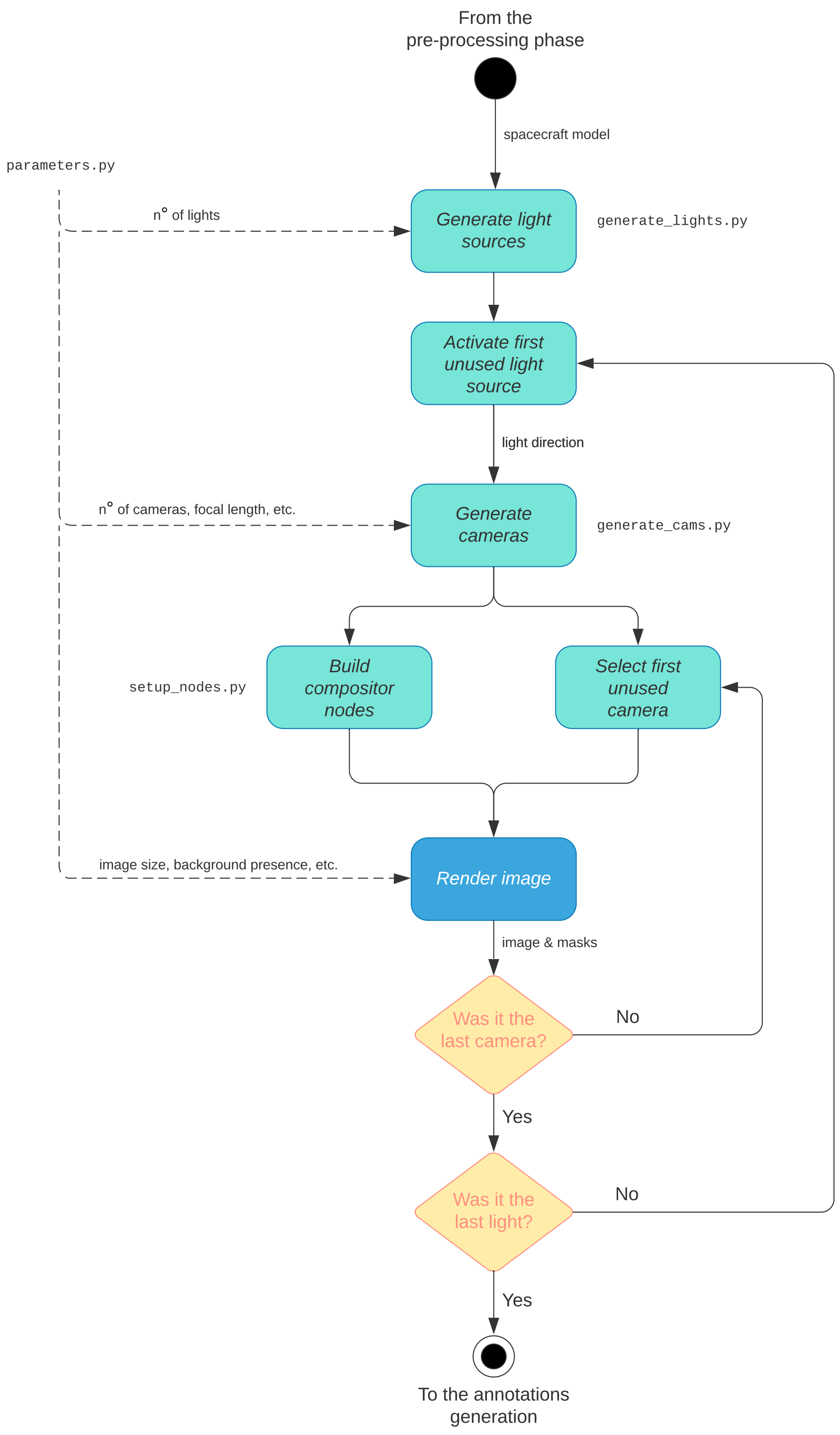


Fig. 1 Scheme of the image generation pipeline in JINS.

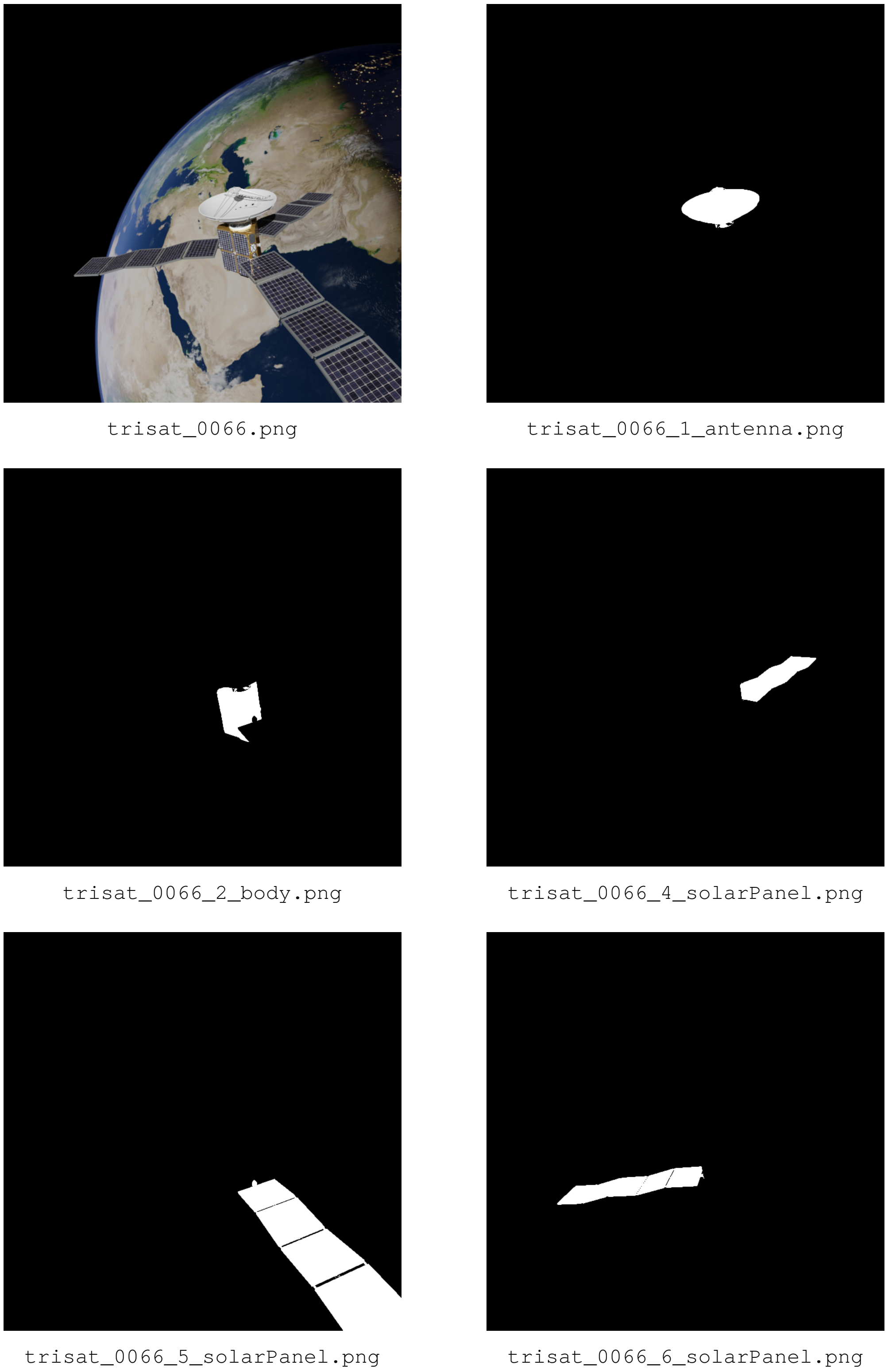

trisat_0066.png
trisat_0066_1_antenna.png
trisat_0066_2_body.png
trisat_0066_4_solarPanel.png
trisat_0066_5_solarPanel.png
trisat_0066_6_solarPanel.png

Fig. 2 Example of a rendered image and the related masks identifying the different components. The object's class is specified in the filename under the images.

C. Annotation file generation

The network does not look at the mask images directly, but rather reads the information from the annotated file. Such information is provided via a `.json` file and it is the only additional information that must be provided to the neural network together with the training pictures. Therefore, another script takes care of the encoding of the such `.json` file stating the class and the polygon that encloses each object. The selected data format for the database is the one of COCO[¶] (Common Objects in COntext), which is one of the most common benchmarking datasets for new algorithms in the field of computer vision and has, therefore, become one of the standards for image annotation. The choice of this particular data format is owed to the desire to make the JINS dataset and code publicly available for other users.
The annotating procedure starts by identifying, based on the name of the image, all of the masks that are related to it. Then, using the API from COCO[‖], it is able to identify the white polygon in each mask and, from that, to build the bounding box. The class of the object is inferred from the filename of the mask image and it is determined by the label assigned to the object in the model pre-processing phase. Through the API other properties can be inferred too, such as the measure of the area of the mask, which is useful in order to neglect objects that are too small in the image.

IV. Results

Among the various available implementations of the image recognition algorithm, `Detectron2` [24] has been selected for this work as it is the latest iteration of the code from the original authors of the Mask R-CNN paper [20], which has been developed by the Facebook AI Research (FAIR) group. Among the different base models provided by Detectron2, the one that was chosen is the *R50-FPN-3x*, which uses a ResNet [25] backbone architecture with 50 layers and a Feature Pyramid Network (FPN) [26] for the regions proposal. This particular architecture has been selected because, despite being one of the least resources intensive options both in terms of training time and memory usage, it still performs adequately well [27].

The number of classes the neural network was trained to recognize is four: antenna, main body, solar panels, and engines. The 'thruster' class was initially taken into account too, however its instances were too small in the generated dataset even for human recognition. Hence, they were always rejected when inferring the ground truth from the masks. As a consequence, closer images would be needed to recognize them.

Six satellites' models[**] have been used to generate the images of the RSOs to be recognized. Figure 3 shows them, together with the names that will be used hereafter to refer to each of them. This is useful since some of the results find

[¶]http://cocodataset.org/#home (last accessed: 11 April 2021)

[‖]https://github.com/cocodataset/cocoapi (last accessed: 11 April 2021)

[**]The resources used in the current work can be found at the following links (last accessed: 11 April 2021):
https://free3d.com/3d-model/small-satellite-308237.html
https://sketchfab.com/3d-models/satellite-f8bd72434281441db77e85ce830f89c1
https://nasa3d.arc.nasa.gov/models
and https://www.cgtrader.com/3d-models/space/spaceship/low-poly-satellite-parts

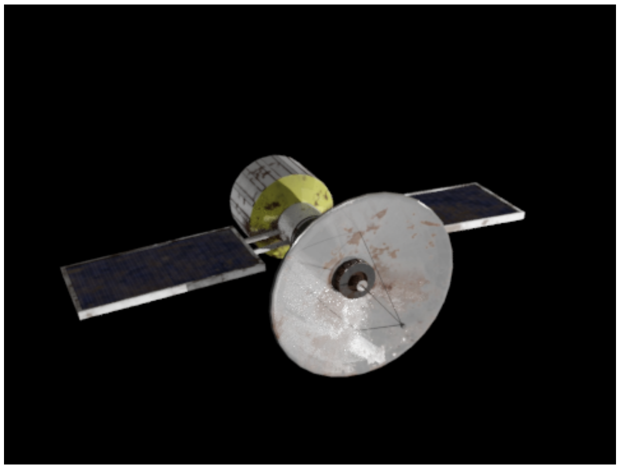
their reason to be in the geometry of the spacecraft itself.

Two different couples of training and testing datasets were produced for each satellite model: one with the Earth in background and one without it. In either case, the number of images is 800 for the training set (80 different lighting conditions and 10 different rendering points of view for each of them) and 200 for the testing set (20 lighting directions and 10 cameras each). As explained in Section III.B.2, the sets of camera viewing directions are randomly selected and, as a consequence, each set differs for each lightning condition. Being the number of classes quite small and the base NN pre-trained on the COCO dataset, the number of images used is enough to provide acceptable results, as shown hereinafter. Good results have actually been achieved in the literature with even smaller datasets using Faster R-CNN [28], but leveraging an higher number of images was preferred both because of the novelty of the task to be performed and the used datasets being fully synthetic.

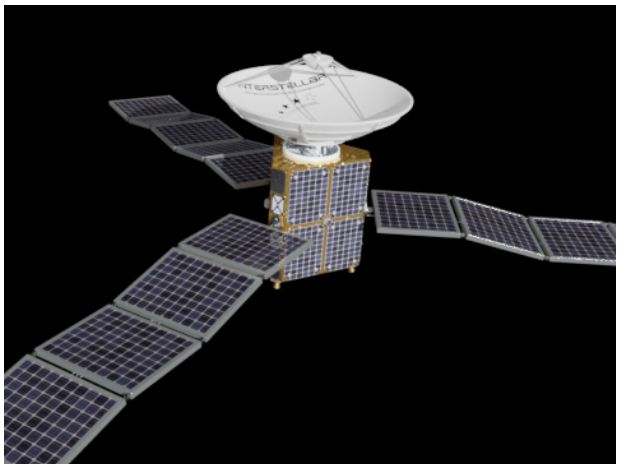
Square images with a 700 px size have been used to build the datasets used for this work. It is common practice in the field of computer vision to include real images in the dataset along with the synthetic ones, as it is already done for pose estimation datasets in aerospace applications such as [11, 16, 28], in order to increase the robustness of the training process. Nonetheless, contrarily to pose estimation data, generating segmentation maps automatically on real data is something that requires much more study and effort. Since, at the moment, it is only possible to annotate segmentation maps of real images by hand, using the small number of available real images as a test dataset was preferred (see Sec. IV.D.3), as it was deemed more useful and insightful than exploiting them in the training dataset.

For the current study, each CNN model has been trained for $2,500$ iterations with a learning rate of 0.0025, which showed to be a good compromise between training time and performances. The inference runs at around 10 FPS for each of the evaluated datasets when using GPUs computing power on Google Colaboratory servers (the actual hardware architecture changes). The inference process is much slower when run on CPU rather then GPU, requiring an average of 4-5 seconds per image (0.2-0.3 FPS). Nonetheless, given the relatively long duration of the observation legs, this is deemed sufficient for a possible on-board implementation. In fact, even when not supported by proper hardware, the method could still be employed to gather additional information on the geometry, configuration, and conditions of the unknown target, e.g. during the observation phases scheduled in the mission paradigm envisioned by Maestrini et al. [7].

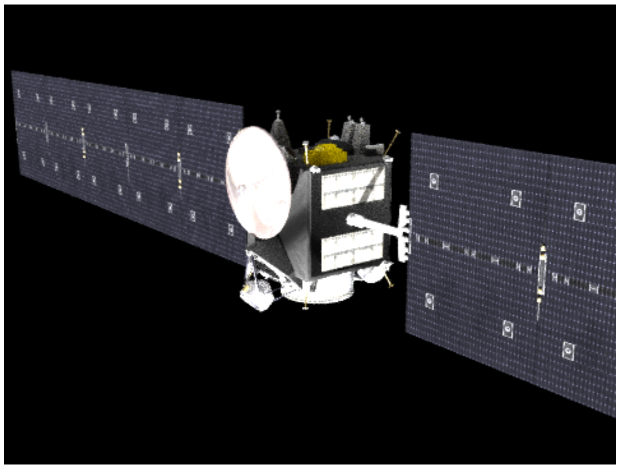
When post-processing the large amount of test data it seemed reasonable to search for a single metric, among the ones usually used in the machine learning field, capable of summarizing the overall performance of the network and the validity of the proposed method. Since the Average Precision (AP) and the mean Average Precision (mAP) also convey information concerning the true and false positives identified instances, they were deemed sufficient to capture the overall performance of the network on the newly proposed task. To understand them let us introduce some other



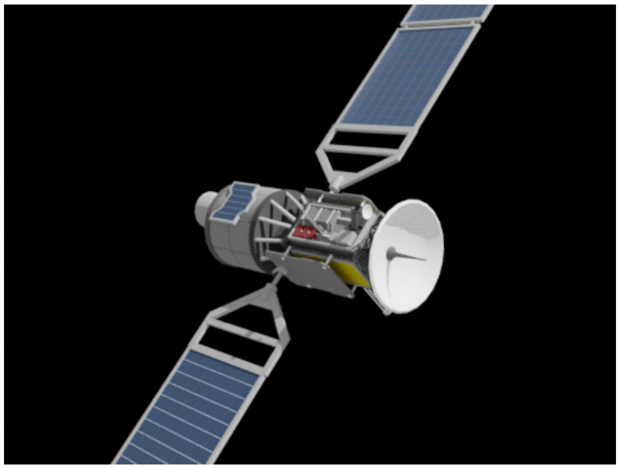
model A



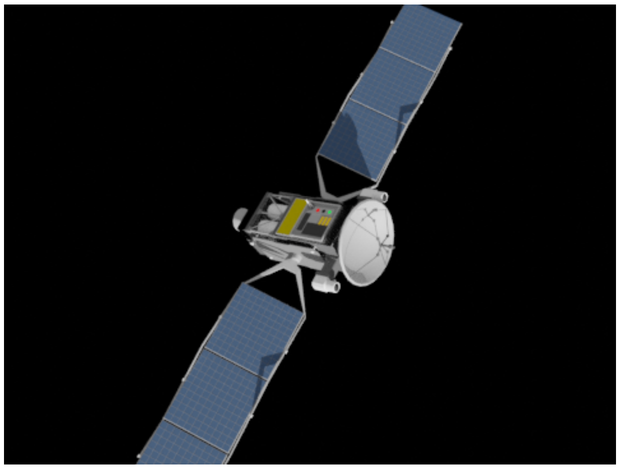
model B



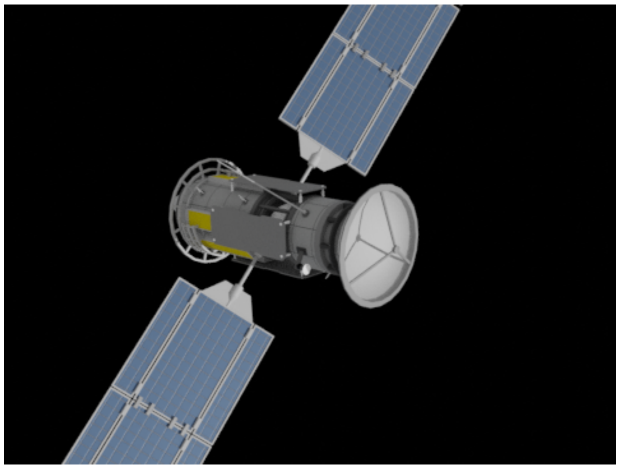
model C



model D



model E



model F

Fig. 3 Spacecraft models used in the present work. The names that will be used to refer to them are here reported under each image.

definitions first [29]:

$$\begin{aligned} Precision: &\quad p = \frac{TP}{TP+FP} \\ Recall: &\quad r = \frac{TP}{TP+FN} \end{aligned} \tag{1}$$

where TP stands for *true positives* and refers to the number of objects correctly identified, while FP are the *false positives*, i.e. instances identified as something they are not, and FN are the *false negatives*, which is the number of objects which are not identified. True/false positives and negatives are identified based on the IoU metric, which is the ratio between the area of the mask as identified during inference and the one reported in the ground truth and the union of the two masks. These quantities can be computed for each class of objects and lead to the definition of

$$Average\ Precision: \quad AP = \int_0^1 p(r)\, dr \tag{2}$$

as the area under the precision-recall curve for each class. Such curve is usually computed by the 11-point interpolation or similar techniques, so that the previous formula could, for example, assume the form

$$AP = \frac{1}{11} \sum_{r_i} p(r_i) \tag{3}$$

where r_i is the *i-th* recall sample in the interval $[0, 1]$.
The mAP is then simply the average of the AP over the classes and it is therefore more robust to small defects in the datasets, while providing a comprehensive and overall index on the quality of the detection. Although simplified, this explanation shall suffice for the aim of this work, for additional details see [29].

Hereafter, the most significant results when performing instance segmentation with a true positive recognition threshold of 0.7 on the described datasets are reported. The choice of this threshold value is justified by the fact that it is commonly used in pre-trained network architectures and that the tool showed low sensitivity against its variation. `Detectron2`, being based on Faster R-CNN, is also capable of performing the object detection task: since the results show the same trends highlighted by the segmentation task, they will not be discussed in details, but are reported in Appendix A for completeness.

A. Results on single model

As a first try, the algorithm was trained over a chosen spacecraft model and tested on different images generated from the same one. While this may seem to be a trivial and useless test to be performed, it actually provides two important insights: firstly, it validates the viability of the proposed approach on the simplest scenario; secondly, it may still represent the realistic scenario of an inspection mission towards a specific target whose complete model is known a priori. The results are shown in Tab. 1 and are in line with the ones in the literature [20]. Analogous results have been obtained by performing the same test on the other spacecraft models, which will not be reported here for the sake of brevity. Notice, in particular, how the highest score, even if by a small margin in this case, is obtained on the recognition of the solar panels: this trend is confirmed also in the tests illustrated in the next sections, where the scoring on the solar panel category is almost always the highest and the more robust to the increasing complexity of the datasets. The reason for such behavior is to be found in the fact that the solar panels are a pretty standardized objects in the industry, featuring very distinctive elements that undergo only minor changes from model to model. This obviously makes it easier for the algorithm to recognize them, since it's based on the identification of patterns that are common to most of the objects belonging to a certain class.

Figure 4 illustrates the results obtained on two images: the masks overlap, up to visual accuracy, to the components to be detected.

Table 1 Results on a dataset including a single model (model A).

	AP-antenna	AP-body	AP-solarPanel	mAP
mask prediction	88.53	89.10	90.22	86.39

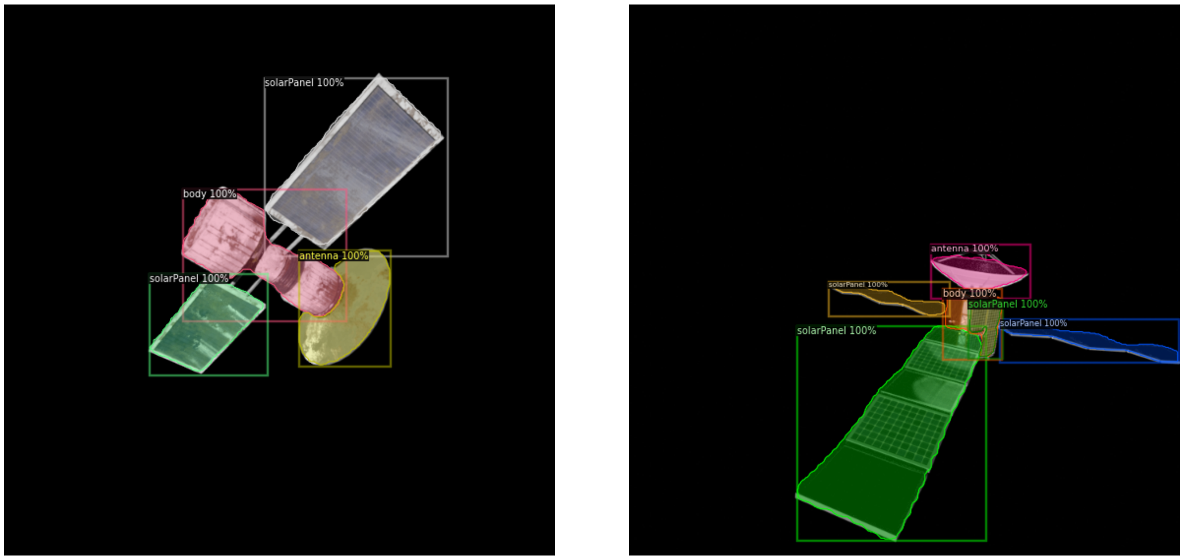


Fig. 4 Results from the inference on the single satellite datasets.

B. Testing on unknown model

It is worth studying the performance of the CNN model when tested on a satellite whose images were not included in the training set. To do so, the algorithm has been trained on two different datasets: the first one counting images based on spacecrafts' models B and C and the second one including also images of models D, E, and F (see Fig. 3). Both have been tested on the same set, which includes model A images only. The differences in the results are, therefore,

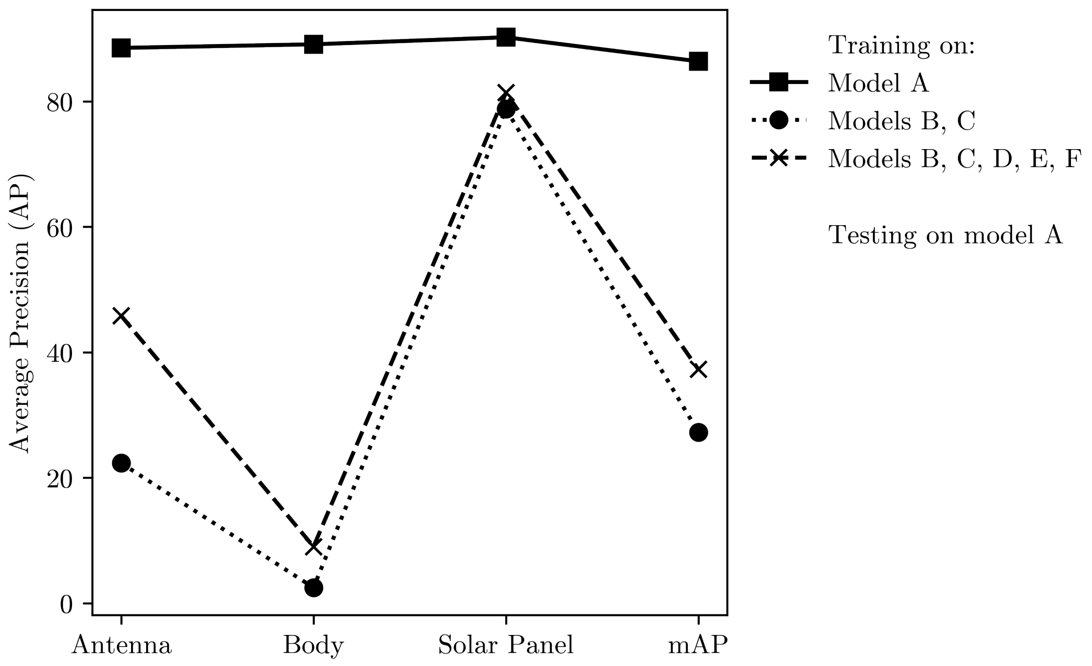


Fig. 5 Results when running inference on a spacecraft model not included in the training dataset.

to be attributed to the variety of the training sets.

Looking at Fig. 5, which reports the results of the test in Sec. IV.A for comparison (continuous line), it is evident that the accuracy of the model drops off sensibly when compared to the case case in which the CNN is trained and tested on the same spacecraft model. The number and variety of the images of the two satellites chosen for the training dataset are evidently not sufficient to grant a good accuracy in the testing phase. For example, the cylindrical body is easily mistaken for the antenna, as highlighted in Fig. 6.

Nonetheless, increasing the number and variety of the spacecraft models included in the training set (dashed line with respect to dotted one in the picture) yields a significant improvement of the results. This is particularly true for those components whose characteristics may differ significantly depending on the satellite (e.g., Body), whereas the segmentation tends to be more robust for components that show similar characteristics (e.g., Solar Panels). In fact, here we can notice a good example of the peculiarity explained in the previous section.

C. Earth background influence

Up to now, the performance of the algorithm on images depicting the RSO on a black background (representing the deep space) has been analyzed. However, how the presence of the Earth in the background of the image can affect the results has been investigated too.

To do so, two training sets have been built, both counting the same five satellites models (A, B, C, E, and F). One of the two, however, uses random images of the planet as the background (see paragraph III.B.3) rather than having a simple

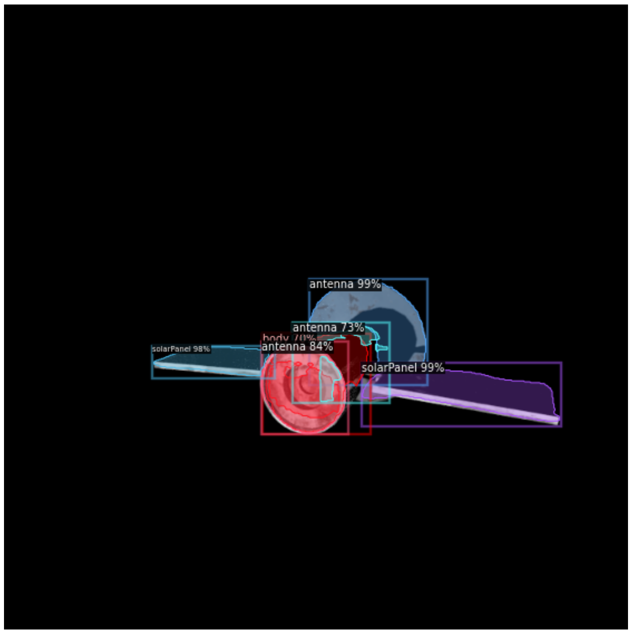


Fig. 6 Inference on a satellite model not present in the training dataset.

black backdrop. Two testing datasets, using the spacecraft model D left out of the training sets, have been generated accordingly.

Figure 7 shows how the algorithm behaves when dealing with different combinations of training and testing sets. The results when training and testing without the planet in background (solid line) are reported as a baseline and we can see how the reduction in performance when training and testing on an analogous dataset featuring the earth in background (dotted line) is minimal.

The same, as expected, is not true when training on a dataset without the Earth in background and testing with the planet in the images (dashed line). This is due to the fact that the clutter resulting from the planet in background makes the borders between the objects less separable. In fact, in Fig. 8 it is possible to observe how some of the masks overlap with other objects.

As a final remark, it is worth specifying that the worse results obtained on the engine class with respect to the other categories is not due to the influence of the planet's presence in the images but to the small number of engine instances occurring in the pictures, making it difficult for the CNN to be properly trained on recognizing this kind of objects. This is demonstrated by the baseline results, which show the same trend.

D. Real life scenarios

Finally, the approach has been tested in a series of conditions which may represent real operative situations, namely:

1) A simulated relative orbit around the target
2) Images in gray-scale encoding

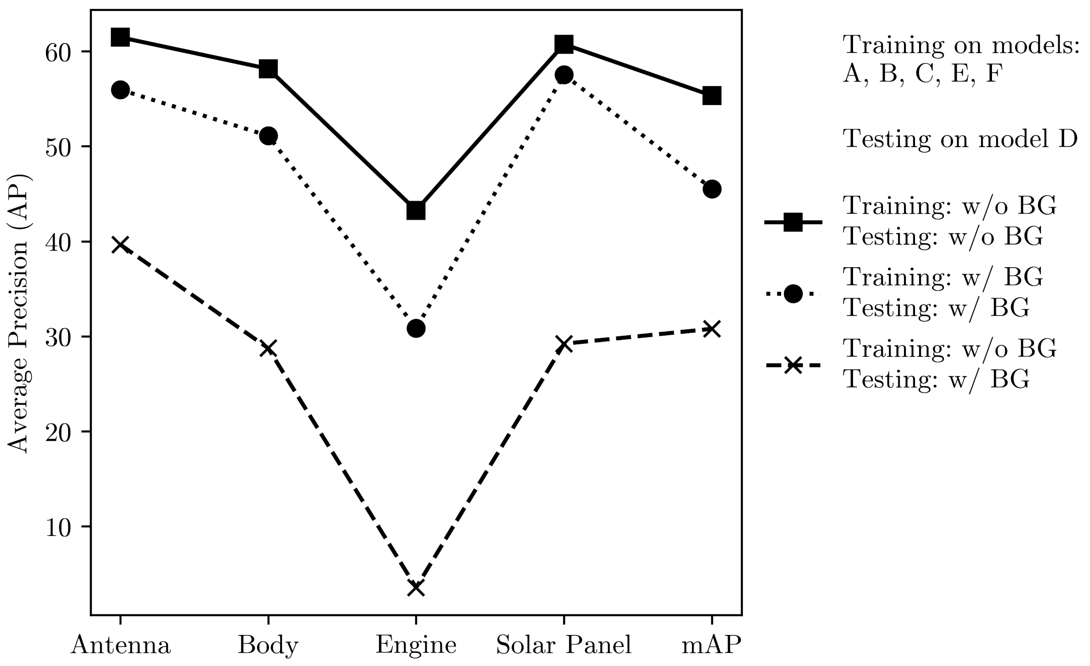
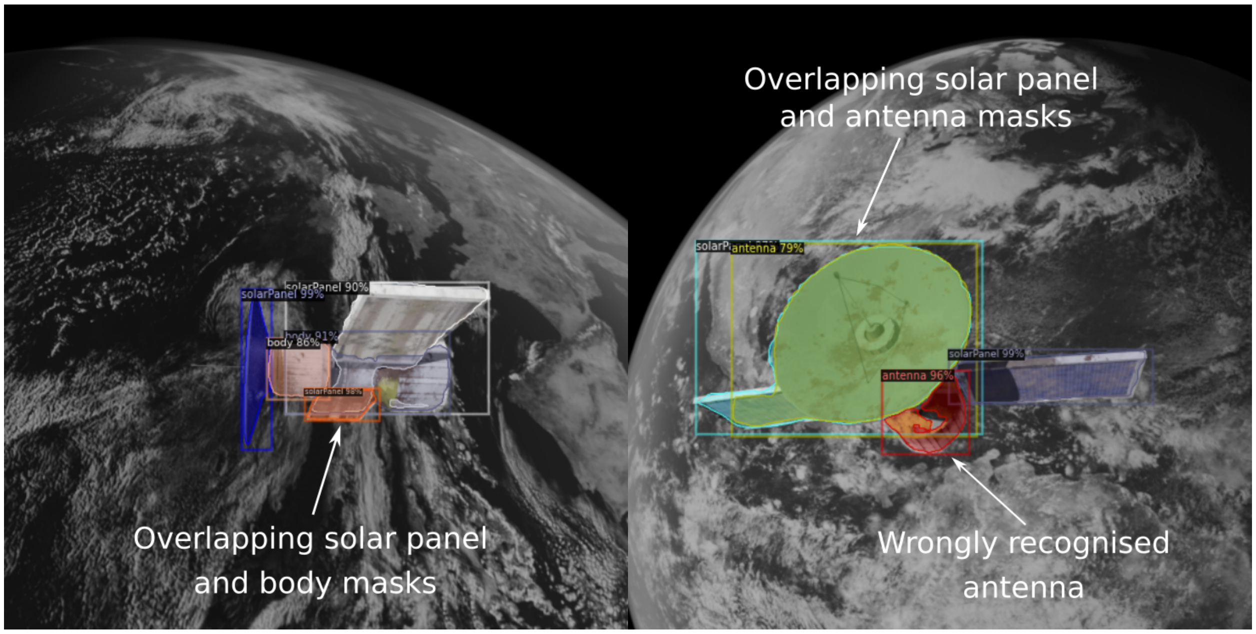


Fig. 7 Influence of the Earth presence in the background (BG) of the image on the inference process. "w/" stands for "with", while "w/o" means "without".



Overlapping solar panel and antenna masks
solarPanel 99%
solarPanel 90%
body 91%
body 86%
antenna 79%
solarPanel 99%
antenna 96%
Overlapping solar panel and body masks
Wrongly recognised antenna

Fig. 8 Results worsening when training on a dataset without the Earth in background and testing on a dataset featuring the planet behind the satellite.

3) Images from a real rendezvous mission

1. Lighting conditions dependence

An animation has been created in Blender in order to simulate the chaser relative orbit around the target spacecraft and the resulting video, similar to what the on-board camera would actually sense on-orbit, has been fed to the recognition algorithm to assess its performance. The orbit has been simulated by a simple motion of the camera around the target, rather than an actual numerical propagation, to the sole aim of obtaining a realistic animation. A more accurate simulation is out of the scope of the present work.

Although the relative orbit would last as long as the orbit around the main attractor and the lighting conditions on the target object would therefore change in time, the lighting direction has been considered fixed in this case for the sake of simplicity. The relative motion of the chasing spacecraft around the target, in fact, is enough to continuously challenge the recognition algorithm with new conditions and to highlight some interesting insights.

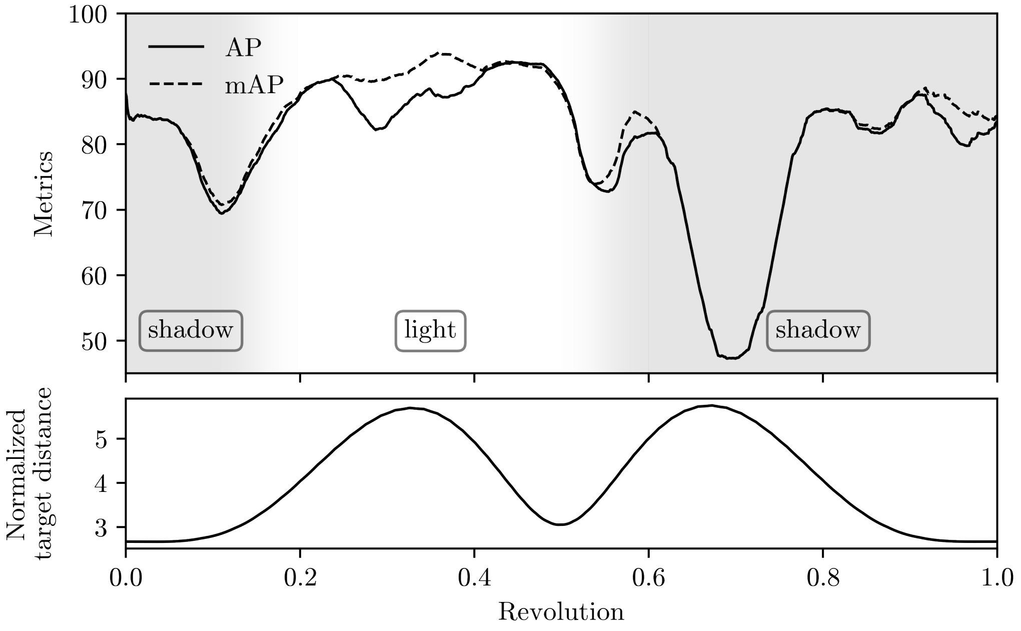


Fig. 9 AP and mAP metrics and relative distance throughout a revolution of the chasing spacecraft around the target, illustrating the dependence of the performance on the lighting conditions.

While the test scores overall values of $AP = 77.070$ and $mAP = 77.917$, other interesting conclusions can be drawn examining the graph in Fig. 9. The upper graph in the picture reports the frame-by-frame variation of the AP and mAP metrics throughout the video, that means throughout a relative orbit of the chasing spacecraft around the RSO. The background of the image visually represents the fraction of the revolution in which the target is well-lit (that means, since the actual lighting conditions do not change, that the point of view of the chasing spacecraft is favorable) and the one in which the target is shadowed. The variation of the distance between the camera and the target, normalized with respect to the spacecraft characteristic dimension of 20 m, is also reported in the lower portion of the picture.

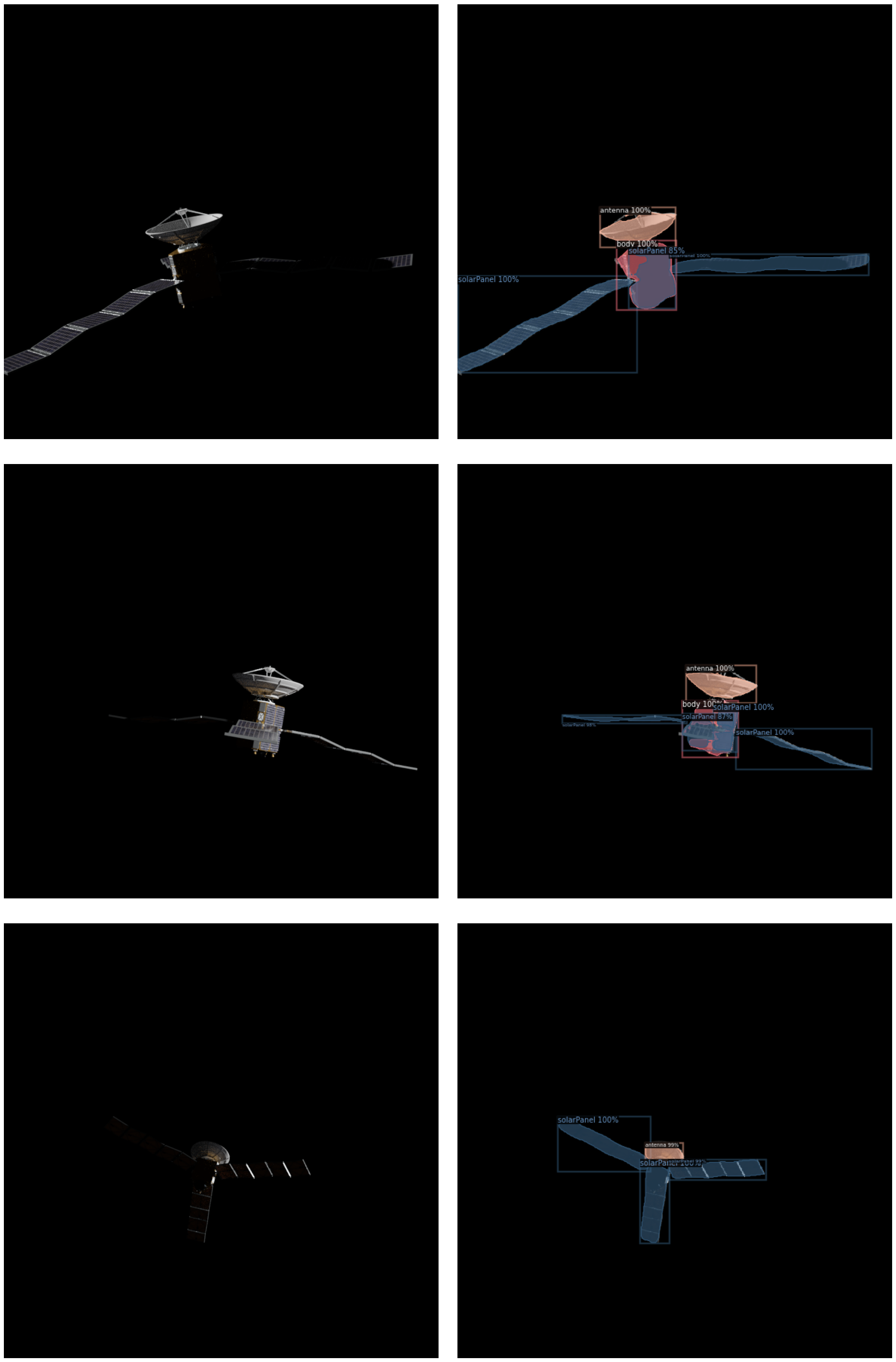


Fig. 10 Some frames extracted from the video simulating the relative orbit around the target object.

It can be easily seen how the mean performance of the algorithm is noticeably better when the target is properly illuminated, while the distance of the camera from the RSO has a marginal effect. Furthermore, it is interesting to notice how the results for the darkest images are all in all not bad and the worst results are obtained during the transition instead. This is due to the fact that the image recognition algorithms are capable of recognizing patterns -even in dark conditions- that are not the same the human eye is used to notice and such patterns are easier to be recognized when the lighting conditions are uniform. Furthermore, the video is generated without adding any noise to the images: this makes it easier for the neural network to distinguish the background from the object and borders of the different items. Recreating a more realistic scene would probably decrease the performance, especially in bad lighting conditions, a problem which is tackled in Sec. IV.D.3.
To a deeper analysis, the reasons causing the three major drops in performance can be individuated. The first one, just before the beginning of the first change in light conditions, is probably due to the bad lighting of the foreground objects as it can be seen in the first row of Fig. 10 and to the fact the camera starts to drift farther from the object, causing part of the body to be confused with a non-existent body-mounted solar panel. The second one, around the half of the revolution (second row of the same picture), is due to the incipient worsening of the lighting conditions and to the occlusion of the body by part of a solar panel, causing a smaller portion of the body to be visible and one of its faces to be mistaken, again, for a body-mounted solar panel. Notice how this second drop is slightly less intense, even though the spacecraft is a little further from the camera. Finally, the pictures of the third row of Fig. 10, explains the reasons of the major drop in performance on the right hand of Fig. 9: the target object is fully in shadow (only some details are visible), the chaser is at the farthest point of its orbit around the target spacecraft and the rear solar panel is perfectly aligned to the body, causing the latter to be mistaken for a part of it, since the bad lighting doesn't allow the algorithm to distinguish the two from the shape only. However, since the time evolution of the metrics does not reproduce the periodicity of the motion, it is safe to assume that, although the higher distance between the chaser and the target may have had a synergical effect, the biggest reason for the severe drop in performance is to be found in the extremely unfavorable lighting conditions.
These considerations, among the others, suggest that the overall performance could be improved by changing the R-CNN algorithm in favor of a Recurrent Neural Network (RNN) [30], since these are able to look at the time sequence of the images and base the prediction on the information received from the previous frames too, rather than looking at the single frame as a standalone picture unconnected to the others (like an R-CNN does).

2. Results on gray-scale images

Most of the cameras on-board the chasing spacecraft shoot gray-scale images, so it is interesting to know how this influences the results. To do so, a python script has been written to turn regular datasets into gray-scale ones. This test case is reported for the sake of completeness, however it is not expected that using grayscale images will significantly impact the performances. Indeed, the first processing step of the neural network always transforms RGB to grayscale by

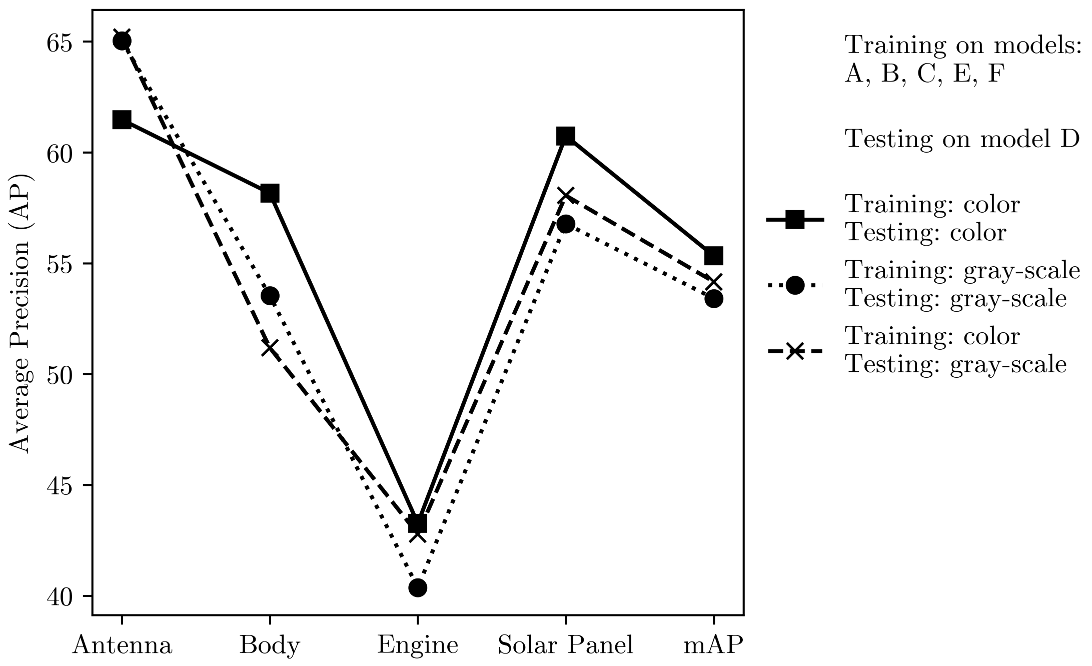


Fig. 11 Results when dealing with images in gray scale encoding.

taking the average of RGB channels for each pixel. In a very similar manner, Blender produces grayscale outputs as a weighted average of the RGB channels, therefore the results should be quite similar.

The dataset used for this test and whose results are reported in Fig. 11, is the one trained on spacecraft models A, B, C, E, and F and tested on model D. Similar results where obtained also using other combinations of training and testing models and with or without the Earth in background. When training and testing on gray-scale images (Fig. 11, dotted line) the results show how the algorithm is quite robust to the change of color encoding with respect to the colored dataset whose results are reported by the solid line for comparison. This further validates the possibility of using the approach in a real scenario. Interestingly enough, the same results are obtained also when training the algorithm on colored images and testing on gray-scale ones (Fig. 11, dashed line), demonstrating how the internal logic of the recognition algorithm puts emphasis on more features than the simple coloring or shape of the objects.
While the comparison of the relative values scored by different datasets allows us to draw the aforementioned conclusions, the considerations on the actual values made in the previous chapters still hold true, since the effects of the absence of model D in the training set and the low occurrence of the engine object in the dataset are still appreciable, regardless of the change in color encoding.

3. Results on Intelsat 901 rendezvous images

Finally, given that gray-scale images do not cause a significant performance drop, the feasibility of the synthetic image generation approach was tested on images from a real mission, namely the rendezvous for the life extension

mission of the Intelsat 901 satellite. As explained in the introduction to this chapter, these images were not included in the training dataset (differently from what is done in previous works, e.g. [11]), but only shown to the network at test time to simulate a real inspection mission of an unknown object. Hence the novelty of this contribution.
Such real images are really noisy and present rings and halos which can worsen the results. In fact, Fig. 12 shows the results obtained using the CNN weights trained on gray-scale images without the Earth in background, where it can be seen how the halos around the objects are mistaken for part of the object itself.

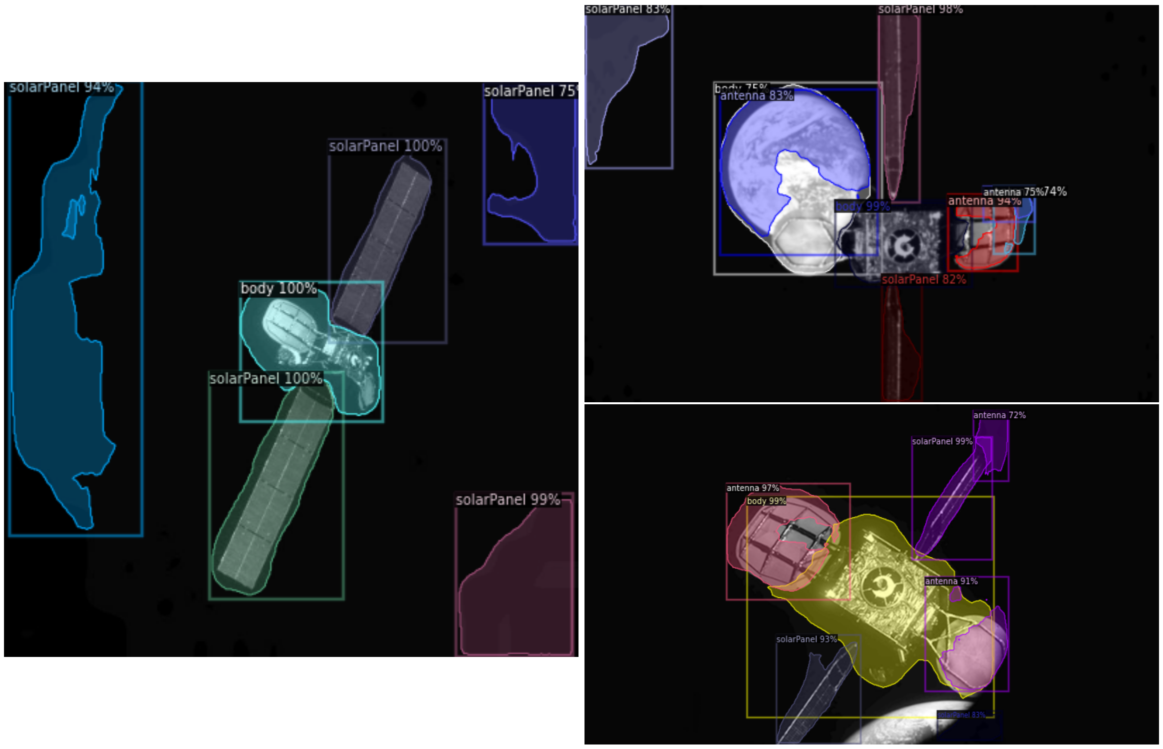


Fig. 12 Inference results when using the model trained on gray-scale images without the Earth in background.

Performing a little image pre-processing, it was possible to obtain a big improvement in the results (see Fig. 13) without having to change the training dataset. In particular, a threshold operation was carried out, which sets exactly to 0-value (that means pitch black) all of the pixels below a specified gray value, therefore getting rid of all the halos.

Notice, however, how the Earth is still not correctly rejected because the algorithm is not trained to recognize it. The only solution to this would be to use a proper training set. In this regard, Fig. 14 shows how, using an algorithm trained on colored images with the Earth in background, the planet is correctly rejected and the results are overall improved to almost visual accuracy, except for an antenna which is not properly recognized.

V. Conclusions

In this work, it has been demonstrated how Convolutional Neural Networks can be used to perform instance segmentation of satellite components in the framework of an inspection mission around a non-cooperative RSO. In

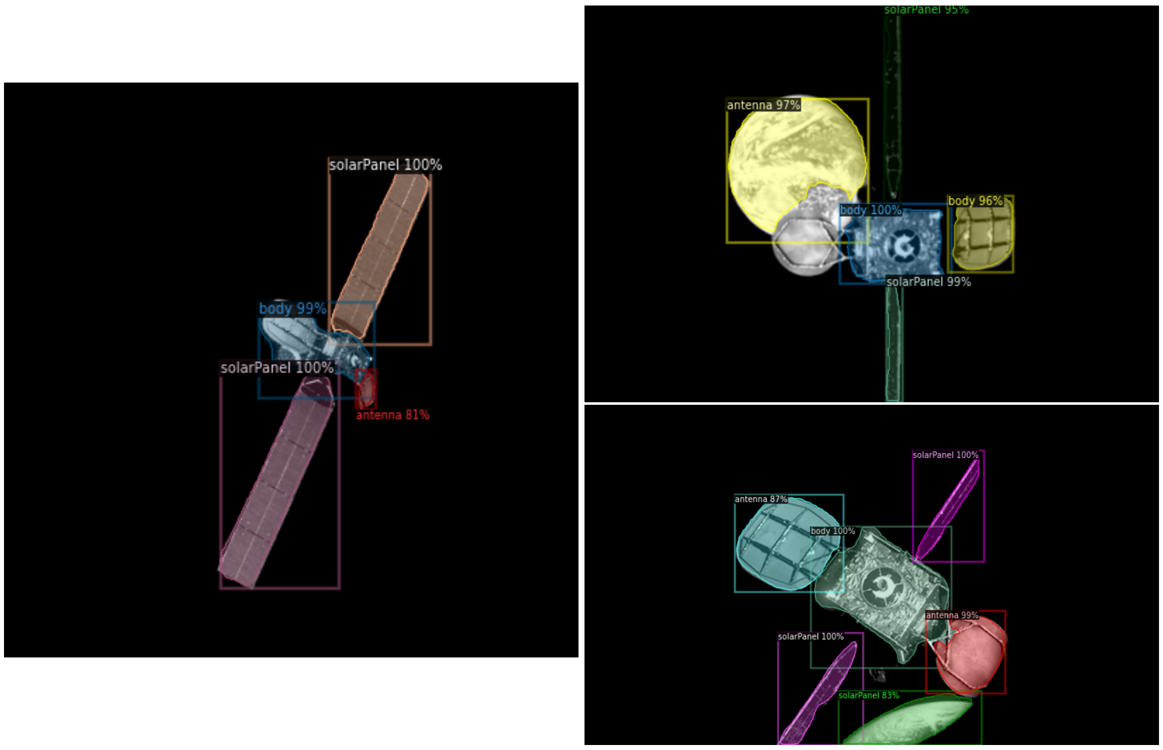


Fig. 13 Inference results when using the same gray-scale model after thresholding the images to get rid of halos.

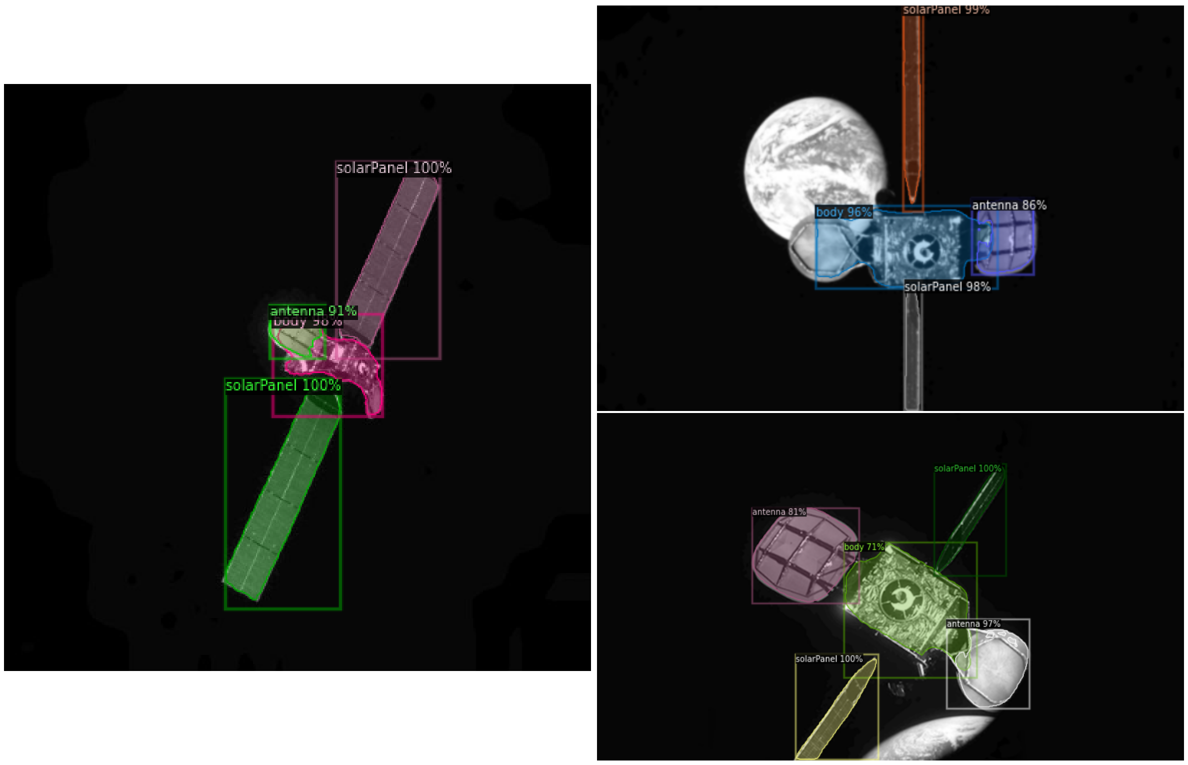


Fig. 14 Inference results when using the model trained on colored images with the Earth in background.

particular, the possibility of generating artificial datasets to be used for the training of such algorithms has been evaluated together with its performance in real life applications. To this aim, a new software, called JINS, has been developed, which takes care of generating an arbitrary amount of suitable training and testing images with minimum intervention by the user. Results have shown how performances are on par with those reported in the literature and, moreover, instance segmentation algorithms trained on such synthetic datasets have also shown to be effective in recognizing objects in images from the real mission Intelsat 901.
The convenience of the proposed approach lies in the ability to automate the dataset generation and the possibility to tailor the learning process on a specific satellite in case the mission is targeted on it. The only limitations of the method have shown to be those intrinsic to the machine learning algorithms used, and those linked to the necessity of choosing a proper training dataset. To the authors' knowledge, relying on an artificial dataset to such an extent has never been done before in this kind of applications in the aerospace field, nor its effectiveness has been proved on real life inspection missions before, hence the relevance of the results obtained in this study.

Other than improving the results by performing a better refinement of the training process, some changes and extensions are planned for the project in order to enhance its performances and range of use. The training datasets can be augmented with images with added noise to make the algorithm better at rejecting rings and halos without any further image manipulation as shown in Sec. IV.D.3. JINS will be turned into a full fledged simulator, using a better model for the Earth, implementing the ability to recreate the real dynamics of the spacecraft motion based on initial conditions and ephemerides, giving the option to create an image based on an input date, etc. Contextually, the software will be adapted to make use of Recurrent Neural Networks when applying the method to real missions, instead of Convolutional Neural Networks, and the image generation tool will be extended to provide the ground truth to train pose estimation algorithms too. Finally, the possibility of building a real simulator with hardware in the loop is being considered too, as the ability to automatically generate segmentation maps on real images is a challenging but promising line of investigation for the further development of the present work.

A. Object detection results

For completeness, the results on object detection corresponding to the ones already discussed on instance segmentation are here reported.

Table 2 Dependence on training model numbers (ref. Fig. 5)

Training models:	A	B, C	B, C, D, E, F
Testing model:	A	A	A
AP Antenna	89.73	23.27	38.91
AP Body	88.37	6.65	15.92
AP Solar Panel	85.47	72.19	74.02
mAP	89.99	29.63	36.86

Table 3 Dependence on Earth presence in background (ref. Fig. 7)

	Training models: A, B, C, E, F; Testing model: D		
	Training: without BG Testing: without BG	Training: with BG Testing: with BG	Training: without BG Testing: with BG
AP Antenna	72.30	67.24	47.42
AP Body	62.61	65.40	37.21
AP Engine	42.06	32.55	37.21
AP Solar Panel	68.72	66.23	38.04
mAP	63.84	58.32	41.15

Table 4 Dependence on images color encoding (ref. Fig. 11)

	Training models: A, B, C, E, F; Testing model: D		
	Training: color Testing: color	Training: gray-scale Testing: gray-scale	Training: color Testing: gray-scale
AP Antenna	72.30	70.23	70.60
AP Body	62.61	57.22	56.92
AP Engine	42.06	34.97	41.53
AP Solar Panel	68.72	66.35	68.97
mAP	63.84	58.94	59.89

References

[1] Klinkrad, H., Beltrami, P., Hauptmann, S., Martin, C., Sdunnus, H., Stokes, H., Walker, R., and Wilkinson, J., "The ESA Space Debris Mitigation Handbook 2002," *Advances in Space Research*, Vol. 34, No. 5, 2004, pp. 1251 – 1259.

[2] Weeden, B., Cefola, P., and Sankaran, J., "Global space situational awareness sensors," *AMOS Conference*, 2010.

[3] Wormnes, K., Le Letty, R., Summerer, L., Schonenborg, R., Dubois-Matra, O., Luraschi, E., Cropp, A., Krag, H., and Delaval, J., "ESA technologies for space debris remediation," *6th European Conference on Space Debris*, Vol. 1, ESA Communications ESTEC Noordwijk, The Netherlands, 2013, pp. 1–8.

[4] Silvestrini, S., Prinetto, J., Zanotti, G., and Lavagna, M., "Design of Robust Passively Safe Relative Trajectories for Uncooperative Debris Imaging in Preparation to Removal," *2020 AAS/AIAA Astrodynamics Specialist Conference*, 2020, pp. 1–18.

[5] NASA's Exploration & In-space Services, "OSAM-1 Successfully Passes Key Decision Point-C," `https://nexis.gsfc.nasa.gov/05292020_osam1_update.html`, 2020.

[6] Gaylor, D. E., and Barbee, B. W., "Algorithms for safe spacecraft proximity operations," *Advances in the Astronautical Sciences*, Vol. 127, 2007, pp. 133–152.

[7] Maestrini, M., and Di Lizia, P., "Guidance Strategy for Autonomous Inspection of Unknown Non-Cooperative Resident Space Objects," *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 6, 2022, pp. 1126–1136. https://doi.org/10.2514/1.G006126.

[8] Sharma, S., *Pose estimation of uncooperative spacecraft using monocular vision and deep learning*, Stanford University, 2019.

[9] Capolupo, F., and Labourdette, P., "Receding-horizon trajectory planning algorithm for passively safe on-orbit inspection missions," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 5, 2019, pp. 1023–1032. https://doi.org/10.2514/1.G003736.

[10] Starek, J. A., Açıkmeşe, B., Nesnas, I. A., and Pavone, M., "Spacecraft autonomy challenges for next-generation space missions," *Advances in Control System Technology for Aerospace Applications*, Springer, 2016, pp. 1–48. https://doi.org/10.1007/978-3-662-47694-9_1.

[11] Sharma, S., and D'Amico, S., "Pose Estimation for Non-Cooperative Rendezvous Using Neural Networks," *CoRR*, Vol. abs/1906.09868, 2019. URL http://arxiv.org/abs/1906.09868.

[12] Pasqualetto Cassinis, L., Fonod, R., Gill, E., Ahrns, I., and Fernandez, J., "CNN-Based Pose Estimation System for Close-Proximity Operations Around Uncooperative Spacecraft," 2020. https://doi.org/10.2514/6.2020-1457.

[13] Huo, Y., Li, Z., and Zhang, F., "Fast and Accurate Spacecraft Pose Estimation From Single Shot Space Imagery Using Box Reliability and Keypoints Existence Judgments," *IEEE Access*, Vol. 8, 2020, pp. 216283–216297. https://doi.org/10.1109/ACCESS.2020.3041415.

[14] Girshick, R. B., Donahue, J., Darrell, T., and Malik, J., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014*, IEEE Computer Society, 2014, pp. 580–587. https://doi.org/10.1109/CVPR.2014.81.

[15] Xu, B., Wang, S., and Zhao, L., "Solar Panel Recognition of Non-cooperative Spacecraft Based on Deep Learnin," *2019 3rd International Conference on Robotics and Automation Sciences (ICRAS)*, 2019, pp. 206–210.

[16] Park, T. H., Märtens, M., Lecuyer, G., Izzo, D., and D'Amico, S., "SPEED+: Next Generation Dataset for Spacecraft Pose Estimation across Domain Gap," *CoRR*, Vol. abs/2110.03101, 2021. URL https://arxiv.org/abs/2110.03101.

[17] Alpaydin, E., *Introduction to Machine Learning*, 2$^{\text{nd}}$ ed., The MIT Press, 2010, Chap. 1.

[18] Ren, S., He, K., Girshick, R. B., and Sun, J., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 6, 2017, pp. 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031.

[19] Girshick, R., "Fast R-CNN," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448. https://doi.org/10.1109/ICCV.2015.169.

[20] He, K., Gkioxari, G., Dollár, P., and Girshick, R. B., "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 42, No. 2, 2020, pp. 386–397. https://doi.org/10.1109/TPAMI.2018.2844175.

[21] Redmon, J., and Farhadi, A., "YOLOv3: An Incremental Improvement," *arXiv*, 2018.

[22] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., and Berg, A. C., "SSD: Single Shot MultiBox Detector," *Computer Vision - ECCV 2016 - 14th European Conference, Proceedings, Part I*, Lecture Notes in Computer Science, Vol. 9905, Springer, 2016, pp. 21–37. https://doi.org/10.1007/978-3-319-46448-0_2.

[23] Pugliatti, M., and Topputo, F., "Navigation about irregular bodies through segmentation maps," 2021.

[24] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R., "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[25] He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, IEEE Computer Society, 2016, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.

[26] Lin, T., Dollár, P., Girshick, R. B., He, K., Hariharan, B., and Belongie, S. J., "Feature Pyramid Networks for Object Detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, IEEE Computer Society, 2017, pp. 936–944. https://doi.org/10.1109/CVPR.2017.106.

[27] Yuxin, W., Alexander, K., Francisco, M., Wan-Yen, L., and Ross, G., "Model Zoo results," https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md, 2019.

[28] Shi, J.-F., Ulrich, S., and Ruel, S., "CubeSat Simulation and Detection using Monocular Camera Images and Convolutional Neural Networks," *2018 AIAA Guidance, Navigation, and Control Conference*, 2018. https://doi.org/10.2514/6.2018-1604.

[29] Powers, D. M. W., "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *Computing Research Repository (CoRR)*, Vol. abs/2010.16061, 2020.

[30] Yu, Y., Si, X., Hu, C., and Zhang, J., "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Computation*, Vol. 31, No. 7, 2019, pp. 1235–1270. https://doi.org/10.1162/neco_a_01199, pMID: 31113301.