SSC21-P2-15

Real-time Satellite Component Recognition with YOLO-V5

Trupti Mahendrakar, Ryan T. White, Markus Wilde, Brian Kish
Florida Institute of Technology
150 W University Blvd, Melbourne FL 32901; 917-213-3039
tmahendrakar2020@my.fit.edu

Isaac Silver
Energy Management Aerospace
440 Cinnamon Drive, Satellite Beach, FL 32937; 321-652-2722
isaac@energymanagementaero.com

ABSTRACT

With the increasing risk of collisions with space debris and the growing interest in on-orbit servicing, the ability to autonomously capture non-cooperative, tumbling target objects remains an unresolved challenge. To accomplish this task, characterizing and classifying satellite components is critical to the success of the mission. This paper focuses on using machine vision by a small satellite to perform image classification based on locating and identifying satellite components such as satellite bodies, solar panels or antennas. The classification and component detection approach is based on "You Only Look Once" (YOLO) V5, which uses Neural Networks to identify the satellite components. The training dataset includes images of real and virtual satellites and additional preprocessed images to increase the effectiveness of the algorithm. The weights obtained from the algorithm are then used in a spacecraft motion dynamics and orbital lighting simulator to test classification and detection performance. Each test case entails a different approach path of the chaser satellite to the target satellite, a different attitude motion of the target satellite, and different lighting conditions to mimic that of the Sun. Initial results indicate that once trained, the YOLO V5 approach is able to effectively process an input camera feed to solve satellite classification and component detection problems in real-time within the limitations of flight computers.

INTRODUCTION

The increasing use of small satellites in Low Earth Orbit (LEO), in particular the deployment of large satellite constellations, and the associated rapid growth of the space debris population, make on-orbit servicing (OOS) and active space debris removal (ADR) an enabling technology for the sustainable, continued growth of spaceflight. Real-world OOS and ADR involve large, non-cooperative target objects that do not feature navigational aids, are not equipped with dedicated capture interfaces, and may have significant tumbling rates, uncertain status of appendage deployment, and structural damage. Therefore, the autonomous characterization of non-cooperative target objects, identification of capture points, planning and execution of safe approach trajectories, capture of the target object, and subsequent attitude stabilization are the missing links on the way to an operational robotic OOS and ADR infrastructure. Distributed Satellite Systems (DSS), swarms of collaborative small satellites with each spacecraft equipped with a relative navigation system, an agile propulsion system, and an adaptive capture mechanism can be a highly scalable solution for OOS and ADR operations with non-cooperative target objects, as illustrated in Figure 1.

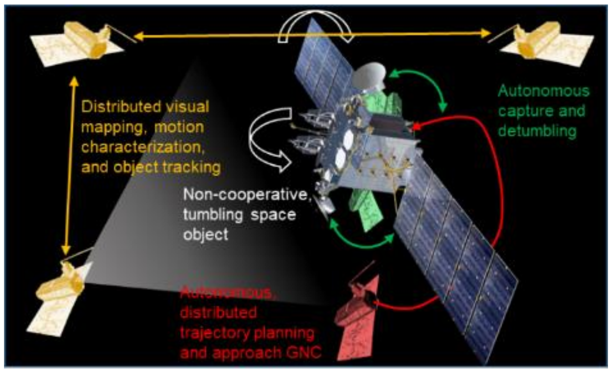

Distributed visual mapping, motion characterization, and object tracking
Autonomous capture and detumbling
Non-cooperative, tumbling space object
Autonomous, distributed trajectory planning and approach GNC

Figure 1: The concept of target object characterization, trajectory planning, approach and detumbling by Distributed Satellite Systems

Collaboratively, the swarming satellites will map the geometry of the target object, characterize its attitude motion, determine its structural health, identify potential capture features and collision hazards, plan safe final approach trajectories to the capture points, and then collaboratively plan and execute detumbling maneuvers. The complexity of this chain of operations, in particular the required capabilities of feature classification and recognition, require the use of machine learning technologies in order to replace the innate capabilities of humans.

The concepts of OOS and ADR have been on the minds of space system designers and mission architects since the dawn of the space age [1]. Ranging from concepts involving pressurized "dry docks", over in-situ maintenance by astronauts, to refueling and repair by autonomous robots, numerous technologies have been developed. The Space Shuttle program repeatedly demonstrated the value of being able to capture a client spacecraft, return it to Earth for repairs and upgrades (Palapa-B2 and Westar VI) or to conduct these operations in orbit (Hubble Space Telescope), and to assemble and supply large spacecraft (ISS) [2]. Due to the cost and risk associated with crewed missions, the focus of OOS research in the 1990s shifted towards robotic on-orbit servicing. Following the groundbreaking Japanese mission ETS-VII in 1997 [3], NASA, DARPA and AFRL demonstrated OOS related capabilities with DART [4], XSS-10 [5], XSS-11 [6], ANGELS [7], MiTEx, and Orbital Express [8]. Orbital Express in 2007 demonstrated a series of end-to-end servicing activities, including autonomous rendezvous and capture, inspection, transferring fuel, and swapping Orbital Replacement Units (ORUs) containing batteries and flight computers. NASA and DARPA are about to follow up on Orbital Express by refueling a spacecraft in Low Earth Orbit during the Restore-L mission [9], and by servicing a GEO spacecraft on the RSGS (Robotic Servicing of Geosynchronous Satellites) mission [10]. Both Restore-L and RSGS are slated to be launched in 2022. In 2020 and 2021, Northrop Grumman flew the Mission Extension Vehicles (MEV) to take over station keeping of GEO spacecraft [11, 12]. The MEV missions mark the first-ever instances of commercial OOS.

All past robotic OOS demonstration missions from ETS-VII through Orbital Express had a highly specialized robotic servicer conduct repair operations on a purpose-built, cooperative servicing target smaller than the servicer. The target spacecraft maintained stable attitude during capture, featured fiducial markings facilitating relative navigation, and were equipped with load-bearing capture interfaces for robotic manipulators. Even the MEVs, docking with and stabilizing spacecraft not designed for OOS, are servicing fully functional spacecraft with stable attitude. Although not equipped with dedicated docking mechanisms, GEO spacecraft are equipped with apogee kick motors and prominent launcher adapter rings on their space deck, which are ideally suited as structural interfaces for station keeping purposes. Restore-L and RSGS will demonstrate more involved servicing operations on non-cooperative servicing clients.

Overall, the safe approach, inspection, capture and servicing on a non-cooperative target spacecraft has not been achieved. After failures of their on-board computers or attitude control systems, target spacecraft can have pronounced tumbling motion. Combined with large antenna apertures or solar arrays, this motion makes approaching the target for inspection or capture hazardous. Structural hard points suitable for capture can be hard to access or can have high rates relative to the approaching chaser. This can lead to high loads during capture that can damage both the target and the chaser, leading to the generation of space debris.

Therefore, in order to make OOS a safe, technologically mature, and economically viable part of spaceflight, major advances in the use of autonomy and artificial intelligence for in-space inspection, characterization of resident space objects, and intelligent path planning are required.

The research reported in this paper closes parts of this capability gap by using the machine learning algorithm YOLO (You Only Look Once) V5, which is based on Convolutional Neural Networks (CNNs), to identify and localize spacecraft bodies, rocket nozzles, antenna apertures and solar arrays. In the OOS context, spacecraft bodies and rocket nozzles represent potential capture locations, whereas antenna apertures and solar arrays defines keep-out-zones, so that the chaser spacecraft does not collide with them or inhibit power generation or communication on the target.

The present paper goes beyond research by Aarestad et al. using similar algorithms in identifying CubeSats for space situational awareness and space traffic management [13], and research by Chen et al. using region-based convolutional neural network (R-CNN) to track different features of satellites for docking [14].

The paper first explains the concepts behind CNNs and the unique properties of the YOLO V5 algorithm in application to aerospace engineering. The paper proceeds to discuss the image database used to train the algorithm and describes the experiments used in the performance evaluation of the algorithm. The experiments were conducted in the Florida Tech Orbital Robotics Interaction, On-orbit servicing, and Navigation (ORION) laboratory, a facility developed to generate high-fidelity representations of spacecraft relative motion and orbital lighting conditions. The paper then discusses the results of the tests and the limitations and growth potential of the algorithms used.

DEEP LEARNING FOR COMPUTER VISION: FROM IMAGE CLASSIFICATION TO OBJECT DETECTION WITH YOLO V5

Deep learning approaches to computer vision have achieved ever-more-impressive successes over the past two decades. Initially, progress was restricted to relatively straightforward image classification tasks. For example, CNNs of LeCun [15] accurately classified small (28-by-28), centered, grayscale images of the 10 handwritten digits by an approach inspired by the neurophysiological studies of the visual cortex of cats by Hubel and Wiesel [16, 17] and the neocognitron of Fukushima [18], and exploited the efficient backpropagation algorithm of Rumelhart, et. al. [19].

Later, tremendous progress was achieved in image classification, peaking with the ImageNet Large Scale Visual Recognition Competition (ILSVRC) [20, 21], which was a yearly competition 2010-2017 that, among other tasks, challenged state-of-the-art image classification models to classify a dataset of 1.2m color images averaging 482-by-418 pixels from 1000 distinct classes. The ILSVRCs gave researchers an opportunity to showcase new developments in deep learning models. The competition ushered in GPU-enhanced CNNs with AlexNet [22], novel architectures such as VGGNet [23] and Inception [24], and new innovations such as advanced optimization techniques [25, 26], initializations, data augmentation (see a survey from [27]), and the models began to be trained quickly enough to permit the use of ensemble learning [28]. By 2015, the ResNet model [29] surpassed human capabilities at the ILSVRC classification task.

With the ImageNet classification problem essentially solved, the ILSVRC and, with it, much of the computer vision community moved on to more challenging tasks than image classification, such as *object localization*, where the goal is to find the location of the most prominent object in an image and draw a bounding box around it. The object localization problem is, therefore, a regression problem where an image is input to the algorithm, and it predicts five numbers:

$$(o, x, y, w, h) \tag{1}$$

This includes four numbers specifying the bounding box: the coordinates of its center (x, y), its width w, and its height h, as we see in the Figure 2.

The remaining number is an "objectness" score o, which is an estimated probability that the predicted bounding box has an object in it. Object localization algorithms attempt to minimize a loss function, typically the sum of squared errors for bounding box locations and dimensions with high objectness score.

A combination of the image classification and object localization tasks is *object detection*, where the goal is not only to localize an object in the image but to simultaneously classify *multiple* objects from *multiple* object classes and localize each one. This paper focuses on solving the object detection problem for objects including spacecraft bodies, thrusters, antennas, and solar arrays applied to individual image frames from a camera feed under heavy computational restrictions. Each image in the dataset used in the project contains zero or more objects that have been manually labeled and localized (see the next section for more details on the dataset).

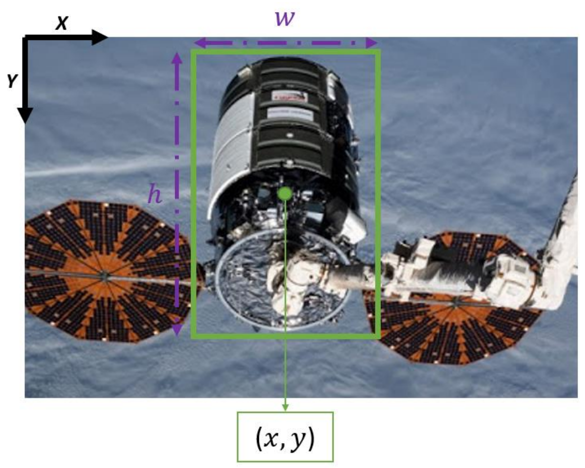


Figure 2: A bounding box for a satellite body

As such, object detectors are regression models that estimate not only the location of *multiple* bounding boxes and objectness scores but also an estimated probability mass function $\boldsymbol{p} = (p_1, p_2, \ldots, p_k)$, where p_i is the conditional probability the bounding box contains an object from class i given that it contains an object and k is the number of classes. Altogether, a successful object detector will produce a (5+k)-dimensional point of the form

$$(o, x, y, w, h, \boldsymbol{p}) \tag{2}$$

and produce such a point for each object in the image. For example, if an image from our dataset contains one solar array and one antenna, an object detector will predict 18 numbers across two 9-dimensional points. For the solar array, the object detector will ideally produce the values

$$(1, x_1, y_1, w_1, h_1, 1,0,0,0) \tag{3}$$

where (x_1, y_1, w_1, h_1) corresponds to a bounding box that tightly surrounds the solar array, the first 1 means the model predicts there is an object in the bounding box, and the (1,0,0,0) implies the object is a solar array with probability 1 and another class with probability 0. Similarly, the point for the antenna should have probabilities (0,0,1,0) with an appropriate bounding box.

How do object detectors find the bounding boxes? Classical (non-neural) object detectors, such as Histogram of Oriented Gradients (HOG) [30], used a "sliding window" approach estimated probability mass functions of the content of overlapping windows sliding left-to-right and top-to-bottom (like one would read in English) over the image, scanning the whole image and classifying each window. These models used non-neural classifiers like support vector machines (SVMs) since the computational cost of this method for traditional neural networks was prohibitively high. The first especially effective neural object detector, OverFeat [31], used a CNN with GPU-acceleration to speed up the sliding windows approach enough to be practical, but another issue emerged: pre-determining the window size did not produce very precise bounding boxes.

From here, multiple popular object detectors emerged. Notably, region-based CNNs (R-CNNs) by Girshick et. al. [32] are based on the idea to segment the image into small chunks, use a selective search algorithm to combine similar regions into larger ones, and propose many candidate region to be fed into a CNN—a few thousand proposals are typical—which extract feature vectors from the regions, which are then fed into an SVM for objectness predictions and a regression algorithm for adjusting the bounding box dimensions for precision before classification. R-CNN's accuracy is much better than its predecessors, but the computation speed of testing is very slow given the cost of selective search and for feeding a large number of regions into the CNN, making it a poor candidate for real-time object detection, even for non-spacecraft hardware. Fast R-CNN [33] feeds the entire image into the CNN, generating a feature map, from which region proposals are identified and selected by a region of interest (RoI) pooling layer and then fed into a fully connected neural network for both classification and bounding box regression. It also initializes the network with a classifier pre-trained to classify the ImageNet dataset, which speeds up training. Faster R-CNN [34] goes further and replaces selective search with a separate region proposal network (RPN) to predict the region proposals. In the same family is Mask R-CNN [35], which detects objects and further identifies the pixels corresponding to the object within the bounding box—performing an *image segmentation* task—at an additional computational cost, and region-based fully convolutional network (R-FCN) [36], which replaces the fully connected layers after the RoI pooling with faster CNNs. These newer variants are fast enough to perform real-time object detection with conventional hardware in many use-cases, but spacecraft onboard computing resources are not sufficient to achieve accurate object detection at a useful framerate with these methods.

The R-CNN family of object detectors are multi-stage models that propose regions with selective search or an RPN, apply the classification model on each region, and then perform post-processing to refine the bounding boxes and eliminate duplicated detections. Single-stage models, such as Single Shot Detectors (SSDs) [37] and the You Only Look Once (YOLO) [38] family of models, attempt to localize and classify objects both with a single neural network, going directly from image pixels to predicted objectness scores, bounding boxes, and class probabilities at once. SSDs achieve slightly less accuracy than Faster R-CNN but run a little faster. YOLO, on the other hand, can run much faster than the others. The original YOLO algorithm ran at 45 frames per second on a previous-generation GPU. Accuracy suffers somewhat with YOLO in exchange for the speed, but it is accurate enough for many use-cases. Importantly, YOLO can run at a lesser but still effective framerate using constrained computational resources suitable for onboard object detection.

How does YOLO detect objects in just one stage so quickly? It partitions the input image into multiple grids of non-overlapping windows, predicts multiple bounding boxes centered in each grid rectangle along with class probabilities in each grid, and predicts class probabilities for objects in each window with bounding box probabilities weighted by objectness scores. It may sound expensive to do this for each window, but YOLO uses the convolutional method originating in OverFeat to make these predictions quickly.

The grid approach to finding bounding boxes typically results in many overlapping bounding boxes that contain the same object. To deal with this, YOLO removes all such bounding boxes with low objectness scores and then uses a method called non-max suppression to choose the best box. Non-max suppression chooses the remaining bounding box with the highest objectness score and compares it to each other remaining bounding box. In this comparison, the area of the intersection of the boxes is divided by the area of their union (IoU)—if IoU is near 1, the box is deemed too similar to the high-objectness box and is removed. This is an incredibly computationally cheap way to exclude lower-quality or redundant bounding boxes.

Like Fast R-CNN and Faster R-CNN, YOLO uses a CNN pretrained for classifying the ImageNet dataset. But, unlike these models, YOLO converts the underlying

20-layer DarkNet CNN (the “backbone” of the model) to perform object detection in its entirety. It first adds four convolutional layers and two fully connected layers (the “neck” of the model) with randomly initialized weights to improve classification accuracy, and then adds a final layer (the “head” of the model) to predict both class probabilities and bounding boxes simultaneously. This segmentation of the CNN into a pretrained backbone, a neck to train to the dataset at hand, and the head to make the final prediction of bounding boxes and class probabilities has been an influential idea that has continued through the YOLO family and other models.

YOLO9000 (YOLO V2) [39] makes some incremental improvements like adding batch normalization to the convolutional layers, using a higher resolution classifier, and fine-tuning the pretrained net to improve accuracy, as well as using finer grid to help with detecting small objects, removing fully connected layers to improve speed, and randomizing training image size to avoid a bias YOLO had toward the scale of training data. But, most notably, YOLO V2 introduced anchor boxes, which allow the net to pre-specify typical sizes, aspect ratios, and/or locations for each class. YOLO V2 selects these using K-means clustering on the training datas’ bounding boxes for each class, resulting in significantly higher accuracy at higher framerates.

YOLO V3 [40] is, as its tech report states, an incremental improvement. It predicts objectness scores with logistic regression, replaces the final softmax layer in the classifier with independent logistic regression classifiers and replaces sum of squared error loss with binary cross-entropy to improve in cases where there are overlapping labels for the same object (e.g. thruster and nozzle), makes multiple predictions at each grid cell, and implements a CNN similar to YOLO V2 but deeper (i.e. with more layers) and with shortcut connections pioneered in ResNets [29], which improve accuracy without much cost to speed. More interestingly, the authors further use an idea from Feature Pyramid Networks (FPNs) [41] that make bounding box and class probability predictions at three different scales and use them all to inform the object detection, which helps with predictions across scales.

YOLO V4 [42], written by a different author than previous versions, makes some more incremental improvements, primarily adjusting how the CNN learns by performing wide-ranging hyperparameter tuning experiments involving the activations, loss functions, CNN architectures, data augmentation, regularization, normalization, and optimization algorithms. This results in small improvements across the board to training time, accuracy, and framerate.

YOLO V5 [43] has been somewhat controversial, as the original author is no longer involved, and a third developer released a PyTorch implementation of a YOLO model named YOLO V5 with claims of superiority that are disputed by the second author. Controversy aside, performance of YOLO V4 and YOLO V5 seem very similar, but PyTorch is somewhat more convenient, so it is used in the remainder of the paper.

TRAINING DATA SET

To accomplish the goals of the research, an image dataset consisting of a total of 523 images of solar arrays, satellite bodies, antennas and thruster nozzles was developed. The images in the database come from the internet, from satellite models used in Kerbal Space Program, and from geometric models used in AGI Systems Tool Kit (STK). The images were chosen based on these criteria: (1) the object should be identifiable and each class should be distinguishable from one another to prevent any errors in labeling the data; (2) the shapes of the components in the images should resemble real life components; (3) no image should be used twice

The dataset was then annotated by drawing bounding boxes around each object within each image with Roboflow [44]. Roboflow is a computer vision image annotating tool capable of automatically exporting annotated images in a convenient format. Figure 3 shows an illustration for the annotations.

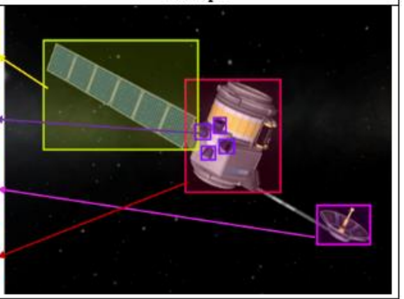
Class	#Annotations	Example
Solar Array	1204	
Thrusters	737	
Antennas	692	
Satellite Bodies	416	

Figure 3: Classes and Annotations

The overall dataset was then split into a training set with 366 images, a validation set with 104 images and a testing set with 53 images. The number of annotations in each set is summarized in Figure 3.

TESTING DATA DEVELOPMENT

The testing video dataset was acquired using the Florida Tech ORION Lab [45]. The ORION Lab’s planar, cartesian Maneuver Kinematics Simulator shown in Figure 4 has a workspace of 5.5 m × 3.5 m. The primary component of the kinematics simulator is a horizontal

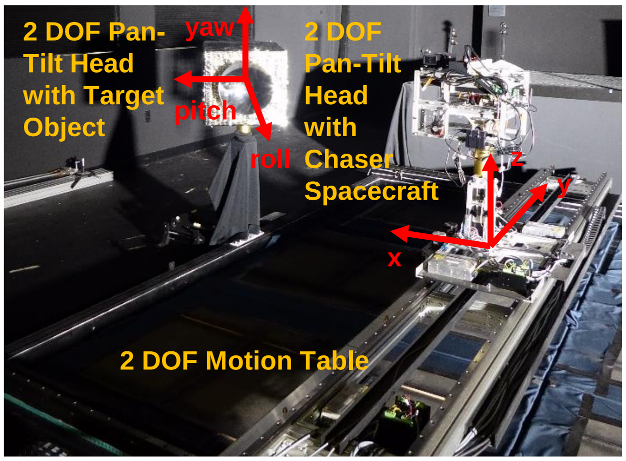

2 DOF Pan-Tilt Head with Target Object
yaw
pitch
roll
2 DOF Pan-Tilt Head with Chaser Spacecraft
z
y
x
2 DOF Motion Table

Figure 4: ORION Spacecraft Maneuver Kinematics Simulator

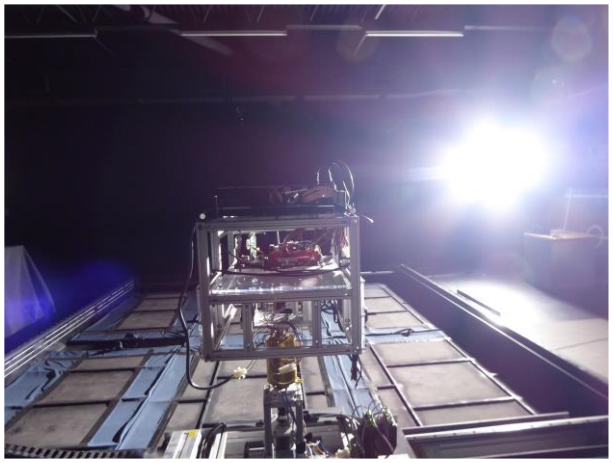


Figure 5: Chaser Spacecraft and LED Light Panel

2 DOF motion table capable of positioning a payload of 80 kg at a maximum speed of 0.25 m/s and a maximum acceleration of 1 m/s^2 along both linear axes. The motion table is designed to carry a wide range of equipment, such as small industrial manipulators or pan-tilt mechanisms. The ORION pan-tilt mechanisms are custom designed to carry a test article with mass 20 kg and dimensions 0.5 m × 0.5 m × 0.5 m. The motion envelope is ±90° in elevation and infinite rotation in azimuth, with maximum rotation rate 60°/s and maximum acceleration 60°/s^2 about each axis. The test article is supplied with power and Ethernet connections via a slip ring around the azimuth axis. The Maneuver Kinematics Simulator currently employs two pan-tilt mechanisms. The stationary pan-tilt head is used to generate the attitude motion of a target spacecraft model. The target model has geometrical and surface features typically found on a satellite, such as parabolic antenna, thruster nozzles, solar arrays, etc. The moving pan-tilt head is designed to carry a spacecraft robotics test vehicle equipped with a number of robotic manipulators, a capture tool, multiple cameras and distance sensors.

As shown in Figure 5, the ORION simulator uses commercial-off-the-shelf components to generate a light source sufficiently bright to exceed the dynamic range of common optical sensors while providing a narrow beam angle. The walls, floor, and ceiling of the testbed are painted a low-reflectivity black and all windows are covered with black-out blinds to fully control the lighting conditions. The selected light source is a Litepanels Hilio D12 LED panel. The panel generates light with a color temperature of 5,600 K (daylight balanced) with 350 W of power. The intensity is equivalent to a 2,000 W incandescent lamp. The intensity can be continuously dimmed from 100% to 0%, and the beam angle can be varied between 10° and 60° using lens inserts. Therefore, the light can be used to simulate solar illumination and also the weaker and diffuse Earth albedo. The LED panel is mounted on a wheeled tripod for quick positioning anywhere within the lab space.

For testing data, sixteen test videos of the target object mock-up were shot by the chaser spacecraft approaching the target to evaluate the performance of the algorithm. Out of the sixteen, videos 1 through 4 included the target vehicle either being stationary or yawing or both yawing and pitching, while the chaser was approaching with a constant velocity in either x, y or both x and y directions. The last 12 videos were shot by turning off the overhead lights and using the LED light panel from Figure 5 while the chaser followed the same constraints used in the first 4 videos.

Some images from the videos along with constraints used are summarized in Figure 6 - Figure 9. The motion and lighting conditions in those test cases were selected to closely resemble real-life rendezvous missions.

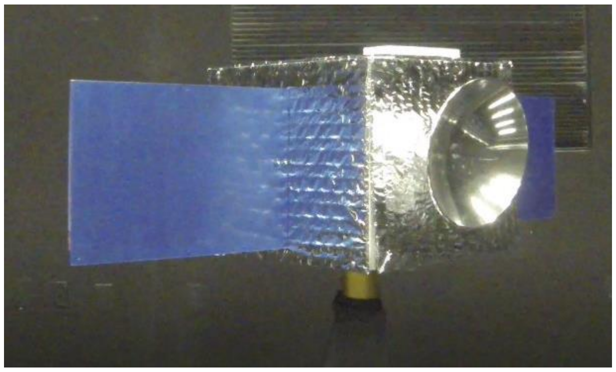
TRAINING AND TEST Results

Training

As expected, the most realistic videos were the last 12 cases, with the light source in-plane with chaser and target. Only test case 3 was analyzed for this paper due to its high complexity in comparison with the other videos.

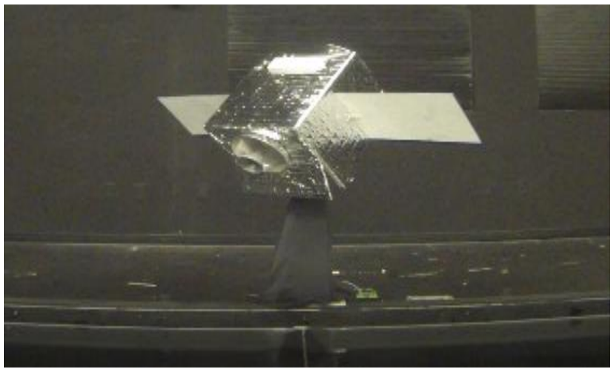
The default Ultralytics YOLO V5 network [43] along with several pre-processing image augmentation techniques listed below were used to obtain results mentioned in the paper.

The following augmentation was applied to create 3 versions of each source image:



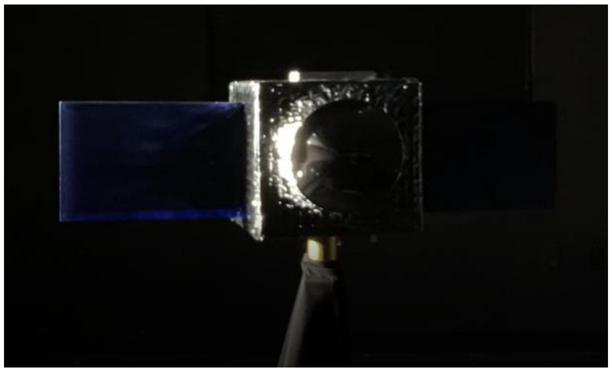
Light source:	Overhead
Target yaw | pitch | roll rates:	10 | 0 | 0 deg/s
Chaser x | y | z velocity:	10 | 0 | 0 cm/s

Figure 6: Test Case 1



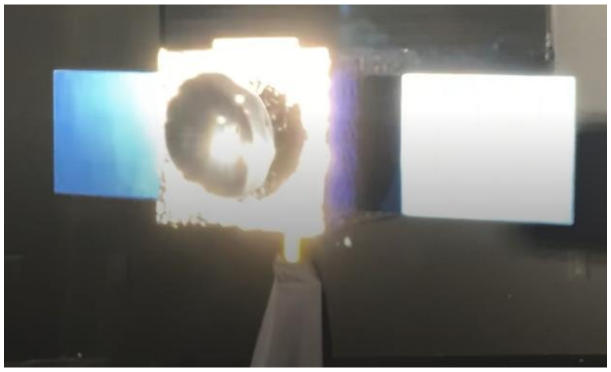
Light source:	Overhead
Target yaw | pitch | roll rates:	20 | 10 | 0 deg/s
Chaser x | y | z velocity:	10 | 10 | 0 cm/s

Figure 7: Test Case 2



Light source:	In plane, at 135° yaw
Target yaw | pitch | roll rates:	10 | 0 | 0 deg/s
Chaser x | y | z velocity:	10 | 10 | 0 cm/s

Figure 8: Test Case 3



Light source:	In plane, at 150° yaw
Target yaw | pitch | roll rates:	10 | 0 | 0 deg/s
Chaser x | y | z velocity:	10 | 10 | 0 cm/s

Figure 9: Test Case 4

1. Randomly crop between 0 and 34 percent of the image
2. Random rotation of between -28 and +28 degrees
3. Random brightness adjustment of between -49 and +49 percent
4. Random Gaussian blur of between 0 and 10 pixels
5. Salt and pepper noise (dark spots in white regions and white spots in dark regions) was applied to 13 percent of pixels.

These augmentation techniques were chosen based on how the images in the dataset could closely resemble spacecraft in outer atmosphere. With these augmentation techniques, the algorithm achieved the highest average true positive values of 75% for the body, 64% for the solar array, 54% for the antenna and 15% for thrusters. The code currently classifies in real-time at the speed of 140 frames per second (FPS) using a Tesla P100 GPU via Google Colab. Figure 10, Figure 11 and Figure 12 contain results of the object detection algorithm and other metrics essential for measuring performance.

In Figure 10, rows of the confusion matrix correspond to the predicted values and columns correspond to the actual values. Ideally, all objects would be classified properly, meaning the diagonal should include all classifications in its row and column, which was approximately true for all classes except thrusters so far.

The bottom row represents false negatives (FN) corresponding objects that were in the frame but failed to be detected by the algorithm, where we noticed some significant problems with thrusters, lesser mistakes with antennas, but broad success with bodies and solar arrays. The rightmost column represents false positives (FP) where the algorithm detected an object where there was none. Here, most mistakes were again associated with thrusters.

Thrusters and antennas currently perform poorly partially because thrusters come in many different shapes and patterns. A classical bell nozzle-shaped thruster, an ion thruster, and a hole in the satellite body

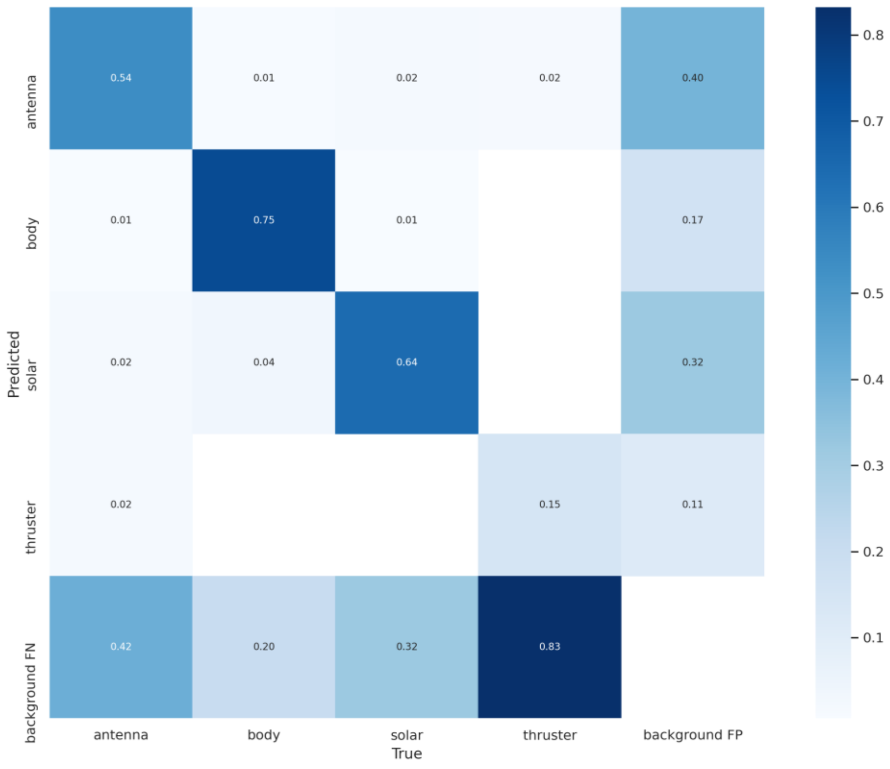


Figure 10: Confusion Matrix

were classified as thrusters. Similarly, antennas have different shapes such as a parabolic, helical, and horn antennas. This lack of segregation in the dataset has led to fewer than the actual number of annotations for these classes causing error in the algorithm.

The plots in Figure 11 demonstrates the trend in the precision, recall and mAP values. Higher the precision, recall and mAP, better the algorithm works.

Precision is the ratio of true positive (TP) values to the sum of true positive and false positive (FP) values. Whereas recall is the ratio of true positive and false negative values.

mAP as defined in equation 4 is the mean average precision over all the classes in the dataset. mAP@0.5 is the mean average precision across all classes at IoU

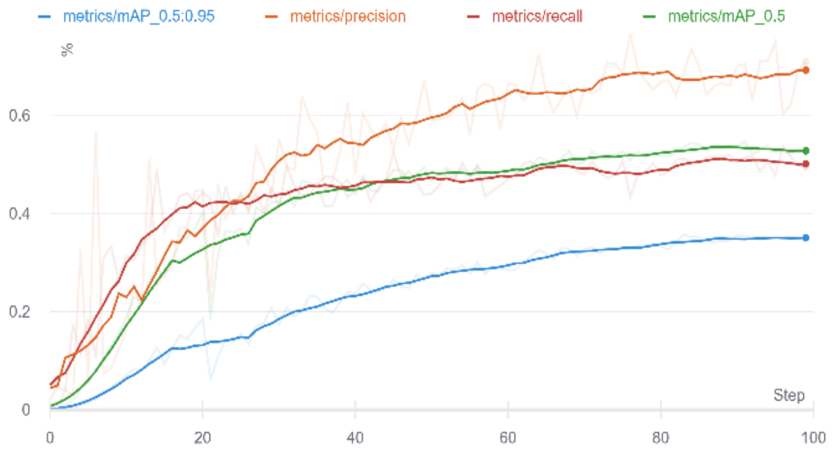


Figure 11: Training Metrics

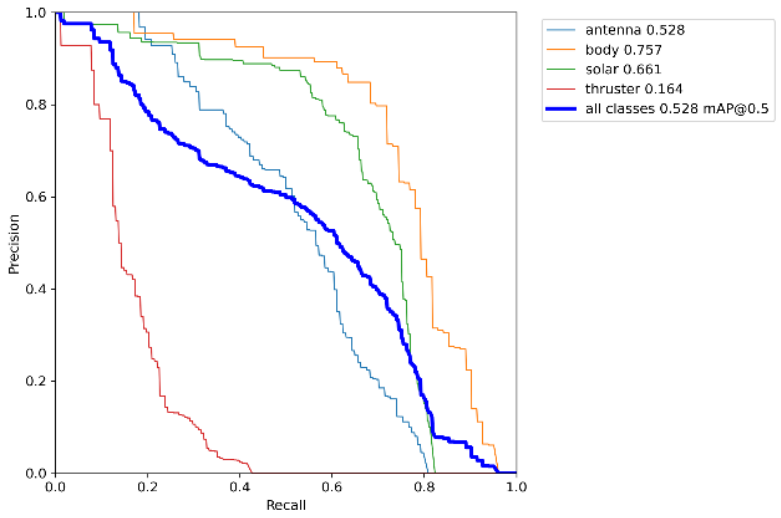


Figure 12: Multi-Class P-R Curve

threshold 0.5. mAP@0.5:0.95 is the average mAP over different IoU thresholds equally spaced from 0.5 to 0.95.

$$mAP = \frac{1}{|classes|}\Sigma_{c\epsilon classes}\#\frac{TP(c)}{\#TP(c) + \#FP(c)} \qquad (4)$$

Figure 12 shows the precision-recall curve for each class. As seen here as well, the precision value drops with increasing recall values for thrusters while, antennas perform almost around the average curve for all classes and, solar panels and body stay above the average curve. The plots indicate the model is undertrained. The results can be improved due to the lack of curves plateauing before or at 100 epochs by simply training the model longer.

Testing

To continue evaluating the algorithm, case 3 video from Figure 8 was tested with the weights generated from the training algorithm. Figure 13 shows a frame from the video before and after using YOLO V5 inference.

Table 1: Error Analysis Summary of Case 3 Test Video

	Average Accuracy	Actual	YOLO Detections	FN	FP
Thrusters	1.10%	16	1	0	0
Antennas	52%	19	35	8	10
Body	52%	37	28	9	1
Solar Panels	62.90%	42	32	9	2

Though the video is inferred at 140 FPS, the inferred video was analyzed every second for 37 seconds for simplicity to get an estimate of the performance of the algorithm. Table 1 below contains a summary of the error analysis of case 3 test video.

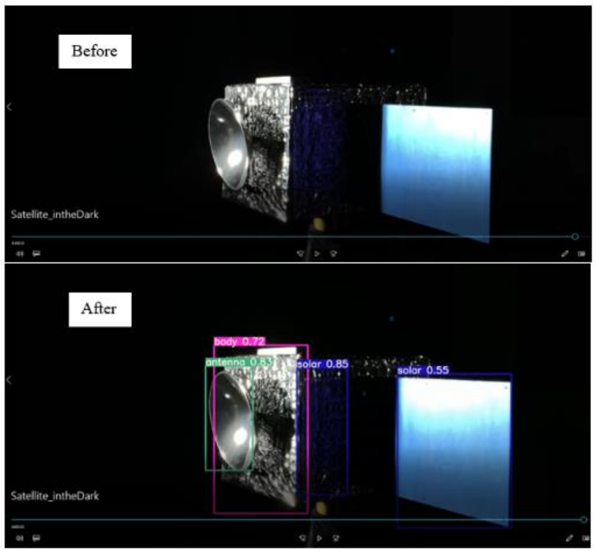


Figure 13: Before and After YOLO V5

The term “actual” in Table 1 corresponds to the feature visibility in the video at that second, “YOLO detections” state how many times the algorithm detected a feature, FN and FP correspond to the correctness of the YOLO detection.

Antennas have the highest FP values. It was seen that the algorithm frequently detected the chaser’s stand and the rails of the test bed as antenna because of its low gain antenna like shape and, an emergency light of the ORION lab in the background because of its parabolic shape. Figure 14 shows a collage of frames of the observation.

All three classes – antennas, body and solar panels showed a high false negative value which is problematic. However, case 3 has the worst lighting conditions out of all the videos that were shot at the ORION lab. Additionally, the training dataset chosen does not resemble the testing dataset accurately since the training dataset images are well lit and relatively easier to identify leading to a spike in the FN.

Finally, the worst performing class is thrusters as discussed earlier.

Some ways to improve accuracy would be to apply more augmentation techniques, generate images of satellites in black background using CAD software Blender as discussed in [13], and segregate the thruster and antenna dataset into multiple other classes based on shape and pattern.

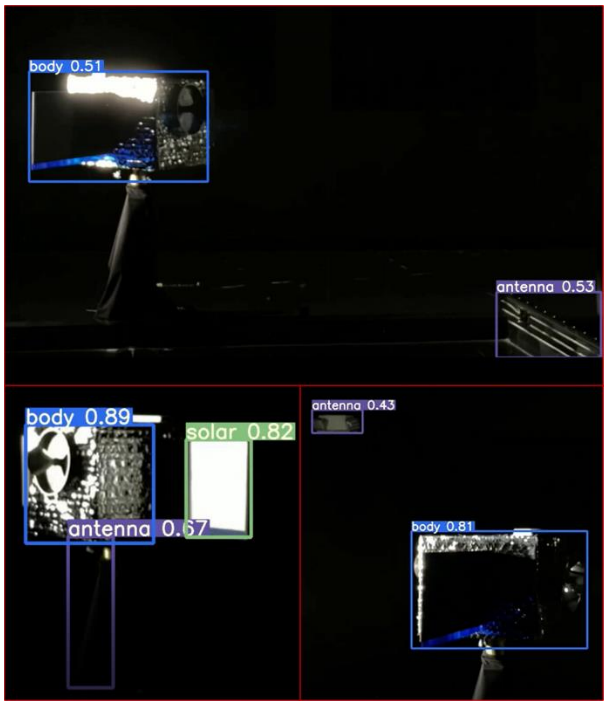


Figure 14: Test Observation - Antenna and Solar Panels

CONCLUSION

Based on initial testing of YOLO V5 on the satellite dataset developed by the ORION lab, test video observation, analysis and results discussed in this paper, real-time detection of different components of satellite is feasible. Since the algorithm is not completely reliable at the moment, ongoing and future research focuses on incorporating additional pre-processing and augmentation techniques, ensemble methods and regression techniques for the bounding boxes to increase the detection accuracy and reliability of the algorithm.

ACKNOWLEDGMENT

We thank Florida Institute of Technology undergraduate students –Nathan Fischer, Andrew Ekbald and Nouraldean El-Chariti for assisting with data accumulation.

REFERENCES

1. M. H. Skeer, "Potential Satellite Servicing Operations and the Impact of Servicing on Satellite Design," Bellcomm, Inc., Washington, DC, 1969.
2. J. L. Goodman, "History of Space Shuttle Rendezvous and Proximity Operations," *Journal of Spacecraft and Rockets,* vol. 43, no. 5, pp. 944-959, 2006.
3. K. Yoshida, "Engineering Test Satellite VII Flight Experiments For Space Robot Dynamics and Control: Theories on Laboratory Test Beds Ten Years Ago, Now in Orbit," *The International Journal of Robotics Research,* vol. 22, no. 5, pp. 321-335, 2003.
4. R. T. Howard and T. C. Bryan, "DART AVGS flight results," *Proceedings of SPIE,* vol. 6555, no. Sensors and Systems for Space Applications, pp. 1-10, 2007.
5. T. M. Davis and D. Melanson, "XSS-10 Micro-Satellite Flight Demonstration Program Results," *Proc. of SPIE,* vol. 5419, no. Spacecraft Platforms and Infrastructure, pp. 16-25, 2004.
6. AFRL, "XSS-11 Micro Satellite," September 2011. [Online]. Available: https://www.kirtland.af.mil/Portals/52/documents/AFD-111103-035.pdf?ver=2016-06-28-110256-797. [Accessed 17 July 2020].
7. AFRL, "Automated Navigation and Guidance Experiment for Local Space (ANGELS)," July 2014. [Online]. Available: https://www.kirtland.af.mil/Portals/52/documents/AFD-131204-039.pdf?ver=2016-06-28-105617-297. [Accessed 17 July 2020].
8. F. G. Kennedy III, "Orbital Express: Accomplishments and Lessons Learned," *Advances in the Astronautical Sciences,* vol. 131, no. Guidance and Control 2008, pp. 575-586, 2008.

9. B. J. Reed, R. C. Smith, B. Naasz, J. Pellegrino and C. Bacon, "The Restore-L Servicing Mission," in *AIAA SPACE Forum*, Long Beach, CA, 2016.

10. G. Roesler, P. Jaffe and G. Henshaw, "Orbital Mechanics," *IEEE Spectrum Magazine*, pp. 45-50, March 2017.

11. C. Gebhardt, "Northrop Grumman makes history, Mission Extension Vehicle docks to target satellite," NASASpaceflight.com, 26 February 2020. [Online]. Available: https://www.nasaspaceflight.com/2020/02/northrop-grumman-history-mission-extension-vehicle-docks-satellite/. [Accessed 25 May 2021].

12. C. Gebhardt, "Mission Extension Vehicles succeed as Northrop Grumman works on future servicing/debris clean-up craft," NASASpaceflight.com, 7 May 2021. [Online]. Available: https://www.nasaspaceflight.com/2021/05/mev-success-ng-future-servicing/. [Accessed 25 May 2021].

13. J. Aarestad, A. Cochrane, M. Hannon, E. Kain, C. Kief, S. Lindsley and B. Zufelt, "Challenges and Opportunities for CubeSat Detection for Space Situational Awareness using a CNN," in *34th Annual Small Satellite Conference*, Logan, UT, 2020.

14. Y. Chen, J. Gao and K. Zhang, "R-CNN-Based Satellite Components Detection in Optical Images," *International Journal of Aerospace Engineering*, vol. 2020, no. 8816187, p. 10, 2020.

15. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, p. 2278–2324, 1998.

16. D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat\textquotesingles striate cortex," *The Journal of Physiology*, vol. 148, p. 574–591, 10 1959.

17. D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, p. 106–154, 1 1962.

18. K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, p. 193–202, 4 1980.

19. D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, p. 533–536, 1986.

20. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

21. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, p. 211–252, 2015.

22. A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12)*, 2012.

23. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.

24. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

25. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint, arXiv:1412.69*, 2014.

26. 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," *The Journal of Machine Learning Research*, vol. 15, p. 1929–1958, 2014.

27. C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, 2019.

28. T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Proceedings of the First International Workshop on Multiple Classifier Systems*, Berlin, 2000.

29. K. He, X. Zhang, S. Ren and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision – ECCV 2016*, Springer International Publishing, 2016, p. 630–645.

30. N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR\textquotesingle05)*.

31. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," in *2nd International Conference on Learning Representations, ICLR*

2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, 2014.

32. R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
33. R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.
34. S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, 2015.
35. K. He, G. Gkioxari, P. Dollar and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.
36. J. Dai, Y. Li, K. He and J. Sun, "R-FCN: Object Detection via Region-Based Fully Convolutional Networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2016.
37. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, Springer International Publishing, 2016, p. 21–37.
38. J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *IEEE Conference*, Las Vegas, 2016.
39. J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
40. J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 8 4 2018.
41. T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
42. A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 23 4 2020.
43. G. Jocher, A. Stoken, J. Borovec, A. Chaurasia, L. Changyu, V. Abhiram, A. Hogan, A. Wang, J. Hajek, L. Diaconu, Y. Kwon, Y. Defretin, A. Lohia, B. Milanko, B. Fineran, D. Khromov, D. Yiwei and F. Ingham, *ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations*, Zenodo, 2021.
44. "Roboflow," 13 May 2021. [Online]. Available: https://roboflow.com/.
45. M. Wilde, B. Kaplinger, T. Go, H. Gutierrez and D. Kirk, "ORION: A Simulation Environment for Spacecraft Formation Flight, Capture, and Orbital Robotics," in *Proceedings of the 2016 IEEE Aerospace Conference*, Big Sky, MT, 2016.