CS230

Semantic Image Segmentation of Imagery of Unmanned Spacecraft Using Synthetic Data

Computer Vision category

William Armstrong
Stanford University
SUNet ID: wrmstrng
wrmstrng@stanford.edu

Spencer Draktontaidis
Stanford University
SUNet ID: sdrak
sdrak@stanford.edu

Nicholas Lui
Stanford University
SUNet ID: niclui
niclui@stanford.edu

CS230: Deep Learning, Autumn 2021, Stanford University, CA. (LaTeXtemplate borrowed from NIPS 2017.)

1 Introduction

Images of spacecraft photographed from other spacecraft operating in outer space are difficult to come by, especially at a scale typically required for deep learning tasks. Semantic image segmentation, object detection and localization, and pose estimation are well researched areas with powerful results for many applications, and would be very useful in autonomous spacecraft operation and rendezvous. However, Wong et al. [22] notes that these strong results in broad and common domains may generalize poorly even to specific industrial applications on earth.

To address this, we have generated a prototype synthetic image dataset labelled for semantic segmentation of 2D images of unmanned spacecraft, and are endeavouring to train a performant deep learning image segmentation model using the same, with the ultimate goal of enabling further research in the area of autonomous spacecraft rendezvous.

2 Related work

Minaee et al. [12] and Ghosh et al. [8] provide recent surveys of deep learning approaches for semantic image segmentation. Treml et al. [20] discusses semantic segmentation using deep learning for autonomous terrestrial vehicles (i.e. self-driving cars).

Arantes et al. [3] present a discussion of machine vision pose estimation for on-orbit autonomous satellite intercept and rendezvous with uncooperative spacecraft using a monocular optical sensor, and Hussain et al. [9] present a solution for such using convoloutional neural networks. Kisantal et al. [10] presents a synthetic dataset of images of unmanned spacecraft labelled for pose estimation, including images generated from 3D computer models, and hardware-in-the-loop simulation using a physical mock-up.

Wong et al. [22] in response to concerns about the availability of training data at scale for domain-specific object recognition tasks, proposes a method of procedurally generating large image datasets from 3D models based on a small number of physical examples, labelled for image classification.

3 Dataset

We prepared a synthetic dataset of 60,000 images in total, consisting of nearly photo-realistic renderings of one of four open-source 3D models of unmanned spacecraft published by NASA [13], with accompanying ground truth labels, using the open source 3D modeling software Blender [4].

Our major hypothesis is that since this the task is fundamentally about object or component recognition, images with a degree of verisimilitude to the human eye would be useful for training deep learning models that would be effective in a practical application.

3.1 3D models

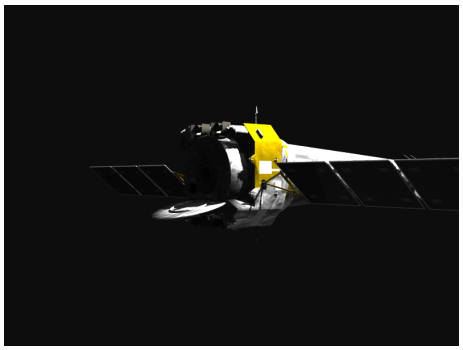


Figure 1: Chandra spacecraft 3D Model

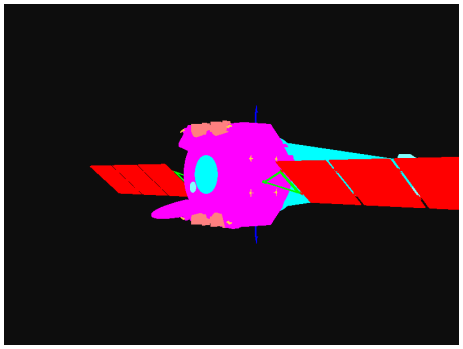


Figure 2: Corresponding segmentation mask

To provide our dataset with a variety of spacecraft configurations, we chose the Chandra X-Ray Observatory, Near Earth Asteroid Rendezvous – Shoemaker (NEAR Shoemaker), Cluster II, and the IBEX Interstellar Boundary Explorer, as 3D models from which to generate training, validation, and test images in this domain.

We made some artistic modifications to the published 3D models in order to be able to generate more realistic looking images where possible, and attempted to simulate the lighting conditions of low-earth orbit by illuminating the model with two light sources: one light source at infinity simulating the intensity, color, and parallel light rays of the sun, and one planar light source to simulate earthshine, i.e. the sunlight reflected by the surface of the earth. As such, our simulated environment assumes that rendezvous is taking place on the day side of the earth, which is perhaps not unreasonable since we are assuming an optical sensor.

We then duplicated the 3D model used for image generation and manually colored each polygon of the duplicate model from a discrete set of colors uniquely associated with one of the class labels. Generation

of the semantic masks was then accomplished by removing all light sources from the Blender scene and modifying the shaders, material properties of each component, and rendering settings such that the raytracer projected only the appropriate color value onto each pixel of the rendered image based on the camera's perspective.

3.2 Class labels

We worked with an industry expert to define eleven class labels for the segmentation task, shown in table 1, along with pixel- and case-level distributions of class prevalence among images generated from the four in-distribution spacecraft.

Table 1: Semantic space and distribution

		% pixels	% cases
0.	None / background	93.95%	100%
1.	Solar panels	1.75%	100%
2.	Solar panel drive shaft	0.03%	49.99%
3.	Antenna	0.05%	73.30%
4.	Parabolic reflector	0.08%	25.00%
5.	Main module	2.90%	100%
6.	Telescope	0.58%	25.00%
7.	Main thrusters	0.05%	77.03%
8.	Rotational thrusters	0.02%	99.99%
9.	Sensors	0.53%	98.27%
10.	Launch vehicle adapter	0.08%	49.44%

Our objective was both to have a single meaningful semantic space that covered a variety of spacecrafts and configurations, and also for the class labels to provide a clear delineation between components to fixate on (e.g., the launch vehicle adapter) and components to avoid (e.g., thrusters) during rendezvous.

3.3 Procedural image generation

We used the Blender API for Python to automate our dataset generation in a scalable way. A Python script moves the camera in a spherical pattern around the spacecraft to one of 5000 positions. For each position, three rendered images were generated with the same aspect, but with different ranges; one at the first position, and one each from approximately twice and three times the distance from the spacecraft, thus creating a total of 15,000 training images from each of the four in-distribution 3D models, for a total of 60,000 image and ground truth pairs. The process was then repeated for the color-coded ground truth 3D model, taking corresponding renders from identical positions in the scene.

3.4 Ground truth representation

The procedurally generated ground truth images were then post-processed using Python's Pillow library to better represent a pixel-wise categorical encoding in the semantic space; we programatically assigned RGB color values in the generated images to their corresponding class labels, resulting in single-channel images with pixel values equal to the cardinal number associated with each category [15]. These single-channel PNG format images were generated both with and without embedded color palettes for human and machine readability.[1]

An example simulated image and ground truth mask can be seen in figures 1 and 2, respectively.

3.5 Out-of-distribution test set

In addition to the 60,000 images used for training, validation, and in-distribution testing, we also generated an additional 1,500 images for out-of-distribution testing.

We chose the Deep Space Program Science Experiment (DSPSE, a.k.a. Clementine) as a spacecraft from which to generate 1,500 synthetic images purely for out-of-distribution testing, in order to evaluate the model's ability to generalize to spacecraft not seen during training.

4 Methods

4.1 Architectures

We trained U-Net, HRNet, and DeepLab deep image segmentation models to determine which architecture performed the best for this task. All models were trained using Python's FastAI and SemTorch (an image segmentation library using PyTorch) libraries [7] [18] [16], to enable rapid testing of different model designs. In each case, a backbone pre-trained on ImageNet [6] was incorporated to leverage transfer learning in extracting features from the input image. [6].

4.1.1 U-Net

Ronneberger et al. [17] describes U-Net, which aims to provide precise localization even when using a smaller dataset than is typically used for image segmentation tasks. SemTorch [7] uses PyTorch's [16] implementation of U-Net.

For the U-Net model we trained, we selected a ResNet34 backbone. A batch size of 8 was chosen as the maximum feasible batch size for this model, given the GPU that was used.

4.1.2 HRNet

HRNet (High-Resolution Net) is a CNN developed specifically to retain and use high-resolution inputs throughout the network, resulting in better performing

[1] The PNG image format encompasses somewhat arbitrary color spaces by allowing a single-channel image to encode values from an arbitrary color palette.

for pixel labelling and segmenation tasks [21]. HRNet aims to provide high spatial precision, which is desirable in this task due to the variety of classes and class imbalance [21].

We selected a pre-trained HRNet30 backbone to perform feature extraction. A batch size of 16 was chosen as the largest feasible batch size given available hardware.

4.1.3 DeepLab

DeepLab is a CNN developed and open-sourced by Google that relies heavily on Atrous Convolution to perform image segmentation tasks [5]. More specifically, we used the latest iteration of the DeepLab model at time of writing, DeepLabv3+, as implemented by FastAI. DeepLabv3+ aims to incorporate the best aspects of spatial pyramid pooling and encoder-decoder models that leads to a faster and more performant model overall [5].

In this case, a ResNet50 backbone was selected over ResNet34 due to limitations with the SemTorch library. A batch size of 16 was chosen by the same criteria as before.

4.2 Loss functions

We experimented with three different loss functions; categorical cross-entropy loss, Dice loss, and a mixture of focal and Dice losses, such choices being motivated by the considerable class imbalance in the data shown in table 1.

4.2.1 Categorical cross-entropy loss

For each pixel, this function computes the log loss summed over all possible classes.

$$\text{CCE}_i = -\sum_{\text{classes}} y \log(\hat{y})$$

This scoring is computed over all pixels and the average taken. However, this loss function is susceptible to class imbalance. For unbalanced data, training might be dominated by the most prevalent class.

4.2.2 Dice loss

The Dice loss function is derived from the Sørensen-Dice Coefficient (see §4.3.1), which is robust to class imbalance as it balances between precision and recall [19].

$$(\text{Dice loss})_i = 1 - \frac{\sum_{\text{classes}} (\text{Dice coef.})_{\text{class}}}{\#\text{ classes}}$$

4.2.3 Dice + focal loss

Focal loss [11] modifies the pixel-wise cross-entropy loss by down-weighting the loss of easy-to-classify pixels based on a hyperparamter γ, focusing training on more difficult examples. The loss is given by:

$$(\text{Focal loss})_i = -\sum_{\text{classes}} (1-\hat{y})^{\gamma} y \log(\hat{y})$$

Dice + focal loss blends Dice and focal loss with a mixing parameter α applied to the focal loss, balancing global (Dice) and local (focal) features of the target mask. We used the default values of $\gamma = 2$ and $\alpha = 1$ during training.

4.3 Model training

We trained nine image segmentation models for this task, one with each combination of architecture and loss function described above, each on the same randomly selected 49,864 images sampled from our 60,000 image dataset, with a validation set of 5306 similarly selected images, and a test set of 6000. The input and output layers were fixed at a size of 256×256, and input images were normalized using statistics from ImageNet [6].

A narrow range of learning rates was selected for each experiment by varying the learning rate over a small number of training batches using FastAI's `lr_find()` method and selecting a region of greatest descent for the loss. Once this region was selected, learning rate annealing was used during Adam optimization with otherwise default parameters. Each model was trained for five epochs, with early stopping at a patience of two; though the loss appeared to plateau in all cases, the early stopping criterion was met in none.

Data augmentation (flip with $p = 0.5$, transpose with $p = 0.5$, rotate with $p = 0.4$) was applied to each batch during training.[2] Weight decay was set to $1\text{e}{-2}$ and batch normalization was used.

4.3.1 Model selection criterion

We chose the Sørensen-Dice coefficient as our final model selection criterion on test data. The Dice coefficient, equivalent to F1 score, is computed pixel-wise between the predicted and target mask where

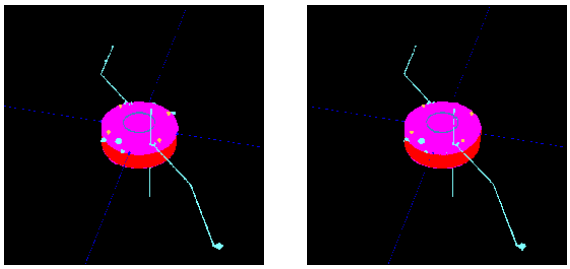
$$\text{Dice coef.} = \frac{2TP}{2TP + FP + FN}.$$

For an aggregate measure of model performance, the Dice coefficient is computed for each class and the arithmetic mean is taken [14].

4.4 Out-of-distribution test

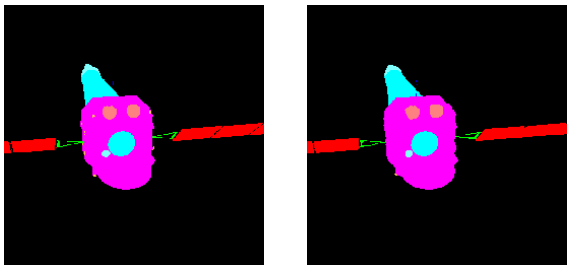
After selecting a model based on our ultimate criterion, we evaluated its performance on the 1500 images from the out-of-distribution test set described in §3.5.

[2] Augmentation was performed using Python's `albumentation` library [1].



(a) True mask (b) Predicted mask

Figure 3: Example true and predicted masks, Cluster II



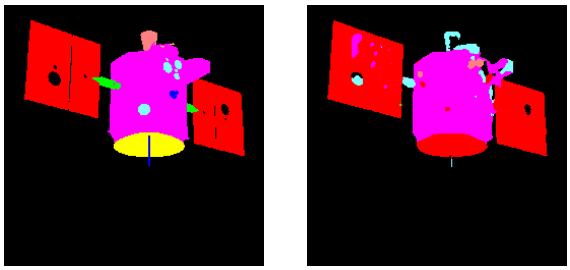
(a) True mask (b) Predicted mask

Figure 4: Example true and predicted masks, Chandra

5 Results

Table 2 shows Sørensen-Dice coefficients for each trained model on validation, in-distribution test, and out-of-distribution (Clementine spacecraft) test sets. Performance on spacecraft present in the training and validation sets is consistently reasonable, however, all of the trained models struggle on the out-of-distribution spacecraft. The U-Net model with categorical cross-entropy loss met our final model selection criterion.

Figures 3a through 4b show example ground truth and model-predicted masks from the in-distribution test images for the selected model; figures 5a and 5b show an example on the out-of-distribution test spacecraft.



(a) True mask (b) Predicted mask

Figure 5: Example out-of-distribution true and predicted masks, Clementine

Table 3 shows per-class Sørensen-Dice coefficients from the selected model on the in-distribution and out-of-distribution test data. Categories 6 (Telescope) and 10 (Launch vehicle adapter) are not present on the Clementine spacecraft.

6 Conclusions

Our initial results on these synthetic data do show some promise for semantic image segmentation in this domain, and that that deep architectures for semantic segmentation can learn to recognize many different spacecraft components and categorize them appropriately by type.

Even with a high degree of class imbalance, Dice loss and Dice + focal loss did not always lead to an improvement in model performance, perhaps owing to CCE loss having a smoother gradient than that of Dice loss, resulting in a less noisy descent path during optimization. [2].

Our out-of-distribution test shows that the selected model does not generalize beyond the four main spacecraft in our dataset for every type of spacecraft component; our selected model is able to identify those larger components (main module, solar panels) of the Clementine spacecraft which have a variety of configurations in the training data, but did overfit to smaller components (sensors, thrusters, etc.), possibly by recognizing a limited number of examples of each. Notably, the selected model mis-classified the parabolic reflector for the Clementine spacecraft almost entirely (Dice < 0.001); our synthetic data contained only one example of this type of component, on the NEAR spacecraft, which the selected model was able to classify quite well on test data (Dice $\approx .95$).

Thus, it would appear that semantic image segmentation for autonomous rendezvous is achievable with these methods if the target spacecraft is known, pending further analysis with physical simulation and on-orbit testing. Image segmentation for arbitrary targets may still be achievable with more representative training data, or perhaps lower variance segmentation models. Shallower architectures may merit exploration, since this domain likely has a much smaller collection of salient features, and a much smaller semantic space, than more general segmentation tasks.

Further work could be done to make our synthetic dataset more representative of the total distribution of unmanned spacecraft; we have based these data on four publicly available examples.

Nonetheless, we here demonstrate an innovative approach to data synthesis for domain-specific semantic

Table 2: Comparison of model results

Model architecture	Backbone	Loss function	Sørensen–Dice coefficient		
			Validation	Test; in-distribution	Test; out-of-distribution
DeepLab	ResNet50	CCE	0.7703	0.7689	**0.2519**
		Dice	0.7597	0.7593	0.2449
		Dice + focal	0.7869	0.7871	0.2502
HRNet	HRNet_w30	CCE	0.7618	0.7618	0.2342
		Dice	0.7712	0.8043	0.2404
		Dice + focal	0.7878	0.7886	0.2371
U-Net	ResNet34	CCE	0.8707	**0.8723**	0.2282
		Dice	0.4423	0.4422	0.2284
		Dice + focal	0.8389	0.8395	0.2357

Table 3: Per-class Sørenson-Dice, U-Net w/ CCE loss

	Class	Sørensen–Dice coefficient	
		Test; in-distribution	Test; out-of-distribution
0.	None / background	0.9991	0.9972
1.	Solar panels	0.9836	0.7673
2.	Solar panel drive shaft	0.7142	0.0259
3.	Antenna	0.6481	0.0003
4.	Parabolic reflector	0.9493	0.0009
5.	Main module	0.9838	0.6678
6.	Telescope	0.9875	NA
7.	Main thrusters	0.8707	0.0100
8.	Rotational thrusters	0.6404	0.0046
9.	Sensors	0.9486	0.0131
10.	Launch vehicle adapter	0.8699	NA

image segmentation tasks which could be applied to similar problems.

7 Contributions

Jesse Trutna of Stanford's Space Rendezvous Lab guided us to pursuing a project in this domain. Kevin Okseniuk, a system test engineer at Momentus, generously assisted the group by helping define our class labels as well as teaching us how to recognize the components they correspond to. Otherwise, all group members contributed equally.

8 Code for this paper

This paper has a github repository containing Jupyter notebooks with ML pipeline and model training code, Python scripts for procedural dataset generation, and Blender files of the 3D models that were used.

References

[1] *Albumentations Python Library v8.4.0*. en. Nov. 2021. URL: https://albumentations.ai/ (visited on 11/29/2021).

[2] Nantheera Anantrasirichai and David Bull. "DefectNET: multi-class fault detection on highly-imbalanced datasets". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 2481–2485.

[3] Gilberto Arantes et al. "Far and proximity maneuvers of a constellation of service satellites and autonomous pose estimation of customer satellite using machine vision". en. In: *Acta Astronautica* 66.9 (May 2010), pp. 1493–1505. ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2009.11.022. URL: https://www.sciencedirect.com/science/article/pii/S0094576509005724 (visited on 10/06/2021).

[4] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Blender Institute, Amsterdam. URL: http://www.blender.org.

[5] Liang-Chieh Chen et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *arXiv:1802.02611 [cs]* (Aug. 2018). arXiv: 1802.02611. URL: http://arxiv.org/abs/1802.02611 (visited on 11/30/2021).

[6] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[7] *Fastai Python Library v2.5.3*. en. Nov. 2021. URL: https://docs.fast.ai/ (visited on 11/29/2021).

[8] Swarnendu Ghosh et al. "Understanding Deep Learning Techniques for Image Segmentation". en. In: *arXiv:1907.06119 [cs]* (July

2019). arXiv: 1907.06119. URL: http://arxiv.org/abs/1907.06119 (visited on 10/07/2021).

[9] Shehzeen Hussain et al. "Satellite Pose Estimation using Convolutional Neural Networks". en. In: (), p. 8.

[10] Mate Kisantal et al. "Satellite Pose Estimation Challenge: Dataset, Competition Design, and Results". In: *IEEE Transactions on Aerospace and Electronic Systems* 56.5 (Oct. 2020). Conference Name: IEEE Transactions on Aerospace and Electronic Systems, pp. 4083–4098. ISSN: 1557-9603. DOI: 10.1109/TAES.2020.2989063.

[11] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[12] Shervin Minaee et al. "Image Segmentation Using Deep Learning: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3059968.

[13] NASA. *Models | 3D Resources*. URL: https://nasa3d.arc.nasa.gov/models (visited on 10/03/2021).

[14] Juri Opitz and Sebastian Burst. "Macro f1 and macro f1". In: *arXiv preprint arXiv:1911.03347* (2019).

[15] *Pillow Python Library v8.4.0*. en. Nov. 2021. URL: https://pillow.readthedocs.io/en/stable/ (visited on 11/29/2021).

[16] *PyTorch Python Library v1.10.0*. en. Nov. 2021. URL: https://pytorch.org/docs/stable/index.html (visited on 11/29/2021).

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv:1505.04597 [cs]* (May 2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (visited on 11/06/2021).

[18] *SemTorch Python Library v0.1.1*. en. Nov. 2021. URL: https://pypi.org/project/SemTorch/#description (visited on 11/29/2021).

[19] Carole H Sudre et al. "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations". In: *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017, pp. 240–248.

[20] Michael Treml et al. "Speeding up Semantic Segmentation for Autonomous Driving". en. In: (Oct. 2016). URL: https://openreview.net/forum?id=S1uHiFyyg (visited on 10/07/2021).

[21] Jingdong Wang et al. "Deep High-Resolution Representation Learning for Visual Recognition". In: *arXiv:1908.07919 [cs]* (Mar. 2020). arXiv: 1908.07919. URL: http://arxiv.org/abs/1908.07919 (visited on 11/30/2021).

[22] Matthew Z. Wong et al. "Synthetic dataset generation for object-to-model deep learning in industrial applications". en. In: *arXiv:1909.10976 [cs]* (Sept. 2019). arXiv: 1909.10976. URL: http://arxiv.org/abs/1909.10976 (visited on 10/06/2021).