(Preprint) AAS 20-126

AUTONOMOUS ON-ORBIT OPTICAL NAVIGATION TECHNIQUES FOR ROBUST POSE-ESTIMATION

Thibaud Teil[*], Samuel Bateman[†], Hanspeter Schaub[‡]

This paper seeks to improve image acquisition and processing methods through the use of Neural Networks. Flight Software algorithms must be developed for robust onboard use as these algorithms interact heavily with the space environment. Testing them reliably therefore requires high-fidelity and computationally fast simulations that allow simulated crafts to navigate visually. Raw camera images are often noisy, imperfect, and contain artifacts. Therefore modeling images requires knowledge of the environment and of the camera. A fast and reliable OpNav simulation opens the door to training Convolutional Neural Networks as it can provide numerous images paired with truth data. The caveat being that both the training environment and the resulting neural network need to be tested and vetted to ensure applicability to real missions.

The proposed research will focus on navigation in proximity to known celestial bodies rather than in a deep space cruise. The scenario used is a spacecraft on orbit around Mars. In this situation, the baseline method for orbit determination is based on Center and Apparent Diameter (CAD) measurements. The primary goal of this research is to assess the accuracy of existing methods and compare them to new Neural Networks pre-trained in the *Basilisk-Vizard* environment. The results demonstrate the ability for neural network based image processing for space applications and display the robustness of the navigation solution.

INTRODUCTION

Spacecraft Navigation and Autonomy

Autonomous Optical Navigation (OpNav) leads to an increase in spacecraft autonomy as it provides measurements that can be gathered without contacting Earth.[1] Aside from lightening the load on ground-based tracking, autonomy can be a back-up when communication failures occur or when ground-in-the-loop navigation is impossible given maneuver time and spacecraft distance.[2]

Autonomy relies on different techniques depending on the mission to which it is applied. For instance, low earth orbit missions have access to Global Position System (GPS) measurements, as well as telemetry data from various sources. On the other hand, deep space missions rely heavily on range and range rate measurements from the Deep Space Network (DSN). Yet there is increasing demand to use the network with limited bandwidth and operational capabilities. OpNav uses images of visible celestial bodies to extract position and velocity information. Therefore, it can rely solely

[*]Graduate Student, Aerospace Engineering Sciences, University of Colorado Boulder.
[†]Undergraduate Student, Computer Science and Applied Mathematics, University of Colorado Boulder
[‡]Professor, Glenn L. Murphy Chair, Department of Aerospace Engineering Sciences, University of Colorado, 431 UCB, Colorado Center for Astrodynamics Research, Boulder, CO 80309-0431. AAS Fellow.

on the spacecraft's interaction with the deep space environment instead of relying on Earth contact — provided that the algorithms used are computationally tractable.

To allow for on-board navigation, Flight Software (FSW) algorithms must be developed and provide robust state estimation allowing for failure detection or mitigation. Testing FSW reliably requires high-fidelity and computationally fast simulations that allow simulated crafts to navigate in flight-like conditions. Neural networks can provide powerful image processing solutions that have been under tremendous development in recent years. Developing new processing methods such as neural networks can broaden the capabilities of previous navigation algorithms and enhance autonomy as a whole. Many types of OpNav measurements exist,[3] from centroid and apparent diameter,[4] star occultation,[5] and landmark tracking[6] to Stereo-Photoclinometry[7] (SPC). Each of these have specific application scenarios depending on the object they require to track, and the distance at which they do so. Centroid and apparent diameter measurements, for instance, find the limb of a body and use the knowledge of its actual size and shape. By extracting direction and distance, the spacecraft's location relative to the body can be determined.[8] The work presented focuses on the use of extracting Centroid and Apparent Diameter (CAD) from the images given a visible planet limb.

Yet these measurements are often noisy, imperfect, and sometimes do not fully observe the spacecraft's states. The goal of this work is to enhance the image processing suite for OpNav using neural networks. The simulation and training of CAD regression networks using noisy images and camera models provides robust solutions to the image processing problem.

The paper first develops the network architecture as well as the training methods. The initial results showcase the choices in training data by analyzing performance on clean, noiseless images. Once the network — named MarsNet — is appropriately trained, the final results compare a robust *Hough Circle* implementation with the the network's navigation solution on imperfect data.

The neural network solution is trained and formatted to fit the CAD FSW stack as a novel contribution, but can provide much more flexibility in the types of measurements it can output. Depending on the desired use and situation, the networks can provide a much broader set of applications from de-noising images to direct feature extraction. The results provided demonstrate the ability for neural network based image processing for space applications.

Background on Neural Networks

Neural networks are a structured set of connected, simple processing elements (colloquially called neurons) operating on a given input. Through repeated non-linear transforms, a trained network can work as a linear classifier or learn a regression task. Although originally inspired by biology, the modern iterations have evolved into many distinct forms ranging from simple Feed-Forward Neural Networks (FFNN),[9] to Convolutional Neural Networks (CNN),[10] as well as Boltzmann Machines[11] and Bayesian Networks.[12] In general, these functions are trained by assuming that the spaces from which the input and outputs are pulled are stochastic. By defining a differentiable metric between outputs, an independent identical distribution of the input-output pairs is pulled from the data. In turn, the average derivative of the distribution is calculated to approximate the true gradient (output with respect to the input) to improve the networks function.[13] Theoretically, a neural network with sufficiently large hidden layers — which provide intermediate transformations between the inputs and the output — are universal function approximators.[14]

In the field of image processing and computer vision, CNNs have seen a great success from med-

ical imagery[15] to digit recognition.[16] In navigation, CNNs are recently used for depth mapping[17] and Terrain Relative Navigation (TRN).[18] The field is constantly growing and evolving: developments with autoencoders[19] have shown promising denoising capabilities and super resolution.[20] Recent results display strong feature detection and generalization notably with the use of deep variational autoencoders.[21] Neural networks are also being developed in order to navigate around other spacecraft for detumbling or docking by pairing physical test-beds with simulated environments to enhance performance.[22] This paper inscribes itself in the intersection between computer vision and spacecraft navigation.

A common task for CNN in image processing is to locate and classify all objects in an image and provide bounding boxes, which is known as the object detection task. In the realm of object detection, modern variants of YOLO[23] and Faster R-CNN[24] are the state of the art: YOLO operates in realtime while Faster R-CNN produces higher quality object detection at a slower speed. These approaches operate on a image feature space produced by a large CNN known as a "backbone", rather then directly on a image. Residual Neural Networks[25] (ResNets) were originally tested and trained on the immense, general classification task ImageNet[26] as classifiers. ResNets allow for much deeper networks and better propagation of signal from the loss resulting in faster learning. Due to richness of their resulting features, ResNets have become nearly ubiquitous in the object detection and localization task as the aforementioned backbone, only losing out in the application to constrained compute environments.[27]

A strong natural image prior is placed on the backbone through training with a larger dataset. By utilizing the feature space of the trained backbone CNN, smaller networks have been shown to learning a different task with a much smaller dataset then would normally be required. This idea of reusing the priors of larger networks to make a new task easier is known as Transfer Learning.[28]

Using only real data to train a CNN for learning for space applications is practically intractable due to the relative sparsity of data and absence of labeled datasets on which to train. Transfer learning can not directly be used either because of the absence of a rich enough prior distribution which would require a large labeled dataset similar to the one to be trained on. However, armed with an accurately simulated environment, a well simulated camera, and faster-than-realtime speeds, CNNs for OpNav are realizable. CNNs allow for the processing of raw measurements and can learn the camera parameters as well as environmental parameters such as lightning and planetary features. Although visualizations continue to improve, the space environment is not highly challenging to model: lighting is purely unidirectional with parallel light rays; near celestial bodies are well mapped; and the background is very dark. Furthermore, training networks for control and navigation tasks on synthetic data which is then applied to real data has been previously successful in robotics,[29] giving further credence to the use of synthetic data for network training. With these concepts in mind and the tools developed throughout this paper, a first instantiation of CNNs for CAD extraction is designed at the intersection of ML and spacecraft navigation.

NEURAL NETWORKS FOR ON-ORBIT NAVIGATION

This paper focuses on a low-level task which uses neural networks for image processing. Supervised learning is a task in Machine Learning (ML) which parametrizes models using a training dataset. The full dataset represents a sample of the input distribution of the camera images. When paired with the associated truth data, the data contains the information needed to approximate the function that maps the input space to the output space.

Although supervised learning has been very successful in recent years, it leads to two key limitations. The first limitation is that the quantity of data needed for training is often large and needs to be associated to a set of truth data (which often requires other sensor modalities or human annotation). The second limitation is generalization: if a model "over-fits" the data, then it can perform exceptionally well on the training data while performing much worse on inputs for which it has never trained on (often called test data). This means the model has learned a feature representation which abuses the training dataset without having generalized near the true distribution of images. Hence, the training data set may allow the network to extract undesirable features that do not perform well on new and real data.

Space imaging suffers from both of those issue for supervised learning in general: there are few real images to train on, and images depend largely on the camera parameters which do not necessarily generalize. The nominal solution to poor generalization — data augmentation — is poorly suited to space imaging because flipping, scaling, and rotational modifications may not be realistic. On the other hand, adding noise and artifacts in the image can be seen as a form of data augmentation specific to the OpNav application, while bringing the simulated images even closer to real images. This subsection intends to harness previous work in OpNav to produce new results in the field of ML for OpNav. The designed network is called MarsNet, and aims to output CAD measurements given a raw camera image as an input.

MarsNet Architecture

Within any network paradigm, sets of hidden layers of varying sizes (number of neurons per layer) can be activated through different activation functions such as the sigmoid function, hyperbolic tangent, or Rectified Linear Unit (ReLU). The activation function is responsible for transforming the summed weighted input of each intermediate representation into an output for downstream layers, while introducing a non-linearity to warp the intermediate representation space to represent non-linear functions. The ReLU activation function is a piecewise linear function that either outputs a scaled value of the input if it is positive, or outputs zero. It is the default activation function for many types of neural networks due to ease of training and better performances in a wide variety of applications. Given it's success throughout the literature — notably because it helps solve vanishing gradients[30] — ReLU is almost always the activation function chosen for CNNs.

However, space applications offer a challenging problem which is not seen in traditional applications of CNNs: the fact that often a non-zero part of the image is nearly black other then the noise of the sensor. After normalization, this provides an unusual distribution of pixel intensities as almost all are very weak or negative. For a normal ReLU, this is problematic as it will eliminate almost all the signal in the first layer. For this application, it was found that Leaky ReLU[31] performed well:

$$\mathrm{ReLU}(x) = max\{\lambda x, x\}, \quad 0 < \lambda \ll 1 \tag{1}$$

Leaky ReLU — Equation (1) — doesn't completely eliminate the signal, yet has the same desired traits of the normal ReLU. Furthermore, if the weights become predominantly negative, the slight slope defined by λ transfers what would otherwise be strictly zero signal back to a positive domain and prevent "dead neurons".

Network optimization happens during the back-propagation step and where methods can either be adaptive (as Adam, Adagrad or RMSprop) or stochastic like commonly used Stochastic Gradient Descent (SGD). Although methods like Adam tend to perform well in the initial portion of training it is outperformed by SGD at later stages of training. Despite some downsides, Adam is chosen for

the training of the ResNets implemented given fast training speeds and high performance on sparse datasets thanks to per-parameter step-size.

The model is implemented and trained using *Pytorch*[*] which provides an array of test scripts and implemented methods to facilitate development. The higher level architecture is a series of Down Convolutions — in which not every pixel is sampled in order to decrease the dimensional size of the image while increasing the perceptive field of each feature pixel. The convolutions are input to ResNet Blocks, which are then followed by a linear output layer to provide the prediction. A depiction of the network is pictured in Figure 1.

Each model trained for 40 epochs with a batch size of 14, using a learning rate scheduler which reduces the learning rate after 5 epochs without improvement. The loss function used is Huber loss[32] because it is less sensitive to outlier predictions from the network, particularly in earlier epochs when the prediction error is arbitrarily large in the regression task. This allows to train with a larger learning rate without running into exploding gradients for this problem, similar in the approach to object detection.[33]

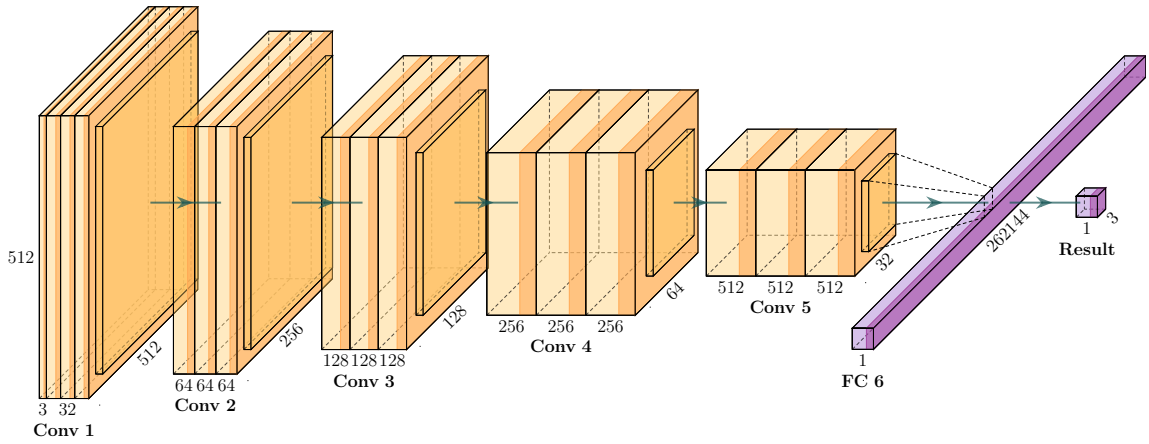


Figure 1: Architecture of MarsNet, each Layer is two ResNet Blocks 2

For each dimension size, three ResNet blocks are applied which each perform the follow sequence twice before summing the input and the output (identity operation central to the ResNet design) :

- Applies a 2D-Convolution to the input
- Activates the convolved input with a "Leaky-ReLU"
- Batch-normalizes the activated layer
- Performs a $\frac{1}{2}$ dropout. In training this means zeroing each neuron with probability one-half, in evaluation weight each neuron output by half to get a cumulative contribution.

The size and number of ResNet blocks are taken from the literature[25] — in order to ensure a efficient and performant implementation. The specific additions are the use of Batch-normalization and Dropout. Batch-normalization makes weight normalization a part of the model architecture as it incorporates it within the network layers. Batch-normalization allows the use of higher learning rates,

[*]https://pytorch.org/

engenders robustness to initializationm and acts as a regularizer.[34] Dropout arbitrarily deactivates neurons in a layer during training. The deactivation of random neurons protects against overfitting by preventing intertwined feature detection. Dropout allows the network to act like a ensemble of distinct models each with different features to rely on, which has been shown to outperform any of the given models.[35]

Simulation Description

This paper builds on previous developments for simulating OpNav. In References 36, 37, the authors develop a open-source simulation that allows for fast, close-loop OpNav scenario simulation. The architecture harnesses two main components: a high-fidelity, faster than real-time, astrodynamics simulation framework *Basilisk*;[38,39] and a sister software package — *Vizard*[40] — to dynamically visualize the simulation environment.[37]

The scenario simulates a spacecraft on an elliptical orbit around Mars as shown in Figure 2. The conditions are designed to allow for variation in the apparent size of the planet. This section specifies the simulation parameters as well as the flight-software algorithms used.

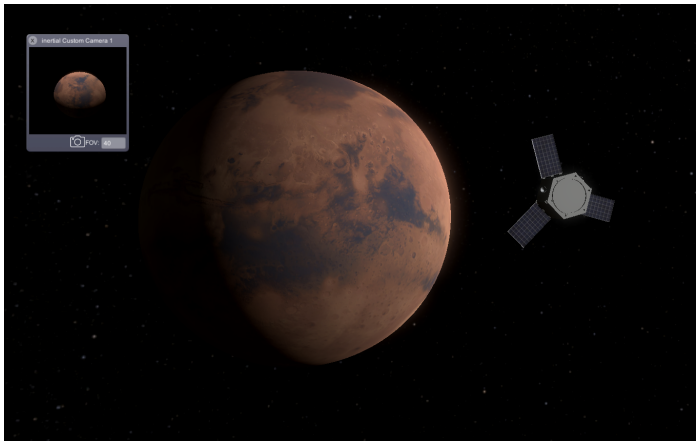


Figure 2: OpNav on Mars Orbit

The simulation uses *SPICE*[*] data, where the simulation begins on December 12th at 22:00 (GMT) 2019. Given the initial conditions of the spacecraft in orbit — seen in Tables 1 and 2 — Mars first appears as a waxing crescent and as the spacecraft reaches apoapse Mars becomes full. Near the end of the simulation Mars begins to go through a waning crescent phase. This allows the FSW algorithms to be tested along a wide variety of lighting conditions and planet sizes.

Table 1: Spacecraft Initial States

$\boldsymbol{\sigma}_{\mathcal{BN}}$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
$\boldsymbol{\omega}_{\mathcal{BN}}$[rad/s]	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
Orbital Elements $(a, e, i, \Omega, \omega, f)$	$(18000\text{km}, 0.6, 10°$ $25°, 190°, 80°)$

[*] `naif.jpl.nasa.gov/naif/`

Table 2: Camera Parameters

$\boldsymbol{\sigma}_{\mathcal{CB}}$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
${}^{\mathcal{B}}\boldsymbol{r}_{\mathcal{CB}}$[m]	$\begin{bmatrix}0 & 0.2 & 0.2\end{bmatrix}^T$
Resolution (pixels)	$\begin{bmatrix}512 & 512\end{bmatrix}^T$
Sensor Size (mm)	$\begin{bmatrix}10 & 10\end{bmatrix}^T$
Field of View (°)	$\begin{bmatrix}55 & 55\end{bmatrix}^T$

The simulation modules assigned to modeling the spacecraft dynamics and environment are described in Table 3. These modules simulate spacecraft attitude gyroscopics and gravity,[41] eclipse, reaction wheels,[42] and star trackers. Eclipsing is also modeled: this usually creates a halt in the spacecraft measurements. In a fully coupled Attitude-OD simulation this means the spacecraft enters a search mode, which intends to allow for the modeling of a fully independent spacecraft. In this scenario, a rough pointing to Mars can be accomplished even with inaccurate knowledge of the spacecraft position. Therefore, in order to focus on the OD solution, the spacecraft goes through a 3 minute guided pointing mode before starting to navigate and point using OpNav.

Table 3: Simulation Parameters

Simulation Modules Instantiated	**Necessary parameters at initialization**
Spacecraft Hub	Inertia $[I] = \mathrm{diag}(900, 800, 600)$ kg·m^2, mass $M = 750$kg
Gravity Effector/Eclipse	December 12th 2019 at 18:00:00.0 (Z) $\mu_{\mathrm{mars}} = 4.28284 \cdot 10^4 \mathrm{km}^3/\mathrm{s}^2$, $\mu_{\mathrm{earth}} = 0.3986 \cdot 10^6 \mathrm{km}^3/\mathrm{s}^2$, $\mu_{\mathrm{jup}} = 1.26686 \cdot 10^8 \mathrm{km}^3/\mathrm{s}^2$, $\mu_{\mathrm{sun}} = 0.327124 \cdot 10^{11} \mathrm{km}^3/\mathrm{s}^2$
Simple Navigation Star Tracker	Attitude error $\sigma_{\mathrm{att}} = 1/3600°$ Rate error $\sigma_{\mathrm{rate}} = 5 \cdot 10^{-5}°$/s
Reaction Wheel Effector	4 Honeywell HR16 Wheels
Wheel orientations	Elevation Angles (el): $40°$ Azimuths angles (az): $45°, 135°, 225°, 315°$ Pos in $\mathcal{B}$ [m]: $\begin{bmatrix}0.8, 0.8, 1.79070\end{bmatrix}^T$ $\begin{bmatrix}0.8, -0.8, 1.79070\end{bmatrix}^T$ $\begin{bmatrix}-0.8, -0.8, 1.79070\end{bmatrix}^T$ $\begin{bmatrix}-0.8, 0.8, 1.79070\end{bmatrix}^T$

Table 3 defines the spin axes of the wheels[43] in the body frame through the elevation and azimuth angles with the equation:

$$ {}^{\mathcal{B}}\hat{\boldsymbol{g}}_s = \begin{bmatrix}\cos(el)\cos(az) & \cos(el)\sin(az) & \sin(el)\end{bmatrix}^T \tag{2} $$

More specific information on the wheel specifications are available on *Honeywell* documents*.

Beyond the pure simulation modules, the simulation also implements several FSW algorithms. These are all developed in C for speed, and compatibility with heritage FSW. The imaging modules encompass the OpNav raw measurements (limbs and circles) as well as the measurement models to provide spacecraft position. Table 4 shows the modules implemented for centroid-based pointing guidance,[37] OD, and control using MRP-Feedback seen in Example 8.14 of Reference 43.

*aerospace.honeywell.com

Table 4: Flight Software Pointing and Orbit Determination

Flight Software Modules Instantiated	**Necessary parameters at initialization**
OpNav Point	minAngle = 0.001°, timeOut = 100s $\omega_{\text{search}} = [0.06, 0.0, -0.06]^\circ/\text{s}$, ${}^{\mathcal{C}}h_c = [0, 0, 1]\text{m}$
relativeOD	α = 0.02, β = 2, κ = 0, noiseSF = 5 $\boldsymbol{r}_{\text{error}} = [10, 10, -10]\text{km}$ $\boldsymbol{r}_{\text{error}} = [0.1, -0.01, 0.01]\text{km/s}$
MRP Feedback RW	K = 3.5 , P =30 (no integral feedback)
RW motor Torque	Control axes are ${}^{\mathcal{B}}[\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3]$

Training Data

As stated in the introduction, the difficulty with using CNNs for OpNav lies primarily in the data for training. There needs to be an easy way to generate large quantities of realistic training data, and provide a test for the fully integrated algorithm. The simulation used throughout this paper provides the ability to do so with a large control over the noise of the data and the camera parameters.

Table 5: Orbital Data Dispersions

Parameter	Dispersion
Semi-Major Axis (km)	$\mathcal{N}[20,000, 3,000]$
Eccentricity	$\mathcal{U}[0.1, 0.6]$
Inclination (°)	$\mathcal{U}[-80, 80]$
True Anomaly (°)	$\mathcal{U}[0, 359]$
RAAN (°)	$\mathcal{U}[-40, 40]$
Off-pointing	$\sigma = [\mathcal{U}[-0.05, 0.05], \mathcal{U}[-0.05, 0.05], \mathcal{U}[-0.05, 0.05]]$

The dispersions used on the spacecraft orbit and position of the planet on the camera plane are given in Table 5. These cover a wide variety of orbits with inclinations nearly reaching from pole to pole and eccentricities going up to 0.6 to allow for many images to be generated in each episode. Each run therefore generates one hundred images according to the random generation of the orbit and position of the planet on the camera.

Depending on the goal, it is possible to vary camera parameters within the training set randomly as well as all simulation parameters. In the first iteration, the training data did not contain any noise and only dispersed the orbit of the spacecraft as well as the position of the planet on the camera frame.

Figure 3 shows the variety of images that the neural network has to train on. The selected images display some of the 90,000 images that were produced. The truth values for the circles are found by using the inverse transform used for OpNav in previous chapters. The images illustrate the variety of perspectives that can be achieved and the ease with which the data can be generated. It is also important to note that although Mars was chosen as a target, *Vizard* easily allows for other planets or bodies to be added instead.

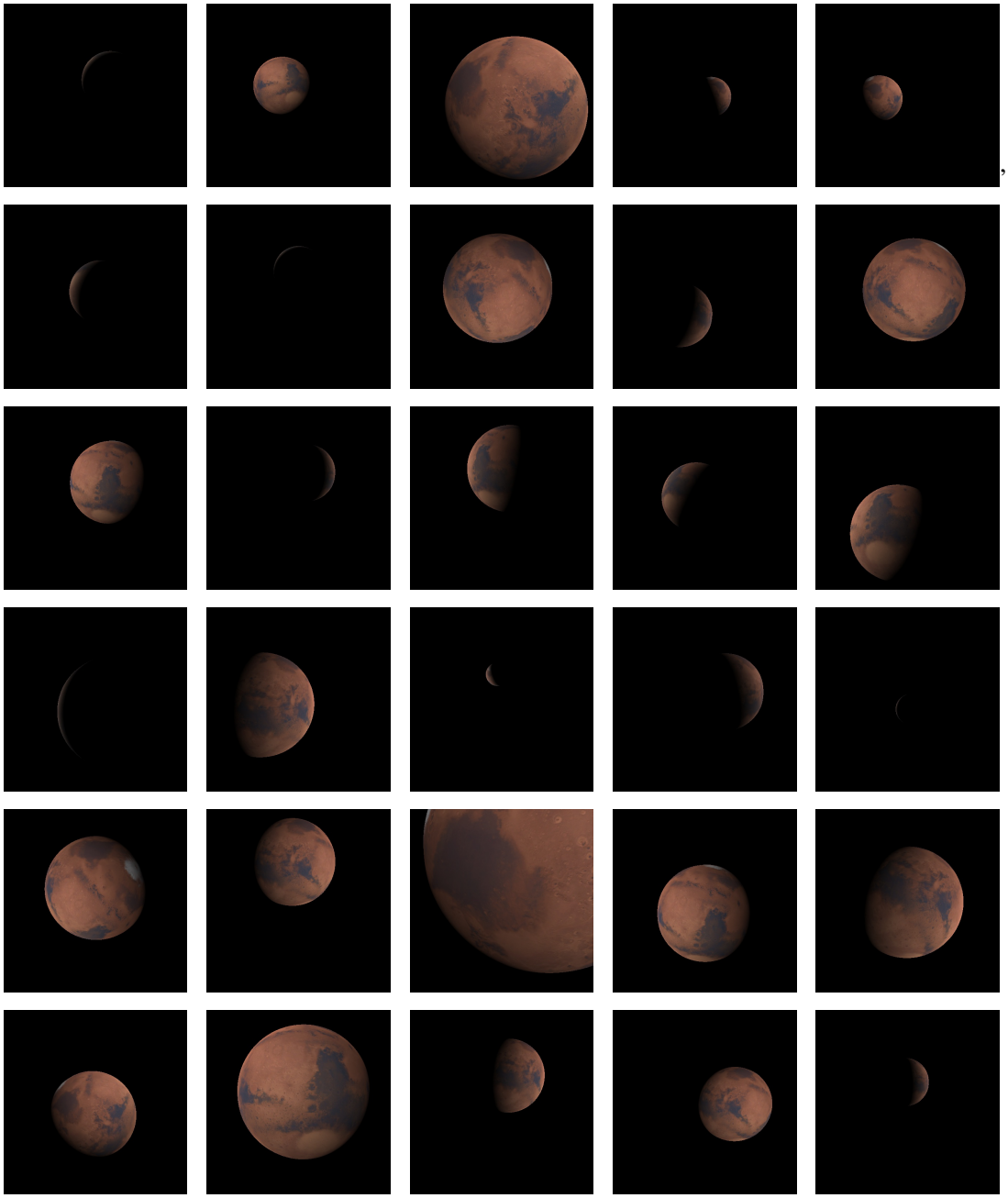


Figure 3: Samples from the CNN training data

NEURAL NETWORK RESULTS

Preliminary Network Evaluation

Once trained the neural network can be tested on a subset of the data (10% isolated into the validation set) that is kept aside. Some of those results are seen in Figure 4. The trained CNN, although a first cut implementation, displays some positive results. Some of the images display very dim if not undetectable limbs to the eye, yet the CNN often produces very accurate and precise

estimates. If no information is in the image, the algorithm guesses the center is in the middle of the sensor and provides a radius roughly equal to half of the image width. This shows that the CNN has learned a bias to the center and mean radius, for which the regression just adds and subtracts from to create the precise localization of the planet.

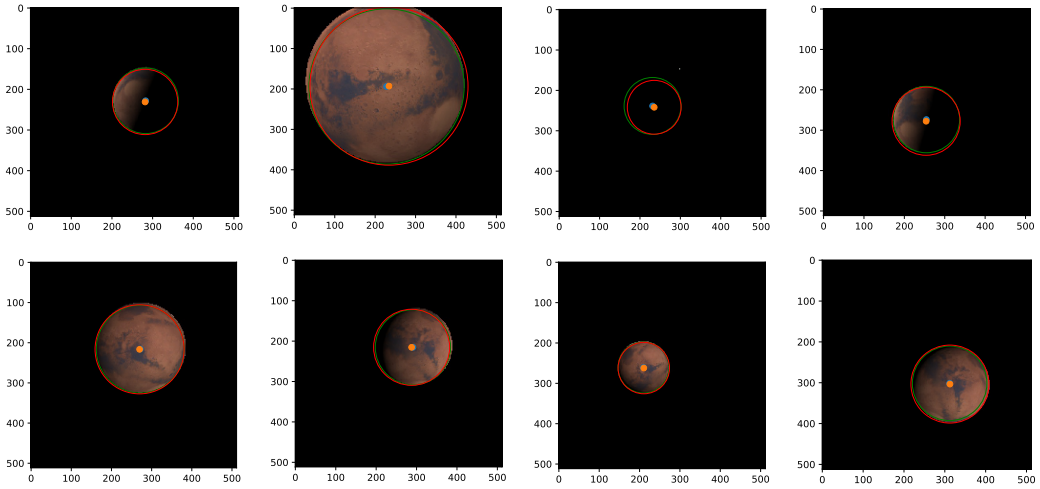


Figure 4: Trained CNN evaluated on new images

Figure 4 shows how fast the CNN was able to learn given the relatively small training dataset. Some limitations in the data set do prevent strong generalization at the moment and will be improved upon. Before flying such an algorithm, it would be key to test it on real images or images with the instrument hardware in the loop. At the current stage of development such results are not available but in progress.

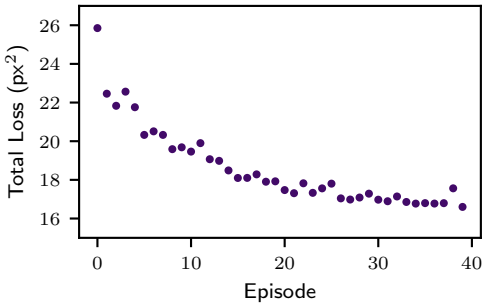


Figure 5: CNN loss during training

Figure 5 shows the total loss during training of the CNN in squared pixels over all episodes. The training was performed by splitting the images in 40 episodes each containing 5500 images. The plot shows the steady decline of the loss on the validation data over the episodes, as is desirable. An ideal plot would be monotone decreasing which, outside a few exceptions, is pictured.

Integration and First Navigation Results

Once the CNN is trained in Python using the generated images, it can be incorporated as a *Basilisk* module through the *OpenCV* DNN library. This allows to load a trained network which

has been exported in the ONNX format, and read in an image to be processed from the simulated sensor. The process is implemented in the `centerRadiusCNN` module found in *Basilisk* under `fswAlgorithms/imageProcessing`.

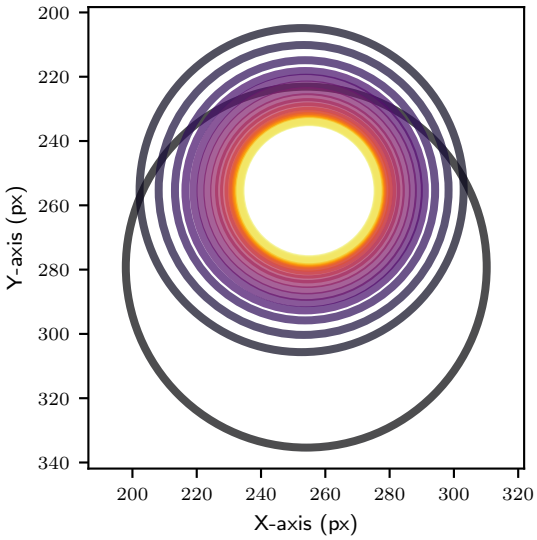


Figure 6: Circles found by the CNN

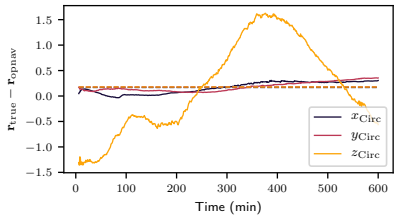
Results from a simulation run with parameters in Tables 1, 3, and 4 are shown in Figure 6. The results are similar to *Hough Cirlces* seen in previous work. All percentages are computed as the norm of the difference between the estimate and the truth, normalized by the truth norm, multiplied by 100. Since the neural network does not output un uncertainty metric, the values are chosen empirically and conservatively:

$$[R_{\text{CNN}}] = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix} \tag{3}$$

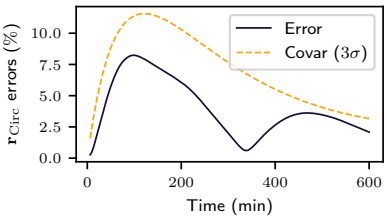
When integrated into a full OpNav stack, the CNN performs well overall. This is seen in Figure 7 as the filter output is of the same order of magnitude as the other methods. The network does show some varying biases that are not akin to those of the other methods. Indeed the measurements seen in subfigure 7a show a large oscillation on the range which does not match the lighting conditions affects seen by *Hough* or *Canny* transforms. This indicates that although the center finding is successful, the radius of the planet is not well matched.

Enhancing Training and Results

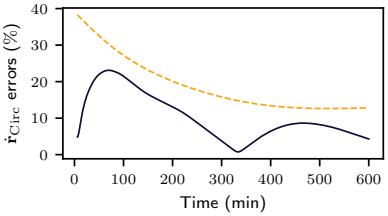
The results presented encouraged more training in an environment that allows for more generalization: done through the addition of image corruptions. By training on a new set of data with imperfect images, the neural network can be expected to better generalize and extract more pertinent features for center and radius finding.



(a) CNN Measurements



(b) CNN Position Errors



(c) CNN Velocity Errors

Figure 7: Measurements and State-Error Covariances for CNN

Table 6: Orbital Monte-Carlo Dispersions

Parameter	Dispersion
Semi-Major Axis (km)	$\mathcal{N}[18,000, 1,000]$
Eccentricity	$\mathcal{U}[0.1, 0.5]$
Inclination (°)	$\mathcal{U}[-40, 40]$
True Anomaly (°)	$\mathcal{U}[-90, 90]$
Off-pointing	$\sigma = [\mathcal{U}[-0.01, 0.01], \mathcal{U}[-0.01, 0.01], \mathcal{U}[-0.01, 0.01]]$
Gaussian Blur	$\mathcal{U}[0, 3]$
Hot-Dead Pixels	$\mathcal{U}[0, 3]$
Cosmic Rays	$\mathcal{U}[0.5, 2]$
Blur	$\mathcal{U}[1, 5]$

Table 6 shows the new set of dispersions used to generate data. Although very similar to the previous set seen in Table 5, these values constrain the images to be closer to the reference orbit. The data is therefore constrained to a smaller torus surrounding the planet, notably by extending less towards the polar orbits. Furthermore, the camera is pointed more directly at the planet which helps to generate more centered data as the pointing scheme will naturally provide much more of these images. This data set therefore aims to target the weaknesses of the previous iteration as it builds on the already working prior.

Most of all, the new data set adds corruptions and noise to the images using previously implemented models.[36] This is done with the aim of enhancing the data and forcing MarsNet to learn useful features rather than overfitting certain components of the image. Having noise and blur also provides more signal in each image to aid optimization, hence partly solving the extremely sparse data problem of space imaging.

The new set of data improved the loss during training greatly. Staring from the prior of the

previously trained net, the loss continued to decrease monotonously and ending at nearly half of the loss of the previous iteration as seen in Figure 8. Due to greater performance, the covariance attributed to the network was updated to be $[R_{\text{CNN}}] = 5[I]_{3\times3}$.

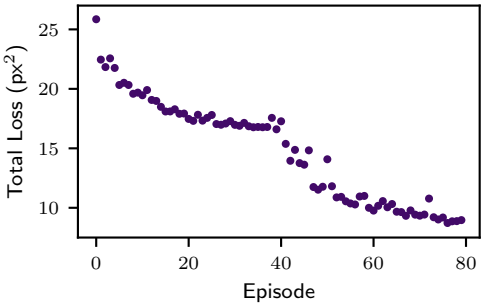
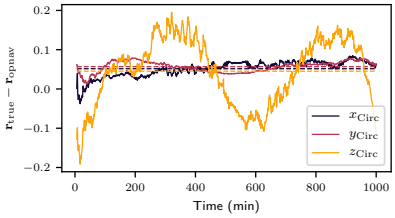
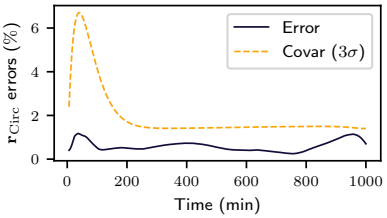


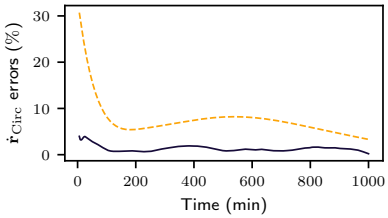
Figure 8: CNN loss with additional training



(a) CNN Measurements



(b) CNN Position Errors



(c) CNN Velocity Errors

Figure 9: Measurements and State-Error Covariances for Enhanced CNN

A few important novelties are interesting with the results for Figure 9. With regard to robustness, the newly trained CNN produces the same results with noisy images or clean images. This stresses the fact that training under corrupt conditions has aided the weights to feature detection to converge towards the key components of the image: the limb and lit disk, as well as specifically removing noise from the image.

One of the novelties of using *Hough Circles* was moving away from ellipses while centering the image in order to benefit from the measurement robustness. The downside, nonetheless, is still that camera distortions naturally falsify the circle positions. The MarsNet trained in this section does not suffer this default: the network trains over expected circle position given the geometry and hence learns the camera perturbations and distortions as a part of it's training. Finally, when looking at fault detection, it became clear that comparing methods with different strengths and weaknesses

would help detect anomalies: if a first method works well with crescents and poorly with full disks while the second performs inversely well, combining them will output reliable faults and results. The CNN trained here shows in Figure 9a that the lighting conditions do not affect performance as it did with other methods, hence providing good supporting data in a multi-measurement paradigm.

The one caveat is that the speed of the network evaluation is one order of magnitude slower on general purpose compute hardware: roughly $100\times$ faster than real-time instead of $1000\times$. This can be improved by decreasing the size of the network through training a smaller network to match this function, though it is not prohibitive for on-board use as is. However, by utilizing a hardware accelerator such as a graphics accelerator or a Field Programmable Gate Array (FPGA), one could easily perform these computations quickly and power efficiently. Furthermore, the use of half-precision floating point or even 8-bit integer arithmetic would enhance speed, use less power, and theoretically show nearly identical regression performance.[44]

COMPARISON TO PREVIOUS METHODS

This section showcases the possible results for autonomous OpNav with a spacecraft taking numerous pictures: 1 image per minute. This approach also enables coupled pointing on a wide range of orbits and permits less accurate methods to perform despite noisy measurements. All of the results shown have the spacecraft perform both duties, given initial conditions that provide the planet in the field of view. Reference 36 develops two methods: Non-Iterative Horizon through Singular Value Decomposition (*NIH-SVD*), and *Hough Cirlces*. The results show that in the developed simulation, *Hough Circles*[45,46] performed similarly to *NIH-SVD*[47] on clean images despite fitting circles rather than ellipses. When noise and artifacts are added to the images, *Hough Circles* proves to be more robust.

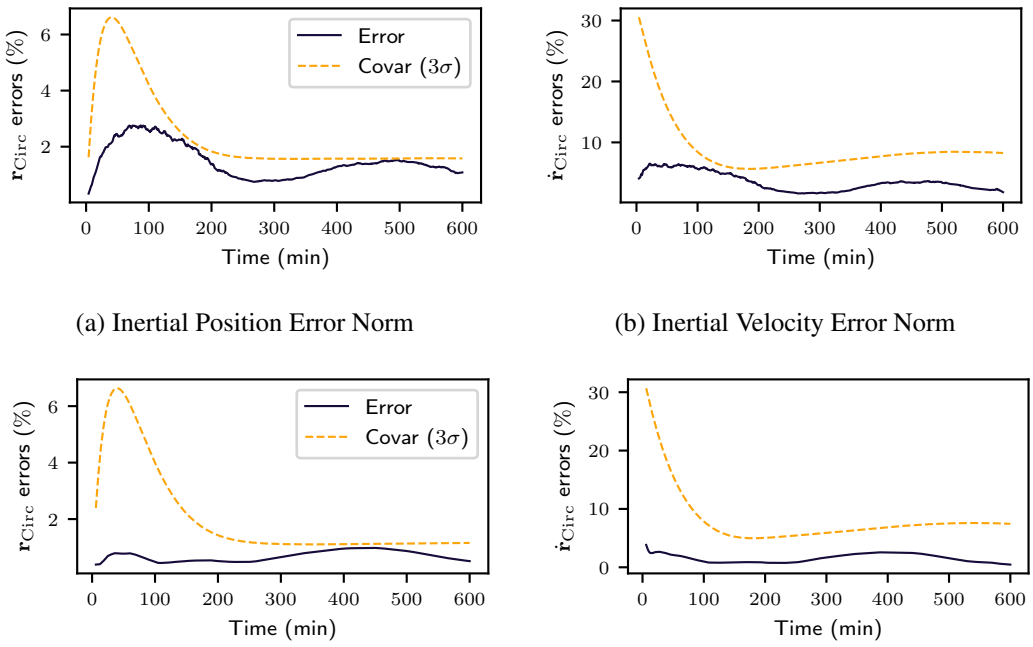
Given the improved performance once again, the results are now shown with a covariance of $[R_{\text{CNN}}] = 2.5[I]_{3\times3}$. It is important to note that the uncertainty metric of the CNNs is an ongoing weakness that requires future work and analysis.

Table 7: Noise Parameters in Simulation

Parameter	Value
Gaussian Blur	2
Hot-Dead Pixels	0.5
Cosmic Rays	1
Blur	3

Figures 10a and 10b show the filter results for the *Hough Circle* method given the initial conditions in Tables 1 and 3 with the noise parameters of Table 7. The position covariance oscillates around 300km ($\sim 2\%$) error and 0.05km/s ($\sim 9\%$) error on velocity, with slightly better performance in the out-of-plane direction. The state estimates have errors in percentages below 1% on position and below 3% on velocity.

Although *Hough Cirlces* performs well given the noise added to the images, Figure 10 shows the CNN performing better than these previous implemented methods under noise. Figures 10c and 10d show the errors within 1% for position and 2% for velocity. The covariances are also lesser than previously notably on the velocity components. Root-Mean-Squared (RMS) errors are one measure of this overall better performance. Indeed, the CNN-based navigation provides RMS errors on



(a) Inertial Position Error Norm

(b) Inertial Velocity Error Norm

(c) CNN Position Errors

(d) CNN Velocity Errors

Figure 10: Relative Errors — *Hough Cirlces*

the position of 0.72% and 1.73% on the velocity. These numbers are a considerable increase on the *Hough*-based method which shows 1.60% and 3.8% errors on the position and velocity respectively. This outlines an overall better filter performance than the previous MarsNet iteration, and better than both other methods studied.

CONCLUSIONS

The developments CNNs for spacecraft autonomy and enhancing image processing is presented. CNNs are integrated in a full OpNav FSW stack. By training under different camera models and orbits, the CNN has proven to outperform all other methods and has provided a new and robust way of doing image processing. Validation using real images as well as further training are still needed, and different network architectures can still be explored. The growing field of ML is being increasingly applied to aerospace, these results tie in to that body of work.

Two possible directions can be taken to keep training these CNNs, the first being to model the camera more accurately in simulation and train a network to capture its defects as well as possible, the second would aim to use a large set of cameras and generalize the circle fitting as much as possible. Both methods show promise for future on-board applications.

REFERENCES

[1] W. M. Owen, "Optical Navigation Program Mathematical Models," Engineering Memorandum 314-513, Jet Propulsion Laboratory, August 1991.

[2] J. A. Starek, B. Açıkmeşe, I. A. Nesnas, and M. Pavone, *Spacecraft Autonomy Challenges for Next-Generation Space Missions*, pp. 1–48. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, 10.1007/978-3-662-47694-9_1.

[3] W. Owen, "Methods of Optical Navigation," 2011.

[4] J. Christian, "Onboard Image-Processing Algorithm for a Spacecraft Optical Navigation Sensor System," *Journal of spacecraft and rockets*, Vol. 49, No. 2, 2012.

[5] M. Psiaki, "Autonomous Lunar Orbit Determination using Star Occultation Measurements," Guidance, Navigation and Control Conference and Exhibit, Hilton Head, SC, AIAA, August 2007.

[6] A. Liounis, "Autonomous Navigation System Performance in the Earth-Moon System," AIAA Space Conference and Exposition, San Diego, CA, AIAA, September 2013.

[7] O. B. R. Gaskell and E. Kahn, "Assessing the quality of topography from stereo-photoclinometry," 2014.

[8] J. Christian, "Optical Navigation Using Planet's Centroid and Apparent Diameter in Image," *Journal of guidance, control, and dynamics*, Vol. 38, No. 2, 2015.

[9] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and Intelligent Laboratory Systems*, Vol. 39, No. 1, 1997, pp. 43 – 62, https://doi.org/10.1016/S0169-7439(97)00061-0.

[10] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," *CoRR*, Vol. abs/1511.08458, 2015.

[11] I. Sutskever, G. E. Hinton, and G. W. Taylor, "The Recurrent Temporal Restricted Boltzmann Machine," *Advances in Neural Information Processing Systems 21* (D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds.), pp. 1601–1608, Curran Associates, Inc., 2009.

[12] J. Cheng and R. Greiner, "Comparing Bayesian Network Classifiers," *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 1999, pp. 101–108.

[13] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning*. Cambridge University Press, first ed., May 2014.

[14] G. M. Maggiora, D. W. Elrod, and R. G. Trenary, "Computational neural networks as model-free mapping devices," *Journal of Chemical Information and Computer Sciences*, Vol. 32, No. 6, 1992, pp. 732–741, 10.1021/ci00010a022.

[15] F. Milletari, N. Navab, and S. Ahmadi, "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation," *2016 Fourth International Conference on 3D Vision (3DV)*, Oct 2016, pp. 565–571, 10.1109/3DV.2016.79.

[16] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional Neural Networks Applied to House Numbers Digit Classification," *CoRR*, Vol. abs/1204.3968, 2012.

[17] J. M. Mern, K. D. Julian, R. E. Tompa, and M. J. Kochenderfer, *Visual Depth Mapping from Monocular Images using Recurrent Convolutional Neural Networks*, 10.2514/6.2019-1189.

[18] I. B. Roberto Furfaro, "Deep Learning for Autonomous Lunar Landing," *Proceedings of the 2018 AAS/AIAA Astrodynamics Specialist Conference, Snowbird UT*, 2018.

[19] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," 2013.

[20] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *J. Mach. Learn. Res.*, Vol. 11, Dec. 2010, pp. 3371–3408.

[21] I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner, "Early Visual Concept Learning with Unsupervised Deep Learning," 2016.

[22] "Satellite Pose Estimation Challenge: Dataset, Competition Design and Results," 2019.

[23] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *CoRR*, Vol. abs/1804.02767, 2018.

[24] R. Girshick, "Fast R-CNN," *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR*, Vol. abs/1512.03385, 2015.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[27] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *ArXiv*, Vol. abs/1704.04861, 2017.

[28] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, *A Survey on Deep Transfer Learning: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4, 2018, Proceedings, Part III*, pp. 270–279. 10 2018, 10.1007/978-3-030-01424-7_27.

[29] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, and F. Yu, "Joint Monocular 3D Vehicle Detection and Tracking," 2019.

[30] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, Vol. 5, March 1994, pp. 157–166, 10.1109/72.279181.

[31] A. Y. H. Maas, Andrew L. and A. Y. Ng., "Rectifier nonlinearities improve neural network acoustic models," Vol. 30, June 2019.

[32] P. J. Huber, "Robust estimation of a location parameter," *Breakthroughs in statistics*, pp. 492–518, Springer, 1992.

[33] R. B. Girshick, "Fast R-CNN," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[34] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *CoRR*, Vol. abs/1502.03167, 2015.

[35] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, Vol. abs/1207.0580, 2012.

[36] T. Teil, H. Schaub, and D. Kubitschek, "Center and Apparent Diameter Optical Navigation on Mars Orbit," *2nd RPI Space Imaging Workshop*, Saratoga Springs, NY, Oct. 28–30 2019.

[37] T. Teil and H. Schaub, "Software Architecture For Closed-Loop Autonomous Optical Navigation Scenarios," *1st Annual RPI Workshop on Image-Based Modeling and Navigation for Space Applications*, Troy, NY, June 4–5 2018.

[38] J. Alcorn and H. Schaub, "Simulating Attitude Actuation Options Using the Basilisk Astrodynamics Software Architecture," *67th International Astronautical Congress*, Guadalajara, Mexico, Sept. 26–30 2016.

[39] P. W. Kenneally, S. Piggott, and H. Schaub, "Basilisk: a flexible, scalable and modular astrodynamics simulation framework," *7th International Conference on Astrodynamics Tools and Techniques (ICATT), DLR Oberpfaffenhofen, Germany*, 2018.

[40] J. Wood, M. C. Margenet, P. Kenneally, H. Schaub, and S. Piggott, "Flexible Basilisk Astrodynamics Visualization Software Using the Unity Rendering Engine," *AAS Guidance and Control Conference*, Breckenridge, CO, February 2–7 2018.

[41] C. Allard, M. Diaz-Ramos, P. W. Kenneally, H. Schaub, and S. Piggott, "Modular Software Architecture for Fully-Coupled Spacecraft Simulations," *Journal of Aerospace Information Systems*, 2018. (in press), 10.2514/1.I010653.

[42] J. Alcorn, C. Allard, and H. Schaub, "Fully Coupled Reaction Wheel Static and Dynamic Imbalance for Spacecraft Jitter Modeling," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 6, 2018, pp. 1380–1388, 10.2514/1.G003277.

[43] H. Schaub and J. L. Junkins, *Analytical Mechanics of Space Systems*. Reston, VA: AIAA Education Series, 4th ed., 2018, 10.2514/4.105210.

[44] Nvidia, "8-bit Inference with TensorRT," May 2017.

[45] T. Weismuller, D. Caballero, and M. Leinz, *Technology for Autonomous Optical Planetary Navigation and Precision Landing*, 10.2514/6.2007-6173.

[46] D. Wokes and S. Wokes, *Surveying and Pose Estimation of a Lander Using Approximative Crater Modelling*, 10.2514/6.2010-8342.

[47] J. A. Christian and S. B. Robinson, "Noniterative Horizon-Based Optical Navigation by Cholesky Factorization," *Journal of Guidance, Control, and Dynamics*, Vol. 39, 2019/01/17 2016, pp. 2757–2765, 10.2514/1.G000539.